BECKHOFF New Automation Technology

Manual | EN

TS5066

TwinCAT 2 | PLC Remote Synchronisation

2021-10-13 | Version: 1.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

2 Overview

Due to the increasing use of decentralised controllers, time synchronisation of different systems is becoming an increasingly important issue. In systems without identical timebase, cyclic transfer of information can lead to beat effects, which can manifest themselves as periodic disturbance in the synchronisation of drives for which axis information is transferred via a network. The TcRemoteSync.lib library offers options for general time synchronisation of information in distributed systems, and special techniques for synchronising NC axes ("distributed axes").

The problem

Notwithstanding nominally identical task cycle times, different systems will have a small phase difference (drift) that is caused by differences in production, different temperature characteristics and different environmental conditions of the oscillators used. The phase difference causes a continuous change in the temporal interrelationship of the system task cycles. Figure 1 shows the situation in schematic form. The sending system S and the receiving system R operate with slightly different cycle times T. $T_S > T_R$ for the case shown. The phase difference causes a shift in the task cycles of the two systems. Periodically, two cycles of system R will occur during one cycle of system S. This situation is indicated by a circle in Figure 1.

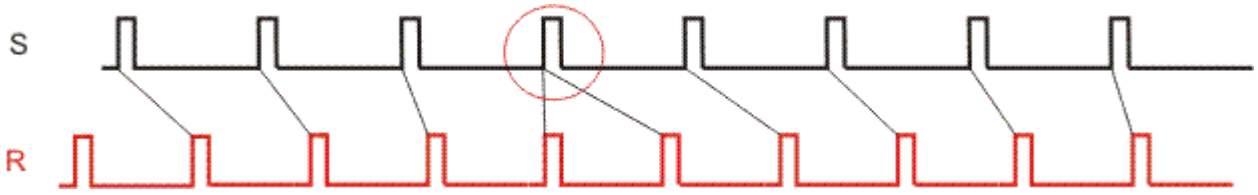


Fig. 1: Figure 1: Change in the temporal interrelationship between the task cycles of two systems S and R with slightly different cycle times (TS>TR).

Distributed systems are synchronized through cyclic sending of information from the sender (S) to the receiver (R). Up to the point indicated in Figure 1 the sent information is receive sequentially, then the information sent is read during two consecutive cycles, and subsequently the system returns to sequential reception. If the cycle time relationship is reversed (i.e. cycle time T_R > cycle time T_S), information sent would be missed on the receiver side.

The solution

If there is no option for physically synchronizing the cycle times of the systems, e.g. for two PCs operated with SERCOS cards, beat effects can be compensated through correction of the information received. The required correction time Δt is determined from the drift of the two systems such that the corrected data S' correspond to temporally equidistant and sequential reception. Figure 2 shows the time response of the original data S to receiver R, and data S' corrected by the correction time Δt. At the point indicated with a circle, the data originally read identically by two receiver cycles are initially corrected with Δt_1 for the first receiver cycle and subsequently with Δt_2 for the second cycle.

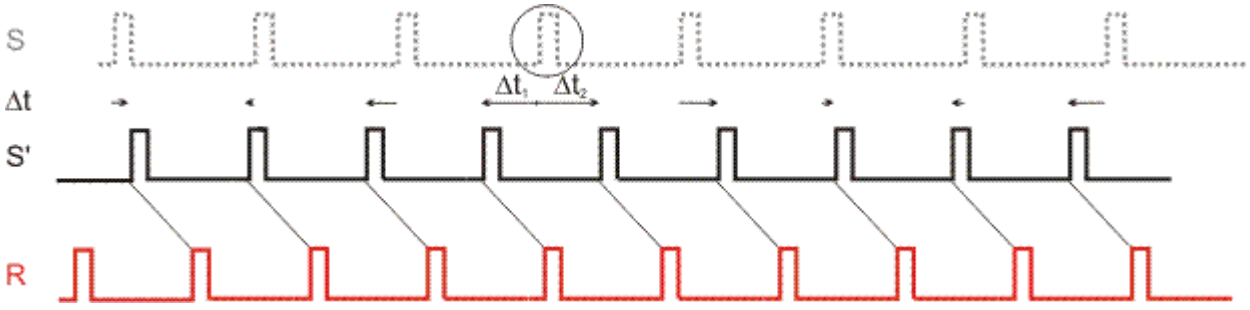


Fig. 2: Figure 2: Corrected information S' after extrapolation of the originally sent information S with correction time Δt. The sign and magnitude of the time correction are indicated by arrows.

In order to be able to correct the data, the variation of the information over time must be known. For the typical application of synchronization of distributed NC axes, for example, set position and velocity can be corrected through extrapolation with the set velocity and acceleration.

Application example

A typical application for the synchronization of distributed systems would be operation of interpolating or coupled NC axes (e.g. master/slave coupling) on different computers ("distributed axes"). Step changes in the set values of the position and velocity caused by duplicate reading or missed information lead to periodic malfunction. Figure 3 shows the actual velocity of a drive operated directly with the received set position (position interface). The disturbances are caused by step changes in the received set positions.

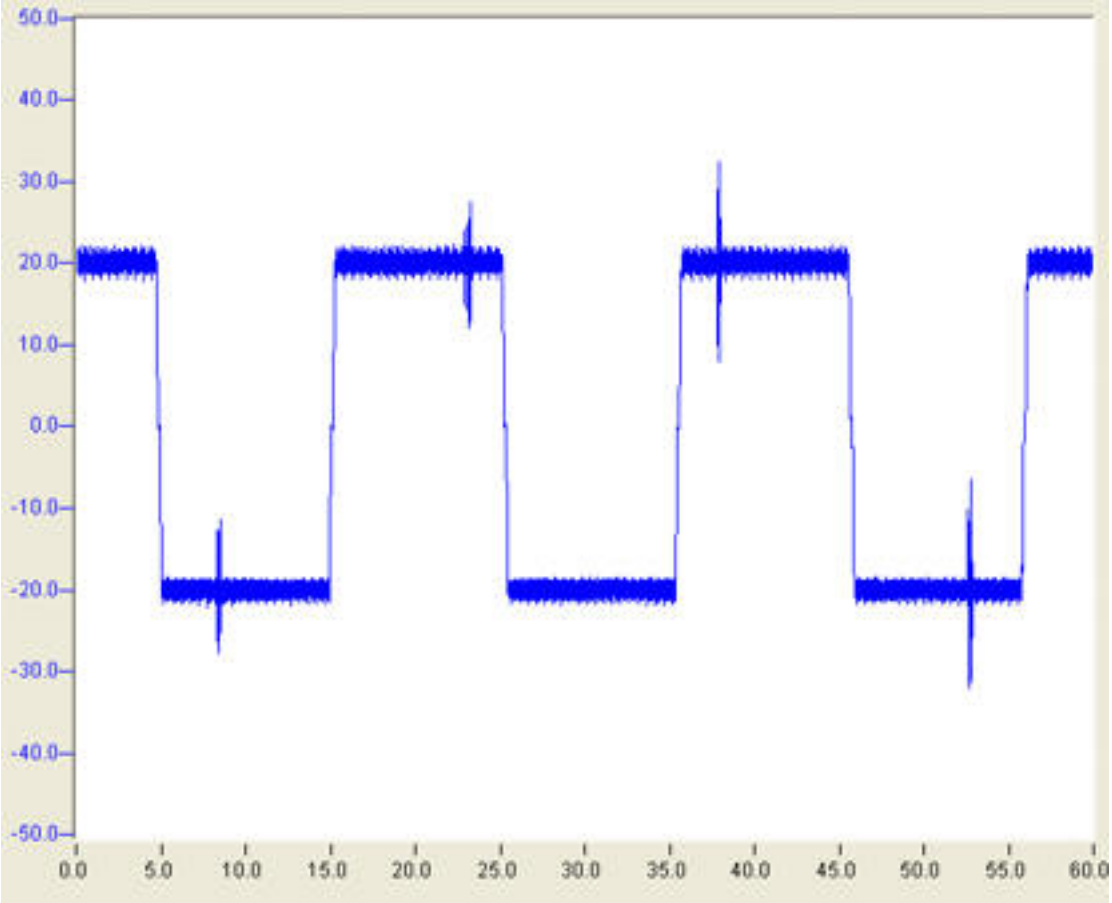


Fig. 3: Figure 3: Actual velocity of an axis operated directly with received set values in the position interface.

The set position can be corrected in the receiver system prior to drive control commands being issued using the method described above. To this end the received set position p_S is extrapolated with the correction time Δt and the received set velocity v_S to set position $p_{S'}$. Optionally, the set acceleration a_S can be used for the extrapolation. The following applies: $p_{S'}=p_S+\Delta t*v_S(+0.5*\Delta t^2 a_s)$. Figure 4 shows the actual velocity of the axis driven with the corrected set position $p_{S'}$. The disturbances shown in Figure 3 are no longer present.

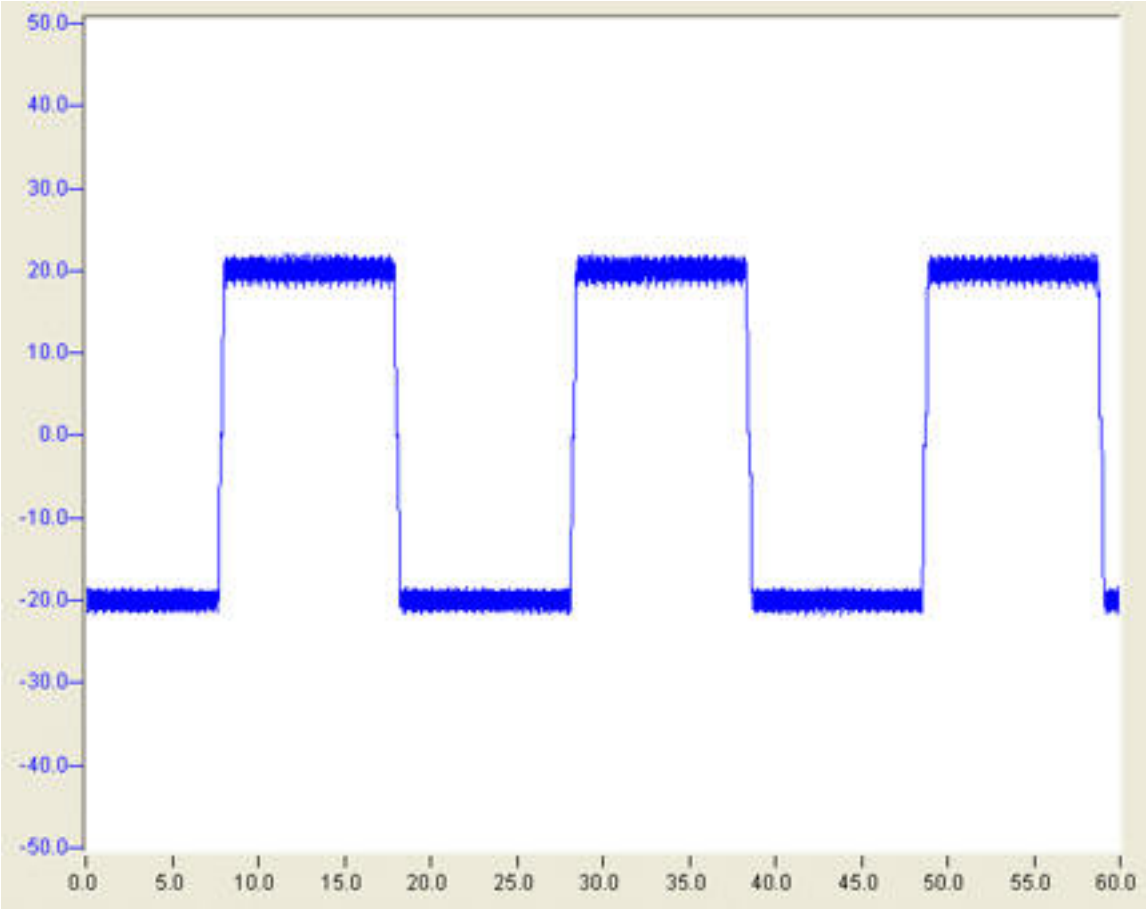


Fig. 4: Figure 4: Actual velocity of an axis operated with corrected set values in the position interface.

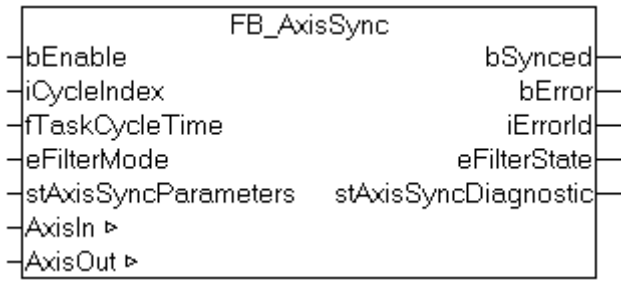
Functions offered by library TcRemoteSync.lib

For synchronising distributed systems, the functionality implemented in the TcRemoteSync.lib library can be used. The following function blocks are used in the receiver system:

The function block FB_AxisSync [▶ 10] is used for axes synchronisation via a network. The set position and, if necessary, the set velocity of the axis information sent, are extrapolated as outlined above. An example [▶ 28] is available for demonstration purposes. Internally FB_AxisSync uses two function blocks, FB_TimeSync and FB_AxisExtrapolateValues. The block FB_TimeSync [▶ 14] is used for determining the beat frequency and for calculating the correction time Δt, FB_AxisExtrapolateValues [▶ 17] extrapolates the set values received with the correction time.

3 PLC API

3.1 FB_AxisSync



This function block is used for synchronizing NC axes via a network ("distributed axes") and for avoiding associated beat effects. Internally the function blocks FB_TimeSync [▶ 14] and FB_ExtrapolateAxisValues [▶ 17] are used. These function blocks can also be used individually.

General

When NC axes are synchronized via a network, information from one axis (sender) is typically distributed cyclically to other axes (receivers) in the form of network variables. If the receivers are operated directly with the transferred information, beat effects caused by small phase differences in the real-time clocks become apparent. The beat effects manifest themselves as cyclic disturbance of the receiver axes, since information may be analyzed twice or skipped, depending on the phase difference. The frequency and duration of the disturbances depends on the phase difference and the jitter of the individual real-time clocks. Figure 1 shows the actual velocity of an axis operated directly via a network variable (position interface). The disturbances caused by beat effects due to step changes in the received set positions are clearly visible.

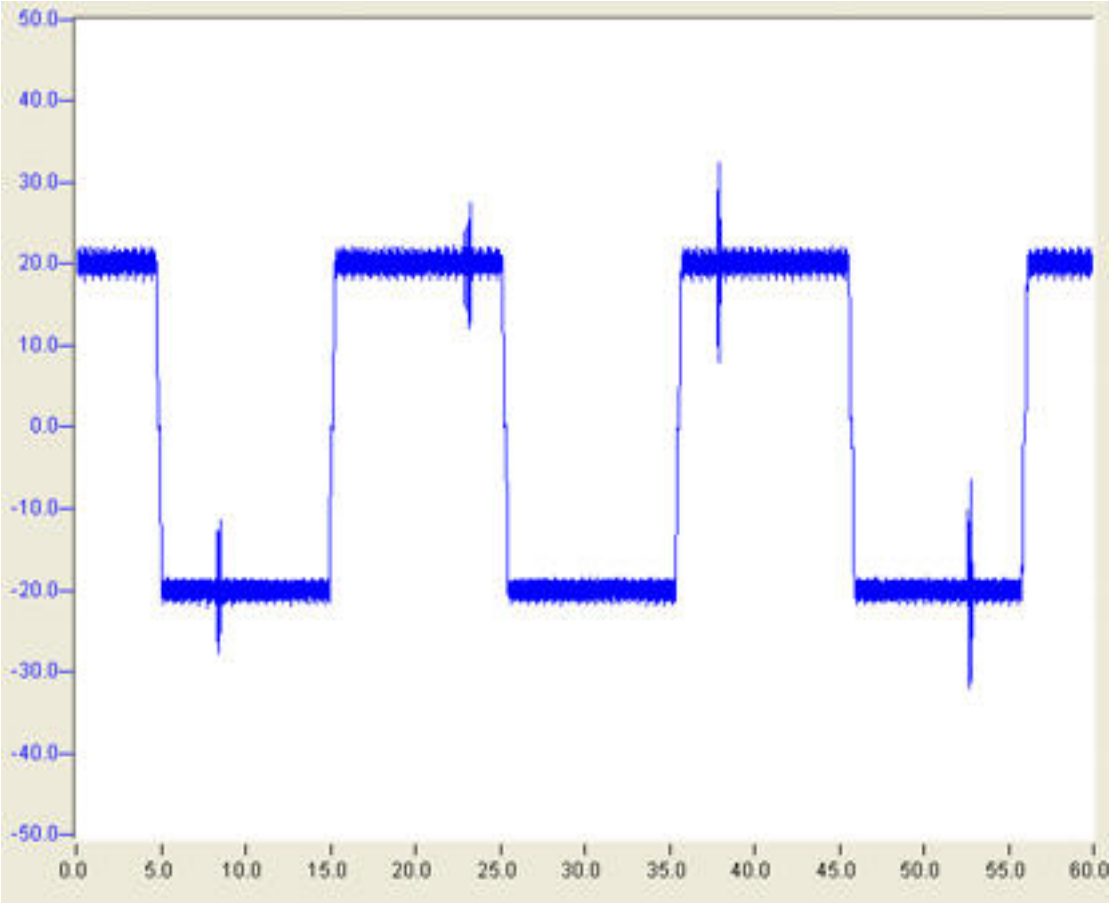


Fig. 5: Figure 1: Actual velocity of an axis directly operated with the received axis information (position interface).

Figure 2 shows the actual velocity of an axis operated with values corrected through this function block. The previously observed disturbances have been corrected.

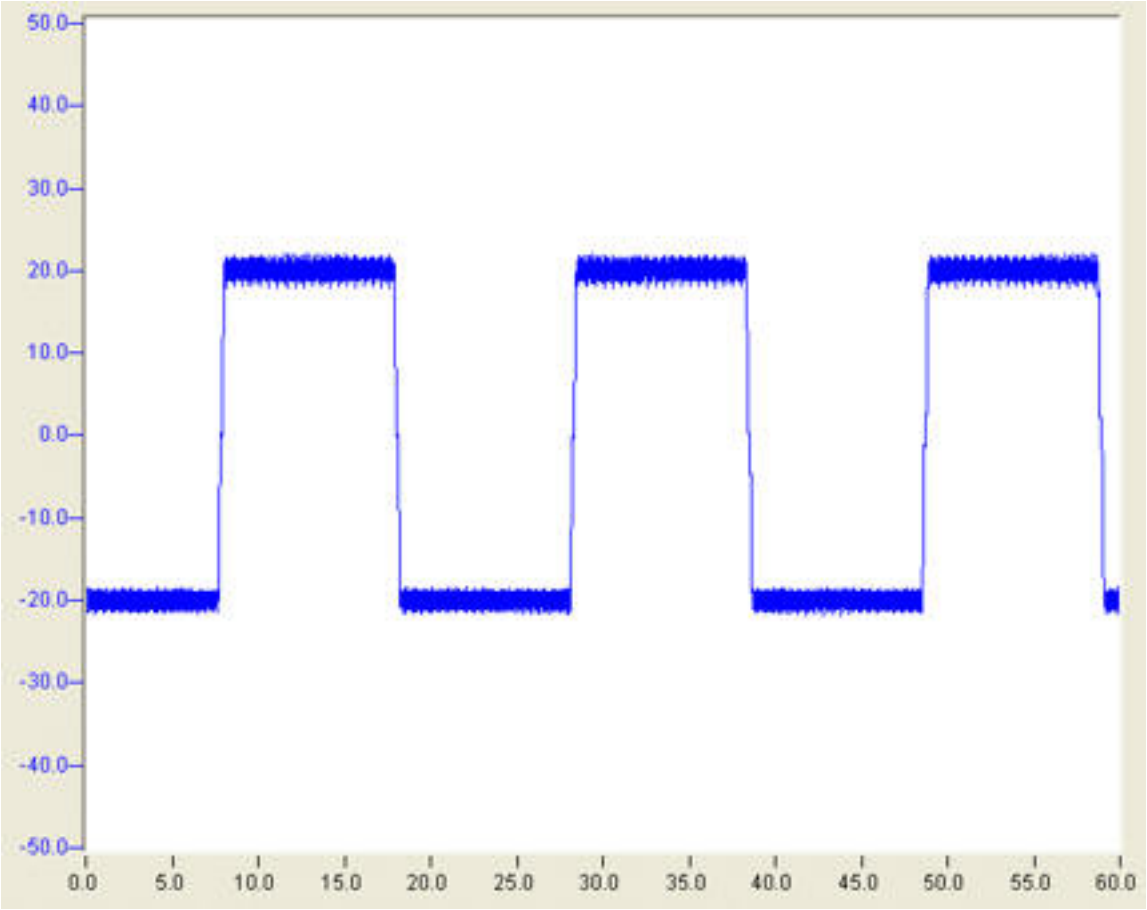


Fig. 6: Figure 2: Actual velocity of an axis operated with corrected axis information.

Description

The beat effects result from information being missed or read twice. These occurrences can be detected and countermeasures taken based on the cycle index, i.e. a counter that is incremented whenever network variables are sent. The function block is designed such that

a) the axis structures received as network variables and the corresponding cycle index are required as input for the block;

b) the axis information (set position and velocity) is corrected internally; and

c) an NC axis can be operated with the corrected information.

The cycle index and the cycle time(s) are transferred internally to the function block FB_TimeSync [▶ 14]. This block determines the frequency of the beat and a correction time for extrapolating the kinematic axis information such that the beat effects are compensated. To this end the correction time is transferred to an instance of the function block FB_AxisExtrapolateValues [▶ 17], together with the received axis structure, which calculates the corrected set positions and velocities. Note that by default the set acceleration is used for correcting the position and the velocity. This feature must be deactivated in the event of non-smooth set acceleration, e.g. for encoder axes (stAxisSyncParameters.stAxisExtrapolateParameters.bUseAccForExtrapolation:=FALSE).The corrected set values are made available in the form of an axis structure and can then be used for operating the axis.

Critical Operating Notes

Please ensure that the fcuntion block FB_AxisSync is called one time only in each plc cylce!

VAR_INPUT

```
VAR_INPUT
    bEnable:              BOOL;
    iCycleIndex:          WORD;
```

```
    fTaskCycleTime:       LREAL;
    eFilterMode:          E_Sync_FilterMode;
    stAxisSyncParameters: ST_AxisSyncParameters;
END_VAR
```

bEnable : The FB is activated through a rising edge. If *bEnable*=FALSE, the input axis structure *AxisIn* is copied unmodified to output *AxisOut*.

iCycleIndex : Counter, which is incremented by 1 and transferred whenever axis structure *AxisIn*is transferred. It is used for determining the beat frequency and for identifying fluctuations.

fTaskCycleTime : Zykluszeit [s].By default it is assumed that both sender and receiver are operated with cycle time *fTaskCycleTime*. In the event of different cycle times *fTaskCycleTime* corresponds to the receiver time, *stAxisSyncParameters.stTimeSyncParameters.fDataCycleTime* corresponds to the sender time.

eFilterMode :The method [▶ 21] for calculating the corrected axis set values can be chosen irrespective of the state of the synchronization (displayed through via output *bSynced*). This does not affect automatic fallback (see ST_AxisSyncParameters [▶ 20].*eFallBackMode*), which is activated if the distance between the calculated and the original position exceeds a certain value. Automatic mode is activated by default, which means that

1. a) The selected mode (ST_AxisSyncParameters.*eStartUpMode*) is used during initialization. Time mode is used by default, i.e. a cycle time is calculated in order to prevent fluctuations.
2. b) As soon a sufficient number (*stAxisSyncParameters.stTimeSyncParameters.iNoOfPeriodsForMeanDrift* +1) of beat effects were detected, the system switches to sync mode, and the set values are extrapolated with the correction time.
3. c) If synchronization is lost for some reason, the system switches to PT1 mode. In this mode the transferred positions are filtered before being transferred to the output.

stAxisSyncParameters : Structure for detailed configuration of the FB. The default values should be adequate in most cases.

VAR_OUTPUT

```
VAR_OUTPUT
    bSynced              : BOOL;
    bError               : BOOL;
    iErrorId             : DINT := 0;
    eFilterState         : E_Sync_FilterState;
    stAxisSyncDiagnostic : ST_AxisSyncDiagnostic;
END_VAR
```

bSynced : TRUE if the receiver is synchronized with the sender, otherwise FALSE.

bError : TRUE in the event of a fault

iErrorId : Provides the error code [▶ 26]

eFilterState : Indicates what method [▶ 21] was used for determining the set values returned in AxisOut. The following options are available:

- Bypass : The values supplied in AxisIn are transferred *unmodified* to AxisOut.
- PT1 : The input values were smoothed with the aid of a PT1 filter.
- Sync : The output values were determined from the input variables via the correction time.
- Time : The time correction is calculated such that any beat effects and fluctuations are compensated.

Further information can be found in the documentation for function block FB_TimeSync [▶ 14].

stAxisSyncDiagnostic : Diagnostic values [▶ 20]

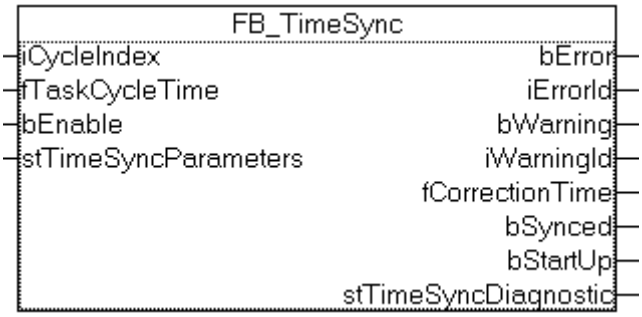
VAR_IN_OUT

```
VAR_IN_OUT
   AxisIn  : NCTOPLC_AXLESTRUCT;
   AxisOut : NCTOPLC_AXLESTRUCT;
END_VAR
```

AxisIn : Input axis structure of the sender.

AxisOut : Output axis structure with corrected axis set values.

3.2 FB_TimeSync



This function block is used for system synchronization. The phase difference of the systems is determined based on the cycle index of received network variables, and a correction time is calculated.

Description

For synchronization of distributed systems the information is sent cyclically by one system and received by the others. Phase differences between the cycle times of the sender and the respective receiver lead to beat effects (see introduction [▶ 7]) through information being missed or read twice. In the following section only the case with nominally identical cycle times on the sender and receiver side is considered. For different cycle times (even-numbered ratios are supported) the following description applies with different values.

Each send process is associated with a consecutive number, i.e. cycle index *m* of the network variables. The following cases may occur for the difference Δm_i between the cycle index received during the current cycle (i) and the previous cycle (i-1) ($\Delta m_i = m_i - m_{i-1}$):

Δm=0: The same data are read during the current and the previous cycle.

Δm=1: The sent data are received continuously.

Δm>=2: One or several pieces of information were not received.

Figure 1 schematically shows the case for Δm=0, i.e. duplicate reading of information, at the position indicated. Figure 2 shows case Δm=2, with sent information being skipped and not analysed.

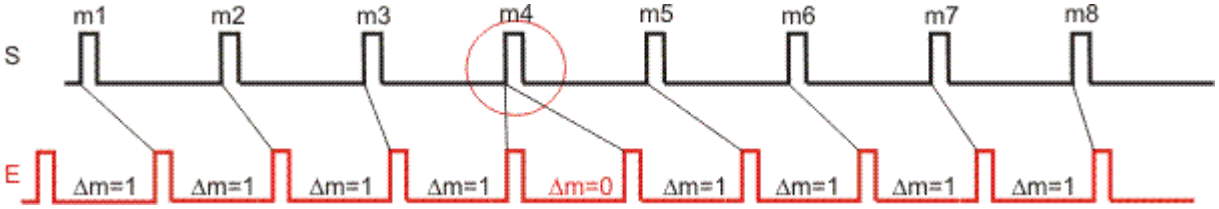


Fig. 7: Figure 1: Beat effect if the cycle time of the sender (S) is greater than that of the receiver (E). Corresponding sender and transmitter cycles are linked through lines.

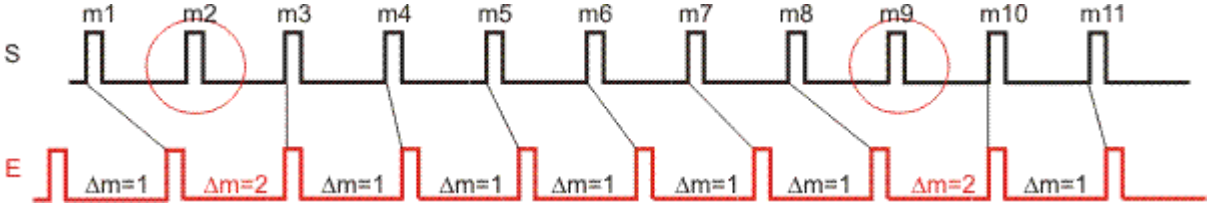


Fig. 8: Figure 2: Beat effect if the cycle time of the sender (S) is smaller than that of the receiver (E).

Identification of beat effects

In the ideal case of times only differing through drift, the occurrence of Δm≠1 would indicate a step change. Due to the jitter of the cycle times around their nominal value, these cases predominantly occur around the beat, causing the step change to be broadened into a transition zone. The width of this zone depends on the magnitude of the clock jitter and on the absolute phase difference. Due to the width of the transition, it has to be identified through a condition that is different from Δm≠1. At discontinuities, deviations from Δm=1 cause the difference between sender and received cycles to increase. A difference that is constant over several (*stTimeSyncParameters.iEndOfTransitionLimit*) cycles is used for identifying the occurrence of a beat. This means that a step change can only be identified a certain number of cycles after its occurrence, which is significant for determining the corrections.

The phase difference between the computers is determined through the number of clock cycles of the receiver between two beats. This value can be used for correcting the beat effects.

Corrections

Figure 3 shows the cycle index of the sent information as a function of the received cycle index for case Δm=2 at the discontinuity. The beat effects cause the gradient of the curve to deviate from the ideal value of 1. In order to avoid step changes, a corrected cycle index can be calculated (dotted line) such that the beat effects are compensated during the time between the step changes. The result is a corrected cycle index on the receiver side that deviates slightly from the transmitted value, but is equidistant and shows no step changes.

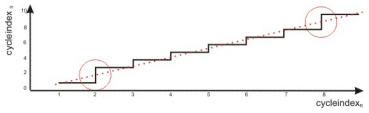


Fig. 9: Figure 3: Received (continuous line) and corrected (dashed line) cycle index. The discontinuities with Δm=2 are indicated by circles.

Since discontinuities can only be detected after the event as described above, the correction calculated with the current drift can also only be applied with a delay. In order to avoid discontinuities of the corrected cycle index even in the event of strong differences in clock times, the drift used for the correction is gradually adjusted between the old valued and the new value once a discontinuity has been identified.

Another option for limiting the effects of strong changes in phase differences is non-uniform correction of beat effects, which means a stronger correction is applied after a discontinuity than just before the expected occurrence of the next discontinuity. If the time between two step changes becomes shorter, the correction is largely completed, and only minor additional corrections will be required. Further information can be found in the description of the configuration parameters [▶ 22].

Correction time

The correction time can be determined from the difference between the corrected and the received cycle index through multiplication with the cycle time. The correction time specifies the time shift required for the information currently being read, in order to enable equidistant interpretation without step changes. If the temporal variation of the value to be corrected is known, it can be extrapolated accordingly. For the cycle index the interrelationship is simple (+1 per cycle time), for extrapolation of axis information such as position and velocity, the velocity and acceleration have to be known (see documentation for function blocks FB_AxisSync [▶ 10] and FB_AxisExtrapolateValues [▶ 17], and introduction [▶ 7]).

Initialisation phase

For calculating the corrections, at least two beat effects must have occurred, in order to be able to calculate a drift. The so-called time mode is available for compensating fluctuations and step changes even before these two beat effects have occurred. In this mode (outputs *bStartUp*=TRUE and *bSynced*=FALSE) the corrected cycle index is incremented by 1 during each cycle, thus compensating step changes in the received cycle index. After a discontinuity has been identified (the time of occurrence of which is not yet foreseeable), the difference is compensated during the course of several cycles (*stTimeSyncParameters.iTimeModeBlendingCycles*). Once the required number of beats

(*stTimeSyncParameters.iNoOfPeriodsForMeanDrift)* has been identified, the correction time is calculated according to the method described above (outputs bStartUp=FALSE and bSynced=TRUE), and the system changes from time mode to sync mode.

VAR_INPUT

```
VAR_INPUT
    iCycleIndex          : WORD;
    fTaskCycleTime       : LREAL;
    bEnable              : BOOL;
    stTimeSyncParameters : ST_TimeSyncParameters;
END_VAR
```

iCycleIndex : Cycle index of the currently received data

fTaskCycleTime : Nominal receiver cycle time in [s]. If the cycle time of the sender differs from this value, it has to be configured via parameter stTimeSyncParameters.fDataCycleTime. Otherwise it is assumed that both cycle times are identical. The cycle times must be even-numbered multiples of each other, i.e. tSender=2ms, tReceiver=4ms or tReceiver=1ms, for example.

bEnable : FALSE deactivates the function block, a rising edge reinitialises and starts it

stTimeSyncParameters :Structure [▶ 22] contains the Configuration paramters

VAR_OUTPUT

```
VAR_OUTPUT
    bError              : BOOL;
    iErrorId            : DINT;
    bWarning            : BOOL;
    iWarningId          : DINT;
    fCorrectionTime     : LREAL;
    bSynced             : BOOL;
    bStartUp            : BOOL;
    stTimeSyncDiagnostic: ST_TimeSyncDiagnostic;
END_VAR
```

bError :TRUE in the event of a fault
iErrorId: Returns error code (E_Sync_ErrorCodes [▶ 26]) in the event of a fault

bWarning : TRUE in the event of a warning, i.e. operation continues under sub-optimum conditions.

iWarningId : Returns the corresponding warning code (E_Sync_WarningCode [▶ 27])

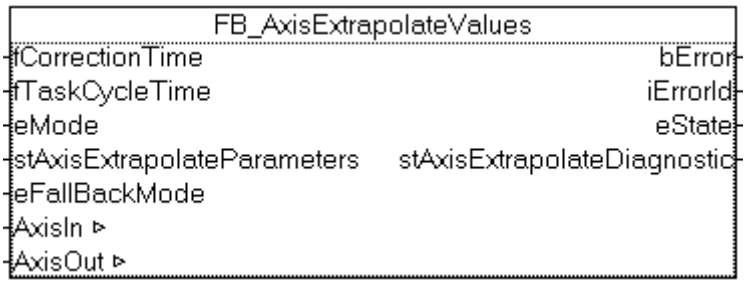
fCorrectionTime : Correction time, i.e. the time used for extrapolating the received information [s]

bSynced : TRUE if sender and receiver are synchronised. The system was able to determine a correction time in sync mode, i.e. the initialisation phase is completed, and the correction time is within the permitted range (i.e. after identification of *stTimeSyncParameters.iNoOfPeriodsForMeanDrift* beats, and the absolute value of the correction time subsequently falling stTimeSyncParameters.fThresholdForSync*TaskCycleTime).

bStartUp : TRUE if the function block is in the initialization phase. During this phase the correction time is calculated such that it can be used for extrapolating the data, although _only_ fluctuations (Δm≠1) are compensated. After identification of a beat during the initialization phase, a correction time of 0 is applied linearly within*stTimeSyncParameters.iExtendedStartUpBlendingCycles* cycles.

stTimeSyncDiagnostic: Structure [▶ 24] with advanced diagnostics outputs

3.3 FB_AxisExtrapolateValues



This function block is used for extrapolating kinematic axis parameters. The set position (and optionally the set velocity) are extrapolated based on the set velocity (and set acceleration) around the specified correction time.

NOTE

By default, the parameter for using the set acceleration for extrapolating the velocity and position is _activated_. Accordingly, adequately smooth, i.e. noise-free and spike-free acceleration values (e.g. analytical values from PTP axes) have to be present in the AxisIn structure. Encoder axes do not meet these acceleration requirements through sampling and dual position differentiation. Application of the acceleration has to be deactivated by setting parameter stAxisExtrapolateParameters.bUseAccForExtrapolation=FALSE.

Description

Beat effects described in the introduction [▶ 7] lead to loss or duplicate reading of sent information. If this information contains set values for axes, the result are step changes in the kinematic variables (set position and set velocity). In axes that are operated directly with these variables, this will cause periodic disturbance. The situation can be rectified by correcting the set values using higher derivative terms (set velocity or set acceleration) and a correction time. The latter is configured (see FB_TimeSync [▶ 14]) to ensure that beat effects and set value fluctuations can be compensated through the extrapolation.

Figure 1 shows the received (continuous line) and the corrected (dashed line) position, with the axis being driven with the position interface of the uncorrected values. This results in increasing fluctuations of the actual velocity (dotted line). Figure 2 shows an overview of the situation.

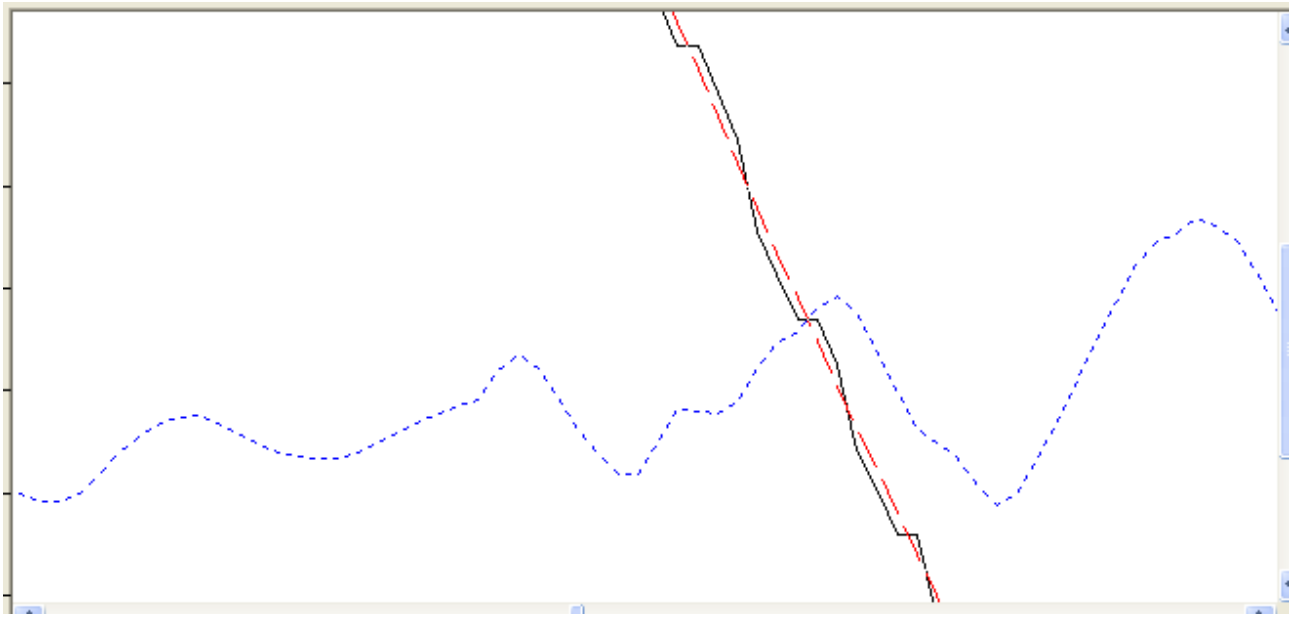


Fig. 10: Figure 1: Uncorrected (black, continuous) and corrected (red, dashed) set position and actual velocity (blue, dotted) of the axis driven with the _uncorrected_ position in the position interface.

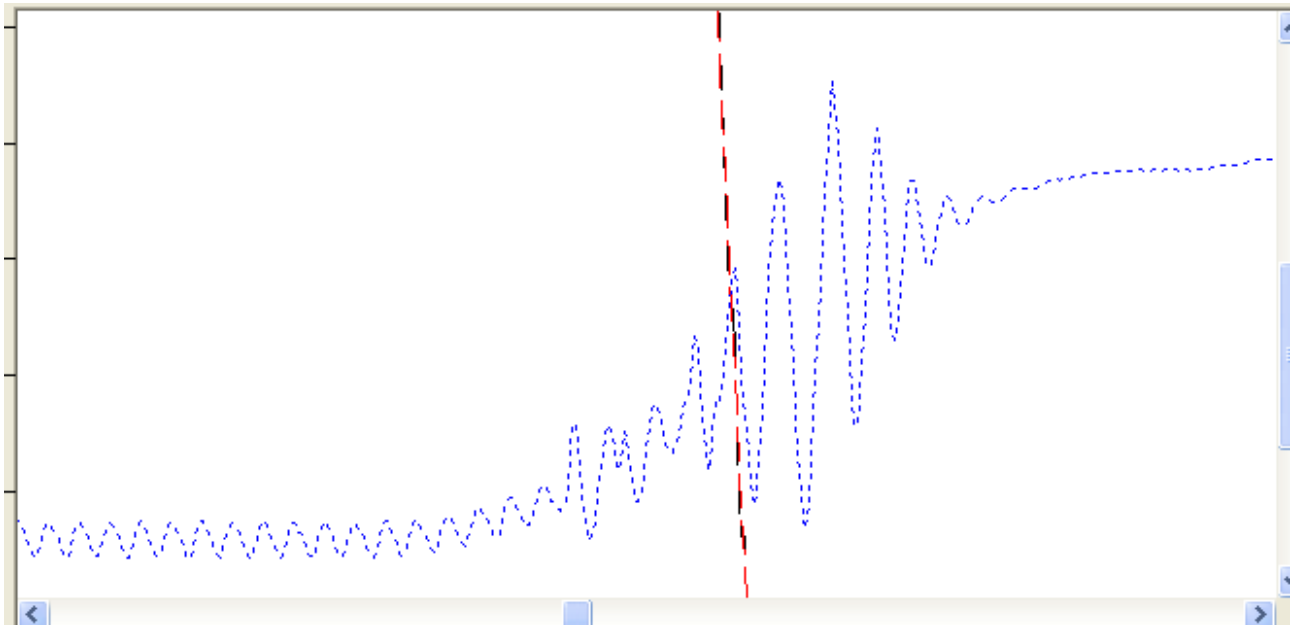


Fig. 11: Figure 2: Overview of the situation, shown enlarged in Figure 1.

Extrapolation of the set values

Two techniques that differ in terms of detail are available for extrapolation of the set values. The first technique only uses the set velocity for extrapolating the set position:

```
AxisOut.fPosSoll:=AxisIn.fPosSoll + AxisIn.fVeloSoll * fTimeCorrection;
AxisOut.fVeloSoll:=AxisIn.fVeloSoll;
```

Furthermore, the set acceleration may also be used for extrapolating the set position and set velocity:

```
AxisOut.fPosSoll:=AxisIn.fPosSoll + AxisIn.fVeloSoll*fTimeCorrection+ 0.5 * AxisIn.fAccSoll
*fTimeCorrection^2;
AxisOut.fVeloSoll:=AxisIn.fVeloSoll + AxisIn.fAccSoll *fTimeCorrection;
```

Requirements:

Depending on the main mode in which the axis is to be operated, certain requirements have to be met, and the associated mode has to be activated. The corresponding combinations are shown in Table 1.

Main axis operating mode	Parameters required in AxisIn	Parameters	Comment
Position interface	Set position Set velocity (set acceleration)	stAxisExtrapolateParameters .bUseAccForExtrapolation=FALSE (=TRUE)	Acceleration can optionally be added as a parameter. (The acceleration requirements described above must be met.)
Velocity interface	Set position Set velocity Set acceleration	stAxisExtrapolateParameters .bUseAccForExtrapolation=TRUE	Analytical acceleration behaviour required, i.e. no encoder axes, for example

Operating modes:

In addition to set value extrapolation, further operating modes are available. Two of these (bypass mode and PT1 mode) are mainly used in cases where no correction time is available for the extrapolation. If the function block FB_TimeSync [▶ 14] is used this is the case during the initialization phase or if synchronization fails for other reasons. This block can also be used with an optional time mode that enables a correction time for avoiding fluctuations to be determined during initialization without knowledge of the beat frequency.

The following operating modes are available:

Mode	Description
E_Sync_ExtrapolateMode_Bypass	All received set values are directly copied to the output, i.e. without modification of the kinematic variables and so without reduction of beat effects.
E_Sync_ExtrapolateMode_PT1	In PT1 filter mode the input position is transferred to the output via a configurable PT1 filter (see stAxisExtrapolateParameters [▶ 25]). Beat effects can thus be reduced.
E_Sync_ExtrapolateMode_Sync	Sync mode is the standard block mode. The kinematic set values are extrapolated according to the correction time as described above.
E_Sync_ExtrapolateMode_Time	In the event of data being read twice/missed, the correction time is calculated such that the influence on the set values are minimised. In contrast to sync mode, the set values are not corrected across the whole range between the beat effects.

Safety procedures

This distance can be limited, in order to avoid the difference between the original-set values and the calculated variables becoming too large in the event of the correction time being calculated incorrectly. The maximum difference is defined through the position or velocity change that can be achieved with the current set velocity or set acceleration in stAxisExtrapolateParameters.fMaxPositionDiffFactor cycles. If the difference exceeds this value, the configurable fallback mode is activated. Bypass of PT1 mode is available for this purpose (see description above).

VAR_INPUT

```
VAR_INPUT
    fCorrectionTime             : LREAL;
    fTaskCycleTime              : LREAL;
    eMode                       : E_Sync_ExtrapolateMode;
    eFallBackMode               : E_Sync_FallBackMode;
    stAxisExtrapolateParameters : ST_AxisExtrapolateParameters;
 END_VAR
```

fCorrectionTime :Extrapolation correction time [s]

fTaskCycleTime :Receiver cycle time [s].

eMode : Pre-selected mode [▶ 25], can automatically be changed through safety procedures (fallback). The actual mode used is displayed as *eState* output.

eFallBackMode : Mode [▶ 21] if the deviation of the calculated set values from the original set values exceeds the value described by the parameter *stAxisExtrapolateParameters.fMaxPositionDiffFactor*.

stAxisExtrapolateParameters : S [▶ 25]tructure with configuration parameters

VAR_OUTPUT

```
VAR_OUTPUT
    bError                    : BOOL;
    iErrorId                  : DINT;
    eState                    : E_Sync_ExtrapolateState;
    stAxisExtrapolateDiagnostic: ST_AxisExtrapolateDiagnostic;
END_VAR
```

bError : TRUE in the event of a fault

iErrorId: Error code (E_Sync_ErrorCodes [▶ 26])

eState : Mode used (E_Sync_ExtrapolateState [▶ 21])

stAxisExtrapolateDiagnostic: Structure [▶ 24] with extended diagnostic outputs

VAR_IN_OUT

```
VAR_IN_OUT
    AxisIn  : NCTOPLC_AXLESTRUCT;
    AxisOut : NCTOPLC_AXLESTRUCT;
END_VAR
```

AxisIn : Received original axis structure

AxisOut : Axis structure for avoiding beat effects with modified set values

3.4 ST_AxisSyncParameters

Configuration structure for FB_AxisSync [▶ 10].

```
TYPE ST_AxisSyncParameters :
  STRUCT
    eStartUpMode               : E_Sync_StartUpMode := E_Sync_StartUpMode_Time;
    eFallBackMode              : E_Sync_FallBackMode := E_Sync_FallBackMode_PT1;
    stAxisExtrapolateParameters : ST_AxisExtrapolateParameters;
    stTimeSyncParameters    : ST_TimeSyncParameters;
  END_STRUCT
END_TYPE
```

eStartUpMode: Specifies which mode [▶ 22] is used for correction of the set values during the initialization phase, before the phase difference was determined.

eFallBackMode: Mode [▶ 21] when the difference between the calculated and original set position is exceeded (see stExtrapolateParameters.fMaxPositionDiff)

stExtrapolateParameters : Configuration structure [▶ 25] of the FB_AxisExtrapolateValues [▶ 17] block.

stTimeSyncParameters : Configuration structure [▶ 22] of the FB_TimeSync [▶ 14] block.

3.5 ST_AxisSyncDiagnostic

Diagnostic structure for FB_AxisSync [▶ 10].

```
TYPE ST_AxisSyncDiagnostic :
  STRUCT
    fPositionDiff               : LREAL;
    fVeloDiff                   : LREAL;
    fCorrectionTime             : LREAL;
    bTimeMode                   : BOOL;
    stTimeSyncDiagnostic        : ST_TimeSyncDiagnostic;
    stAxisExtrapolateDiagnostic : ST_AxisExtrapolateDiagnostic;
  END_STRUCT
END_TYPE
```

fPositionDiff: Current difference between original and calculated set position: *AxisIn.fPosSoll* - *AxisOut.fPosSoll*

fVeloDiff: Current difference between original and calculated set velocity: *AxisIn.fVeloSoll* - *AxisOut.fVeloSoll*

fCorrectionTime: If *bSynced*=TRUE, *fCorrectionTime* contains the correction time for the synchronisation

If *bSynced*=FALSE and *bTimeMode*=TRUE, the *fCorrectionTime* value is calculated for avoiding fluctuations

bTimeMode:TRUE if the time mode of the FB_TimeSync block is active

stTimeSyncDiagnostic: Structure [▶ 24] with diagnostic outputs of the FB_TimeSync [▶ 14] block

stAxisExtrapolateDiagnostic: Structure [▶ 24] with diagnostic outputs of the FB_AxisExtrapolateValues [▶ 17] block

3.6 E_Sync_FallBackMode

Specifies which mode is used for modifying the set values if synchronisation is lost

```
TYPE E_Sync_FallBackMode :
(
    E_Sync_FallBackMode_Bypass  := 1
    E_Sync_FallBackMode_PT1     := 2
);
END_TYPE
```

E_Sync_FallBackMode_Bypass: The set values supplied by the sender, including all beat effects, are not modified.

E_Sync_FallBackMode_PT1: A PT1 filter passes the set positions on to the receiver axis.

3.7 E_Sync_FilterMode

Specifies which mode is used for modifying the set values.

```
TYPE E_Sync_FilterMode :
(
    E_Sync_FilterMode_Auto   := 0,
    E_Sync_FilterMode_Bypass := 1,
    E_Sync_FilterMode_PT1    := 2,
    E_Sync_FilterMode_Time   := 4
);
END_TYPE
```

E_Sync_FilterMode_Auto: Automatic mode: During initialisation the selected StartUpMode is used, in synchronised state the extrapolation with correction time is used, and in the event of synchronisation loss the specified FallBack mode is used.

E_Sync_FilterMode_Bypass: The set values supplied by the sender, including all beat effects, are not modified.

E_Sync_FilterMode_PT1: A PT1 filter passes the set positions on to the receiver axis.

E_Sync_FilterMode_Time: The time correction is calculated such that any beat effects and fluctuations are compensated.

3.8 E_Sync_FilterState

Specifies which mode is currently used for modifying the set values.

```
TYPE E_Sync_FilterState :
(
    E_Sync_FilterState_Bypass := 1,
    E_Sync_FilterState_PT1    := 2,
    E_Sync_FilterState_Sync   := 3,
    E_Sync_FilterState_Time   := 4
);
END_TYPE
```

E_Sync_FilterState_Bypass: The set values supplied by the sender, including all beat effects, are not modified.

E_Sync_FilterState_PT1: A PT1 filter passes the set positions on to the receiver axis.

E_Sync_FilterState_Sync: The set values are extrapolated with the correction time.

E_Sync_FilterState_Time: The time correction is calculated such that any beat effects and fluctuations are compensated directly after their occurrence. In contrast to sync mode, there is *no* preventive compensation of the calculated beat effects.

3.9 E_Sync_StartUpMode

Possible operating modes for extrapolating the set values on startup, i.e. in the absence of a calculated beat frequency.

```
TYPE E_Sync_StartUpMode :
(
    E_Sync_StartUpMode_Bypass  := 1,
    E_Sync_StartUpMode_PT1     := 2,
    E_Sync_StartUpMode_Time    := 4
);
END_TYPE
```

E_Sync_StartUpMode_Bypass: The set values supplied by the sender, including all beat effects, are not modified.

E_Sync_StartUpMode_PT1: A PT1 filter passes the set positions on to the receiver axis.

E_Sync_StartUpMode_Time: The time correction is calculated such that any beat effects and fluctuations are compensated.
(Details can be found in section describing the FB_TimeSync [▶ 14] startup mode)

3.10 ST_TimeSyncParameters

Configuration structure for the FB_TimeSync [▶ 14] block

```
TYPE ST_TimeSyncParameters :
  STRUCT
    fDataCycleTime                : LREAL := 0.0;
    fThresholdForSync             : LREAL := 0.05;
    fMaxCycleIndexDifference      : LREAL := 7;
    iAgeOfDataLimit               : INT   := 7;
    iNoOfPeriodsForMeanDrift      : INT   := 1;
    fDelayTimeOffset              : LREAL := 0.0;
    fSlope1Numerator              : LREAL := 0.95;
    fSlope1Denominator            : LREAL := 0.5;
    iEndOfTransitionLimit         : INT   := 90;
    iNewDriftBendingCycles        : INT   := 90;
    iExtendedStartUpBlendingCycles: INT   := 90;
    bForceTimeMode                : BOOL  := FALSE;
    bAutomaticReInit              : BOOL  := FALSE;
    nSysCmd                       : DWORD := 1;
  END_STRUCT
END_TYPE
```

fDataCycleTime: Sender cycle time in [s]. If fDataCycleTime=0 it is assumed that sender and receiver operate with same cycle time. The cycle times must be even-numbered multiples of each other, i.e. t_{Sender}=2ms, $t_{Receiver}$=4ms or $t_{Receiver}$=1ms, for example. If*fDataCycleTime*<FB_TimeSync.fTaskCycleTime, the parameter *fMaxCycleIndexDifference* has to be adjusted accordingly (either by deactivating (=0) or be setting to a minimum value of *FB_TimeSync.fTaskCycleTime/fDataCycleTime*+2). The same applies to *FB_TimeSync.iAgeOfDataLimit*.

fThresholdForSync : Threshold value of the correction time from which synchronisation is activated. In order to avoid excessive differences between the extrapolated and the original values, bSynced=TRUE is only set if the correction time is smaller than the part of the cycle time specified by the parameter, i.e. if:

ABS(CorrectionTime)<(fThresholdForSync*TaskCycleTime)

fMaxCycleIndexDifference : Specifies the maximum value for the difference between the corrected and the original cycle index. If the actual value exceeds this value, error *E_Sync_TimeSync_Error_MaxCycleIndexDiffExceeded* is issued. fMaxCycleIndexDifference=0 deactivates this check. If the sender and receiver operate with different cycle times, *fMaxCycleIndexDifference* has to be adjusted as described under *fDataCycleTime*.

iAgeOfDataLimit: Maximum "age" of the data used in task cycles. For continuous reading, i.e. a new data set is received from the sender during each cycle, *ST_TimeSyncDiagnostic.iConsecutiveEqualDataCounter* =0. During each cycle in which the data remain constant the value is incremented by 1. In order to be able to respond to beat effects, the *minimum value for is 1iAgeOfDataLimit*. The higher the value, the more data lost during transfer, for example, are accepted. If the limits is exceeded, error

E_Sync_TimeSync_Error_MaxDataAgeExceeded and *bSynced*=FALSE is displayed. iAgeOfDataLimit=0 deactivates the check. If the sender and receiver operate with different cycle times, *iAgeOfDataLimit* has to be adjusted as described under *fDataCycleTime*.

iNoOfPeriodsForMeanDrift : The drift between sender and receiver used for regulation is determined as the mean value between the last *iNoOfPeriodsForMeanDrift* values.

fDelayTimeOffset : Offset in [s] that is added to *fCorrectionTime*. Positive values refer to an additional extrapolation into the future, e.g. for compensating runtime differences.

NOTE

Any changes of this value have a _direct_ influence on the correction time *fCorrectionTime* and therefore on the extrapolated values (see FB_AxisExtrapolateValues [▶ 17] or FB_AxisSync [▶ 10], for example). Accordingly, step changes in the kinematic variables used for control purposes caused by rapid changes in *fDelayTimeOffset* must be avoided. The greater *fDelayTimeOffset*, the greater the deviation between the values originally transferred and the extrapolated values. The parameter fMaxPositionDiff in the ST_AxisExtrapolateParameters, for example, must be adjusted accordingly.

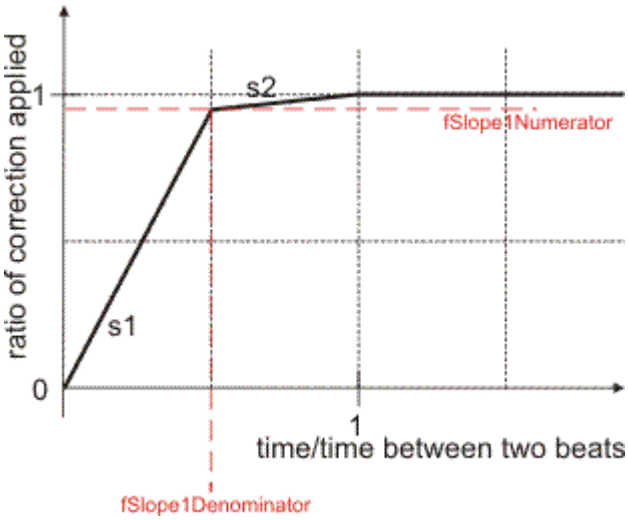
The following two parameters are particularly relevant for strong fluctuations in phase difference:

fSlope1Numerator : Part of the correction that is executed within the time *fSlope1denominator***fTaskCycleTime*. In order to compensate the beat effects, the received cycle indices have to be corrected by +/- 1 between two step changes. If a discontinuity occurs significantly earlier than expected, the correction will not yet be completed, leading to a large difference to be corrected during the next cycle. The parameters *fSlope1Numerator* and *fSlope1Denominator* can be used to implement a disproportionately large part of the correction directly after a discontinuity. *fSlope1Numerator* specifies which part of the correction is implemented during the first period after a discontinuity, *fSlope1Denominator* specifies the length of the period as a fraction of the expected time between the beat effects. The scaling factor for the corrections during the remaining interval until the expected occurrence of the beat automatically results from the requirement that 100% of the correction must have been applied at the expected time of occurrence. The remaining part (1-*fSlope1Numerator*) of the correction is applied at time (1-*fSlope1Denominator*)**fTaskCycleTime*. Once the expected time of occurrence of the beat has been reached, the correction remains constant.

The following slope values are used by default:

s1 = 0.95/0.5 =*fSlope1Numerator*/*fSlope1Denominator*

s2 = 0.05/0.5= (1-*fSlope1Numerator*)/ (1-*fSlope1Denominator*)



fSlope1Denominator : Specifies the time in fractions of the cycle time *fTaskCycleTime*, during which the part of the correction specified through *fSlope1numerator* is applied.

iEndOfTransitionLimit : Number of task cycles during which the difference between the sender and receiver cycle counter must remain constant, in order to identify a beat. Large values are required particularly for small beat frequencies. In this case, the low relative frequency of the two systems leads to a long transition phase. Cycle time jitter causes phases of constant counter difference, the length of which increases with decreasing system drift.

iNewDriftBendingCycles : Number of task cycles for linear transition when a new drift is determined. The stronger the deviation of the new drift from the previous value, the greater the change in correction time. In order to reduce the effects of this change, linear interpolation within *iNewDriftBendingCycles* cycles between the correction times determined with the old drift and the new drift is used. After *iNewDriftBendingCycles* only the current drift is used.

iTimeModeBlendingCycles : In TimeMode, the correction time is linearly reduced to 0 within *iTimeModeBlendingCycles* task cycles when a beat effect is identified. TimeMode is automatically active, as long as *synchronisation could not be achieved (bSynced*=FALSE). In this case, the function blocks (FB_AxisSync [▶ 10] or FB_AxisExtrapolateValues [▶ 17]) used evaluating the correction time should be used as in sync mode or discarded.

bForceTimeMode: Time mode is used continuously, i.e. the system does _not_ change to sync mode when the synchronisation conditions have been reached. TimeMode is automatically active, as long as *synchronisation could not be achieved (bSynced*=FALSE). In this case, the function blocks (FB_AxisSync [▶ 10] or FB_AxisExtrapolateValues [▶ 17]) used evaluating the correction time should be used as in sync mode or discarded.

bAutomaticReInit: Automatic reinitialisation after a fault. Faults are deleted and the startup phase initiated after a cycle. *Activate with caution. Troubleshooting is made more difficult if the cause of the fault was not eliminated.*

nSysCmd : reserved

3.11 ST_TimeSyncDiagnostic

Extended output structure for FB_TimeSync [▶ 14]

```
TYPE ST_TimeSyncDiagnostic :
  STRUCT
    iCycleIndexDiff              : DINT;
    iTotalEqualDataCounter       : UINT;
    iConsecutiveEqualDataCounter : UINT;
    fDrift                       : LREAL;
  END_STRUCT
END_TYPE
```

iCycleIndexDiff : Difference between the current *iCycleIndex* and the value received during the previous task cycle. For continuous sending/receiving the value is 1. 0 if a date is read twice, 2 if a sent date was not received.

iTotalEqualDataCounter : Total number of cycles during which no new data were received since the FB was activated.

iConsecutiveEqualDataCounter : Current number of successive cycles during which no new data were received, i.e. during which *fCycleIndex*=0 applied successively.

fDrift : Current sender and the receiver phase difference in [ppm=parts per million]. A negative sign means that the cycle time of the sender is greater than that of the receiver.

fDrift := 1E6/(number of cycles between two step changes)

3.12 ST_AxisExtrapolateDiagnostic

extended output structure for FB_AxisExtrapolateValues [▶ 17].

```
TYPE ST_AxisExtrapolateDiagnostic :
  STRUCT
    fPositionDiff : LREAL := 0;
```

```
    fVeloDiff     : LREAL;
  END_STRUCT
END_TYPE
```

fPositionDiff: Current difference between original and calculated set position: AxisIn.fPosSoll - AxisOut.fPosSoll

fVeloDiff: Current difference between original and calculated set velocity: *AxisIn.fVeloSoll* - *AxisOut.fVeloSoll*

3.13 ST_AxisExtrapolateParameters

Configuration structure for FB_AxisExtrapolateValues [▶ 17].

```
TYPE ST_AxisExtrapolateParameters :
  STRUCT
    fMaxPositionDiff        : LREAL := 0;
    bUseAccForExtrapolation : BOOL  := TRUE;
    fT1factorPos            : LREAL := 3.0;
    fT1factorVelo           : LREAL := 3.0;
    fBlendingTime           : LREAL := 0.06;
    nSysCmd                 : DWORD := 1;
  END_STRUCT
END_TYPE
```

fMaxPositionDiff : Maximum permitted deviation of the received position from the calculated position in [velocity * s] (e.g. in mm if the velocity is specified in mm/s). For *fMaxPositionDiff*:=0 the position deviation is _not_ monitored. Otherwise *E_Sync_Extrapolate_Error_MaxPosDiffExceeded* is issued if the limit is exceeded, and the system switches to the configured FallBackMode (see FB_AxisExtrapolateValues [▶ 17]).

bUseAccForExtrapolation : If TRUE, the set acceleration is used for extrapolating the set velocity. This is the default and must be deactivated for encoder or NCI path axes, for example.

fT1factorPos : Time constant of the PT1 filter for the set position, specified in multiples of the fTaskCycleTime.

fT1factorVelo : Time constant of the PT1 filter for the set velocity, specified in multiples of the fTaskCycleTime.

fBlendingTime : Time during which linear interpolation is used between the calculated set positions for switching between PT1 filter to sync mode and vice versa. [s]

nSysCmd : reserved

3.14 E_Sync_ExtrapolateMode

Operating modes for calculating the set values

```
TYPE E_Sync_ExtrapolateMode :
(
    E_Sync_ExtrapolateMode_Bypass := 1,
    E_Sync_ExtrapolateMode_PT1    := 2,
    E_Sync_ExtrapolateMode_Sync   := 3,
    E_Sync_ExtrapolateMode_Time   := 4
);
END_TYPE
```

E_Sync_ExtrapolateMode_Bypass: The set values supplied by the sender, including all beat effects, are not modified.

E_Sync_ExtrapolateMode_PT1: A PT1 filter passes the set positions on to the receiver axis.

E_Sync_ExtrapolateMode_Sync: The time correction is calculated such that any beat effects are compensated over the whole cycle duration.

E_Sync_ExtrapolateMode_Time: The time correction is calculated such that any beat effects and fluctuations are compensated during and directly after their occurrence.

3.15 E_Sync_ExtrapolateState

Operating modes for calculating the set values

```
TYPE E_Sync_ExtrapolateState :
(
    E_Sync_ExtrapolateState_Bypass := 1,
    E_Sync_ExtrapolateState_PT1    := 2,
    E_Sync_ExtrapolateState_Sync   := 3,
    E_Sync_ExtrapolateState_Time   := 4
);
END_TYPE
```

E_Sync_ExtrapolateState_Bypass: The set values supplied by the sender, including all beat effects, are not modified.

E_Sync_ExtrapolateState_PT1: A PT1 filter passes the set positions on to the receiver axis.

E_Sync_ExtrapolateState_Sync: The time correction is calculated such that any beat effects are compensated over the whole cycle duration.

E_Sync_ExtrapolateState_Time: The time correction is calculated such that any beat effects and fluctuations are compensated during and directly after their occurrence.

3.16 E_Sync_ErrorCodes

Error (Hex)	Error (Dec)	Error type	Error name	Description
16#4b50	19280			-
16#4b51	19281		E_Sync_TimeSync_Error_MaxCycleIndexDiffExceeded	**"Maximum cycleindex difference exceeded "** The difference between the corrected and the calculated cycleindex is larger than allowed
16#4b52	19282		E_Sync_TimeSync_Error_MaxDataAgeExceeded	**"Maximum packet age exceeded "** The packet age is larger than (configurable) value of iMaxPackageAge SPS-cycles.
16#4b53	19283		E_Sync_TimeSync_Error_UnknownSyncLoss	"**Sync loss for unknown reason"** none of the above reasons caused sync loss
16#4b54	19284			-
16#4b55	19285			-
16#4b56	19286		E_Sync_Error_WrongParameter	**"wrong parameter used"**
16#4b57	19287			-
16#4b58	19288			-
16#4b59	19289		E_Sync_Extrapolate_Error_MaxPosDiffExceeded	**"Maximum position difference exceeded "** The difference between the actual position transmitted by the master and the position calculated for syncing is larger than the fMaxPositionDiff (configurable by input-values)

3.17 E_Sync_WarningCodes

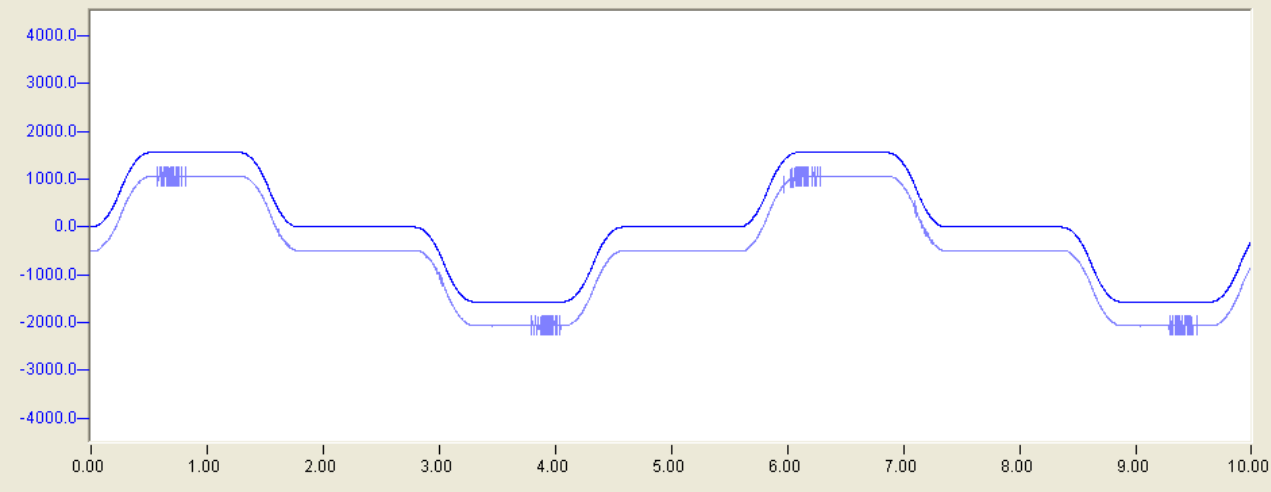
Warning (Hex)	Warning (Dec)	Error name	Description
16#1	1	E_Sync_Warning_DriftChangedSignificantly	

4 Example

4.1 Synchronisation of axes on different PCs ("distributed axes")

(Example and quick reference guide)

This quick reference guide describes the synchronisation of NC axes in receiver PCs with an axis connected to a sending PC. The "master axis" data are cyclically transferred to the receivers via network variables. Phase differences in the real-time clocks lead to beat effects, causing individual axis information to be read twice or missed on the receiver side. If the information is used directly for operating the receiver axes, these step changes in the position or velocity lead to jerky motion. Figure 1 shows (slightly offset in y) the set and actual velocity of an NC axis directly operated in the position interface with the received values. The duration and frequency of the beat effects depends on the phase difference of the systems used.



For minimising the beat effects on the receiver side the PLC function block FB_AxisSync is used below. It uses internally the function blocks FB_TimeSync and FB_AxisExtrapolateValues.

The FB_TimeSync block initially determines the time difference between the beat effects. Subsequently a correction time is calculated and used for extrapolating the received the information into the future or calculating back into the past, in order to obtain equidistant corrected information without beat effects. The step changes in the kinematic axis variables caused by the beat effect are corrected through small changes across the complete interval between two beat effects.

Two techniques that differ in terms of detail are available for extrapolation of the set values. The first technique only uses the set velocity for extrapolating the set position:

AxisOut.fPosSoll:=AxisIn.fPosSoll + AxisIn.fVeloSoll * fTimeCorrection;

AxisOut.fVeloSoll:=AxisIn.fVeloSoll;

Furthermore, the set acceleration may also be used for extrapolating the set position and set velocity:

```
AxisOut.fPosSoll:=AxisIn.fPosSoll + AxisIn.fVeloSoll*fTimeCorrection+ 0.5 * AxisIn.fAccSoll
*fTimeCorrection^2;
AxisOut.fVeloSoll:=AxisIn.fVeloSoll + AxisIn.fAccSoll *fTimeCorrection;
```

The second method, i.e. extrapolation taking onto account acceleration, is used as standard. If no adequately smooth acceleration value is available, application of the acceleration deactivated as described under FB_AxisExtrapolateValues [▶ 17].

The corrected axis information is made available as NCTOPLC_AXLESTRUCT.

Example

The example https://infosys.beckhoff.com/content/1033/tcplclibremotesync/Resources/zip/3465096075.zip consists of the following files:

sample_axissync_master.tsm

System manager setup of the sending PC. An axis is simulated ("MASTER") and its information sent as network variables via RT Ethernet.

sample_axissync_slave.tsm

System manager setup of the receiver. The information sent by the sender is received by the RT Ethernet device and linked with the variables of the PLC project. A simulated axis ("corrected master") is linked with the PLC inputs ("Nc2PlcAxis") and outputs ("Plc2NcAxis") for external set value generation. The corrected master information is available as PLC output ("MAIN.MasterNcToPlcFeedback").

sample_axissync.pro

PLC project for the receiver. FB_SyncAxis is called in MAIN. Input *iCycleIndex* is of particularly significance, since this cyclically updated index forms the basis for the whole control system. The original master axis information is transferred with the parameter AxisIn, the modified values are available as SyncAxis.AxisOut. A rudimentary state machine for controlling the NC axis is also implemented in the project.

sample_axissync.scp

ScopeView template.

More Information:
www.beckhoff.com

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com