SIEMENS

FAQ • 10/2015

Isochronous mode with PROFINET

SIMATIC S7-1500 / TIA Portal V13 SP1

https://support.industry.siemens.com/cs/ww/de/view/109480489

This entry originates from the Siemens Industry Online Support. The conditions of use specified there apply (www.siemens.com/nutzungsbedingungen).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.industry.siemens.com.

Contents

1 Introduction

1.1 Isochronous mode with PROFINET

1.1.1 Overview

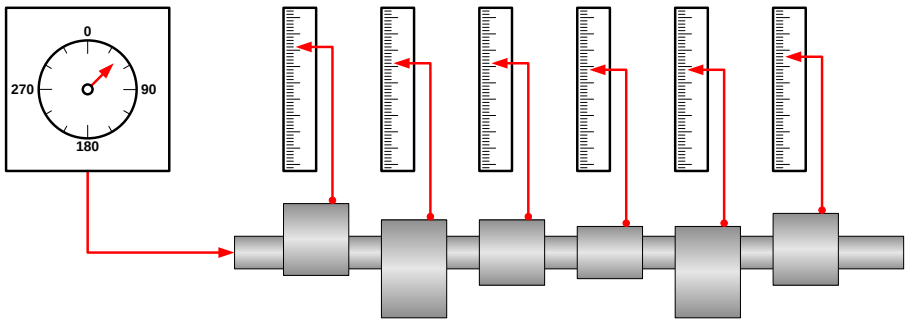
The isochronous mode is always used in the SIMATIC S7-1500 when multiple parameter values which are comparable or in a relationship to one another are to be processed in one cycle. This requirement is fulfilled, for example, in the following scenarios.

- Simultaneous acquisition of multiple measured values which are to be acquired via various input modules.
- Execution of control processes on the SIMATIC S7-1500 CPU in which the acquisition of the measured values and output of the actuator signals is via various modules.
- Coherent control of multiple drives via the drive functions integrated in the SIMATIC S7-1500.

1.1.2 Function

The coherent acquisition of multiple measured values in one program cycle is taken as an example for demonstrating the isochronous mode.

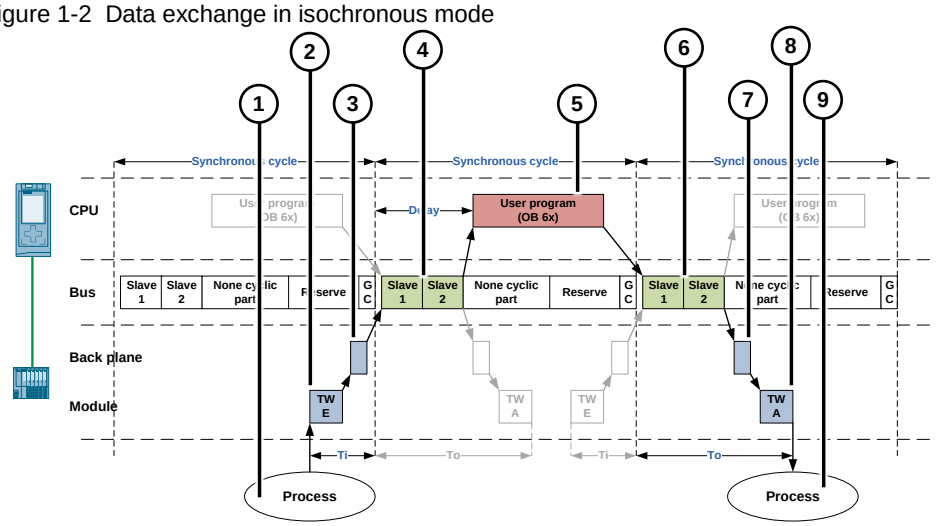
Figure 1-1 Example for demonstrating the isochronous mode function



A camshaft is to be measured precisely during one rotation. Here it is important that the single measured values of the camshaft measurement and the current angle of rotation of the camshaft are acquired at the same point in time so that these measured values can be evaluated and compared in the SIMATIC CPU.

The isochronous mode is used in the SIMATIC CPU to achieve simultaneous acquisition of the measured values. This function permits the measured values on the single modules of the distributed IOs of the SIMATIC CPU to be acquired at the same time and transferred to the SIMATIC CPU in a defined time grid. Reactions to the evaluation of the measured values in the SIMATIC CPU can then be output via this function likewise in a defined time grid via the modules of the distributed IOs.

Figure 1-2 Data exchange in isochronous mode



The complete run of data acquisition, processing and data output is given below.

Table 1-1 Data exchange in isochronous mode

No.	Description
1	The required measured value is acquired in the process
2	The acquired measured value is processed and digitized in the input module.
3	Then the digitized measured value is transferred in the distributed IOs via the backplane bus from the input module to the header module.
4	The measured value is then sent at the beginning of the bus clock to the SIMATIC CPU via the isochronous data exchange.
5	After transfer, in the SIMATIC CPU the Synchronous OB is started, in which the desired measured value is then processed.
6	The result of the measured value processing or the reaction to the measured value is then transferred at the start of the bus clock to the distributed IOs via the isochronous data exchange.
7	The output signal is then transferred again from the header module to the output module.
8	There the output signal is processed - converted into an analog value, for example.
9	The output signal is then transferred to the process.

The isochronous data exchange ensures that all the measured values are acquired in the process at the predefined time Ti before data transfer via the communication bus and at the predefined time To they are transferred to the process after transfer to the communication bus.

The times Ti and To can be calculated individually by the TIA Portal depending on the number of isochronous modules in one distributed IO station. However, the times Ti and To can be set the same for all the distributed IO stations, which is strongly recommended if you want to acquired measured values from multiple stations.

1.1.3 Special Features in the System Configuration

Currently you can only use the isochronous mode in conjunction with the SIMATIC S7-1500 via a system configuration with distributed IOs. The isochronous mode is currently not available in the central configuration of the SIMATIC S7-1500.

The modules of the distributed IOs can be connected to the SIMATIC S7-1500 CPU via PROFIBUS or PROFINET. However, you must make sure that the header modules of the distributed IOs support the isochronous mode.

1.2 Aim of this Document

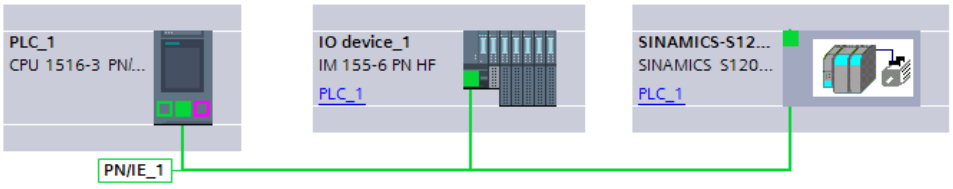
1.2.1 Motivation

This document is designed to help you get familiar with the setting options and interrelations of the isochronous mode in the TIA Portal based on sample parameterizations.

1.2.2 Configuration of the Sample Project

The sample project on which this document is based is configured as follows.

Figure 1-3 Configuration of the sample project



The sample project consists of:

- One SIMATIC S7-1500 CPU to process the measured values.
- One distributed IO to acquire the measured values.
- One drive to deliver the position to which the measured values refer.

All the components in this sample project are connected to each other via an isochronous PROFINET connection.

2 Signal Acquisition and Signal Output on the Process

2.1 Function

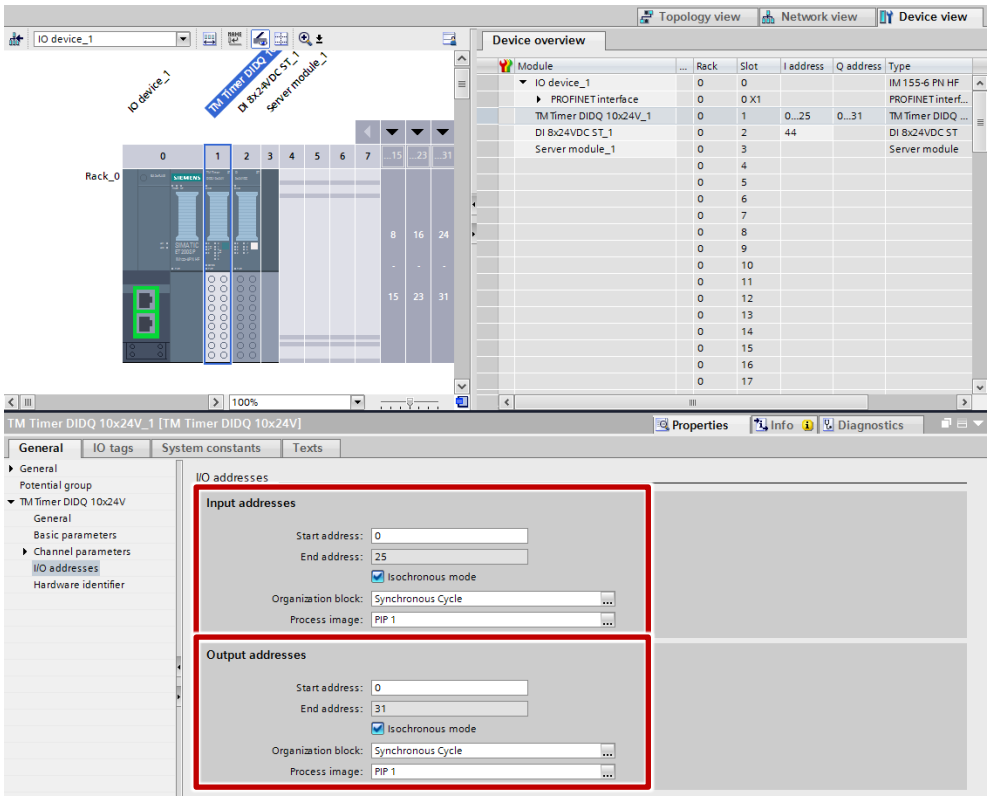
The acquisition of signals in a process and the output of the control signals to a process are done via a distributed IO.

2.2 Settings

2.2.1 IO Modules in the Distributed IO

You set the parameters for isochronous mode of the module via the Properties for setting the IO addresses of the corresponding IO module designated to acquire the signals of the process and output them again to the process.

Figure 2-1 Setting the IO addresses of the module



For the isochronous mode of this IO module the inputs and outputs of the module must be assigned to a process image partition (PIP). The data of this process image partition can then be updated synchronously with the organization blocks (OB) defined in this mask.

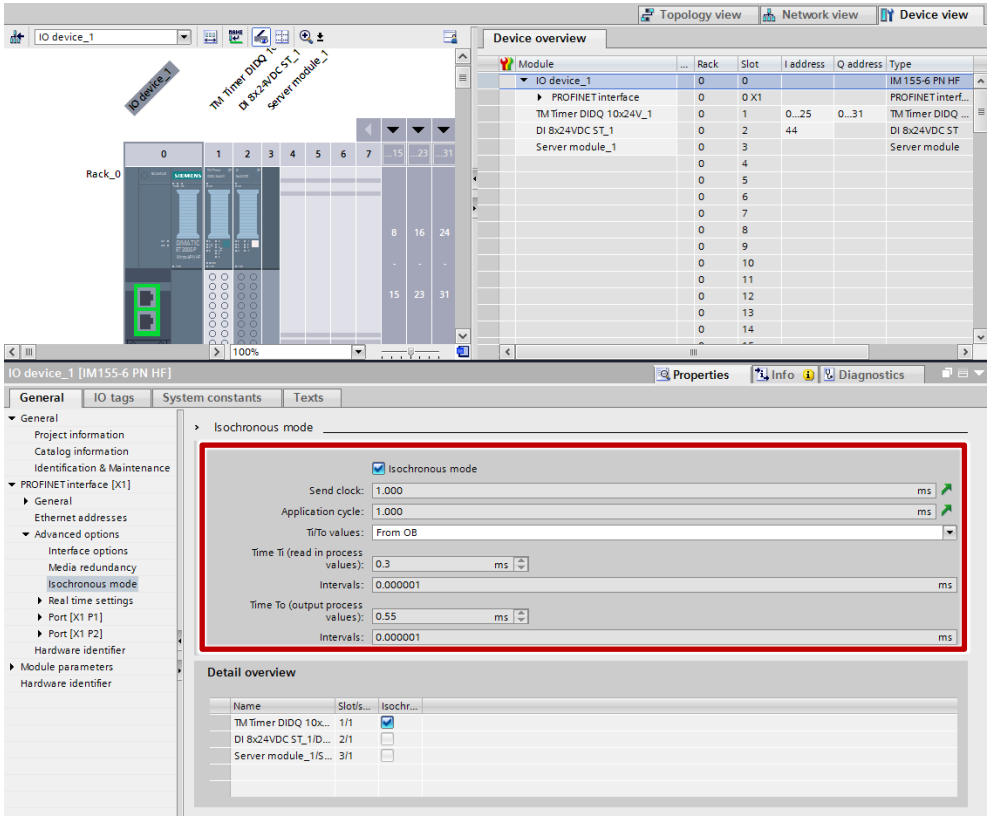
Note By assigning the inputs and outputs of the module to a process image partition (PIP) they are excluded from the automatic updating of the IO data in the cyclic organization block MAIN.

Note Only one process image partition (PIP) can be assigned to each Synchronous OB synchronous cycle.

Only a limited number of inputs and outputs can be assigned to each process image partition (PIP).

If a module of the distributed IO is set to isochronous mode, the bus interface of the distributed IO is also automatically set to isochronous mode - the PROFINET interface of the header module in the present example.

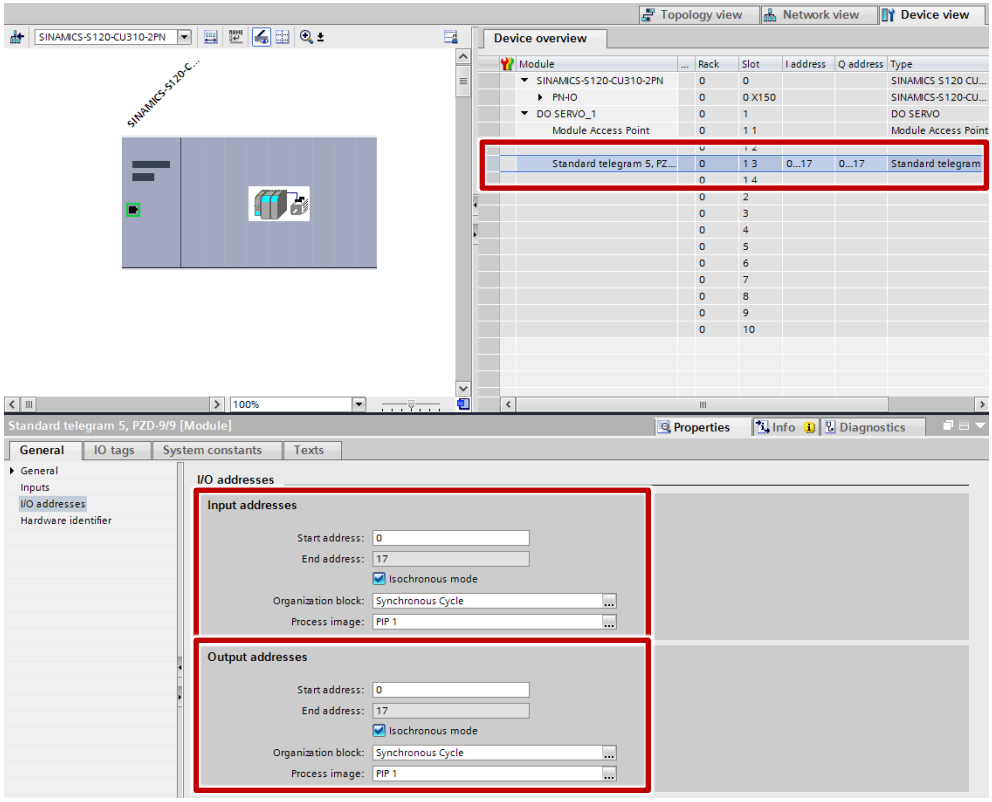
Figure 2-2 Automatic setting of clock synchronicity on the header module



2.2.2 Drive as Distributed IO

With a drive as distributed IO you can likewise set the parameters for isochronous operation via the Properties for setting the IO addresses. Here, however, you have to make sure that the IO addresses of the drive are assigned to the corresponding drive telegram. The relevant parameters are in the Properties of the drive telegram.

Figure 2-3 Setting the IO addresses of the drive telegram



In addition you must also enable clock synchronicity on the bus interface of the drive, in our example the PROFINET interface.

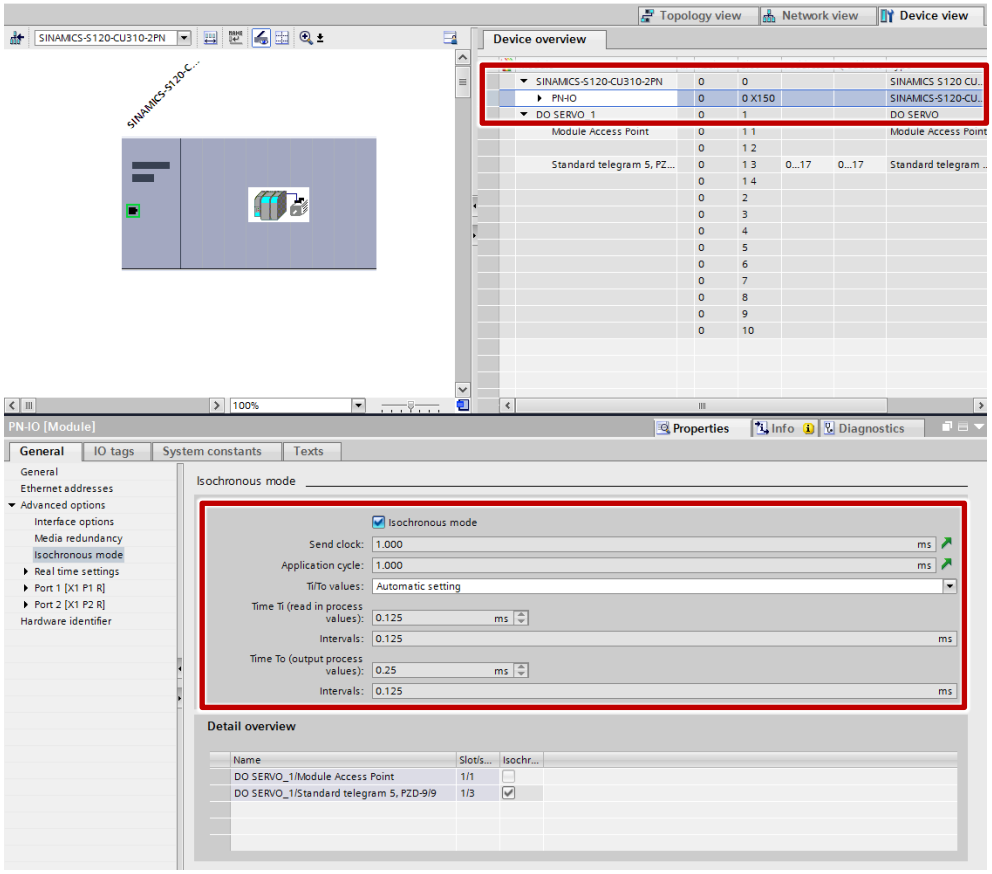
Via these settings you can then also define in the drive the Send cycle clock for the isochronous connection and the times Ti and To for inputting and outputting the signals and telegram data in the drive.

Note

In the detailed view of the clock synchronicity settings you can check whether the desired drive modules and axes are also in isochronous mode.

If necessary you can set a check mark in the overview to put the drive module or axis into isochronous mode.

Figure 2-4 Setting the clock synchronicity of the drive's PROFINET interface



2.2.3 Drive with Technology Object Axis

If the drive is controlled via the internal CPU Motion Control functions, when a technology object Axis is created and the drive is assigned to this technology object, the necessary settings are made automatically on the drive.

Figure 2-5 Assignment of the drive to the technology object Axis

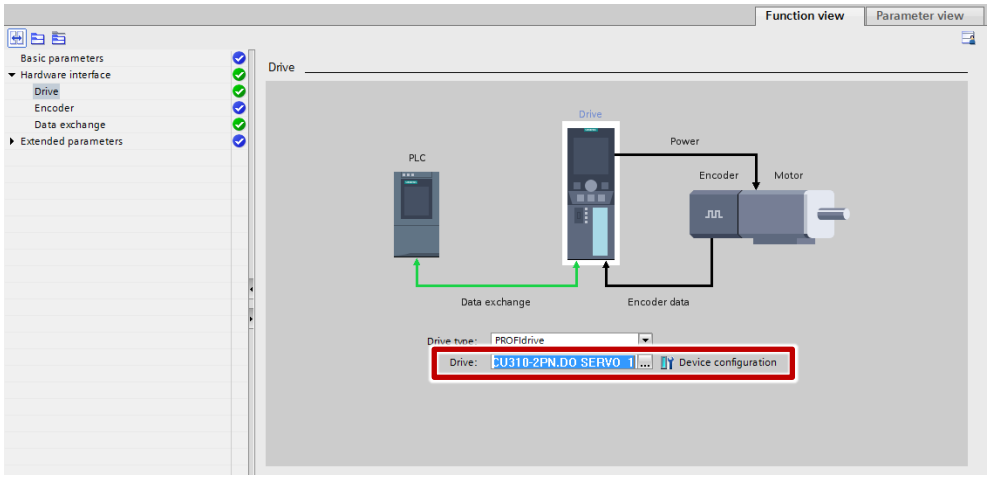
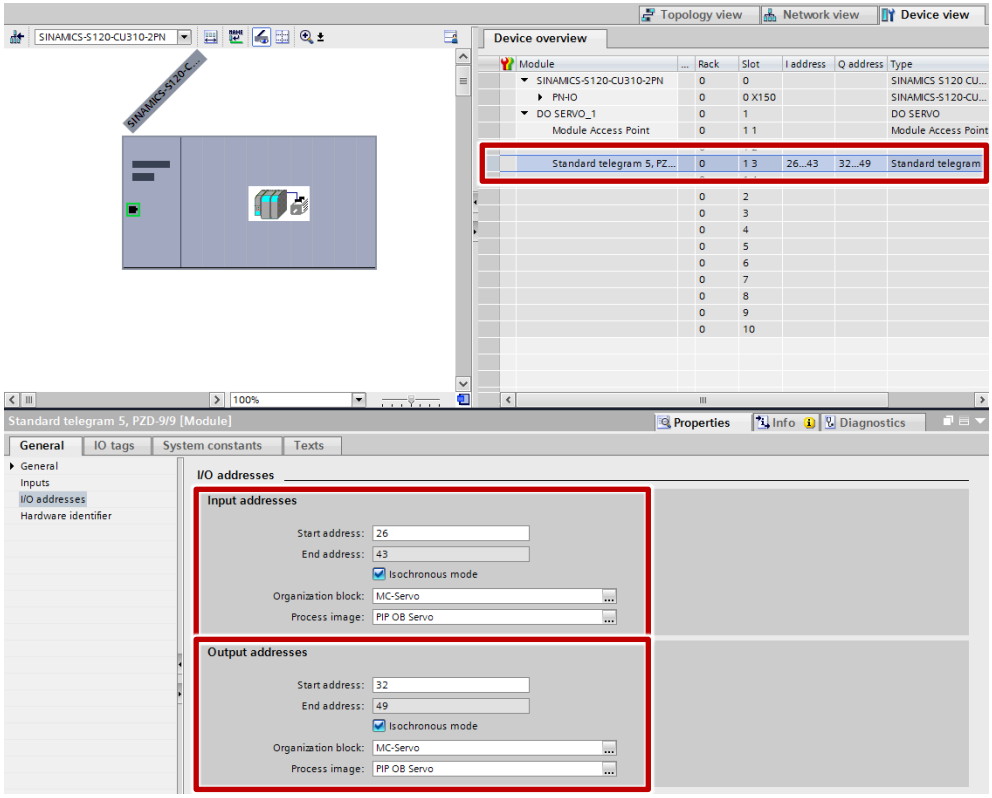
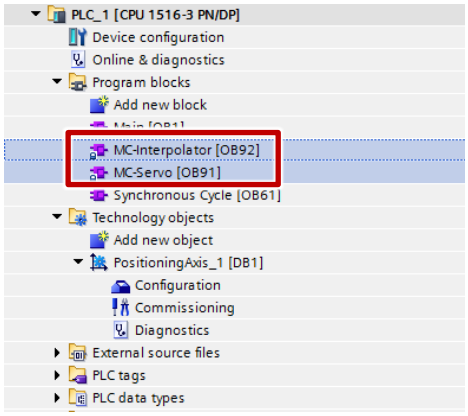


Figure 2-6 Setting the IO addresses of the drive telegram



There is another special feature when the drive is to be controlled via the internal CPU Motion Control functions. When you create an axis in the SIMATIC CPU, the organization blocks MC Servo and MC Interpolator are created automatically in the program.

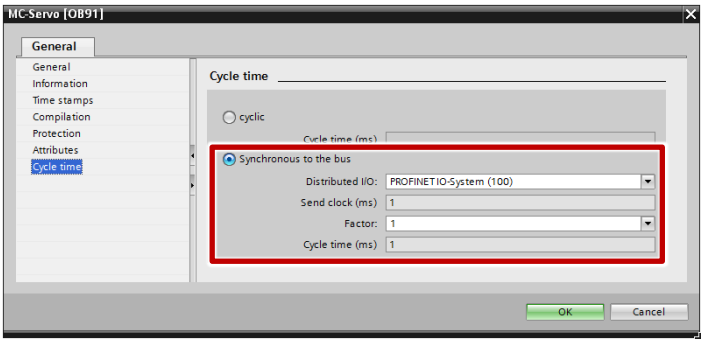
Figure 2-7 Program blocks when using internal CPU Motion functions



When the axis is created, the drive is then automatically assigned to the organization block MC Servo and the process image partition (PIP) OB Servo.

To completely set the isochronous connection between the technology object Axis in the SIMATIC CPU and the drive, you must set processing of the block to be synchronous with the data bus in the Properties of the organization block MC Servo which you reach via the pop-up menu of the block in the tree view.

Figure 2-8 Properties of the organization block MC Servo



2.2.4 Scaling of the Synchronous OB

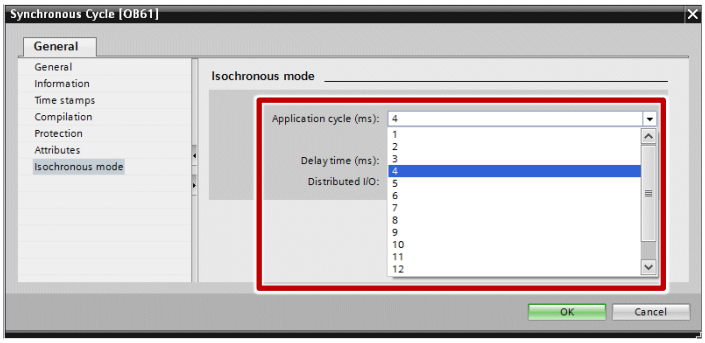
You can scale the application cycle of a synchronous OB to the Send cycle clock of an isochronous PROFINET IO system. You can set an integer multiple of the Send clock cycle as factor. For the application cycle you can use values up to 14 times that of the Send clock cycle (maximum 32 ms).

In this way you can reduce the SIMATIC CPU load and enable isochronous processing of signals in the CPU with a cycle time of the synchronous OB that is longer than the Send cycle clock of the isochronous PROFINET connection.

Scaling of the OB Synchronous Cycle

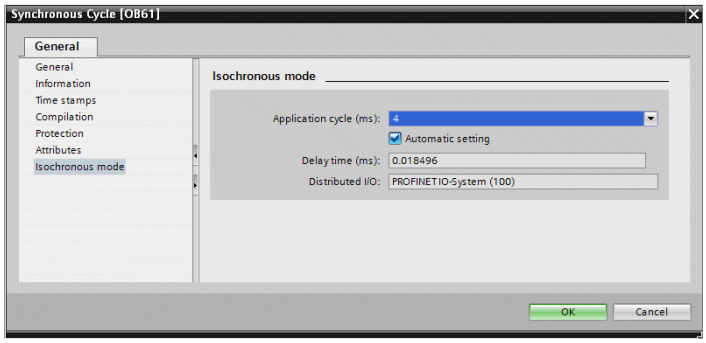
In the Properties of the organization block Synchronous Cycle – which you reach via the pop-up menu of the block in the tree view – you can set processing of the block to be synchronous with the data bus.

Figure 2-9 Setting of the scaling factor on the OB Synchronous Cycle



Via the factor you can reduce the SIMATIC CPU load by executing the organization block Synchronous Cycle. With the settings shown in the example the organization block is called only after every fourth data exchange via the data bus in the SIMATIC CPU.

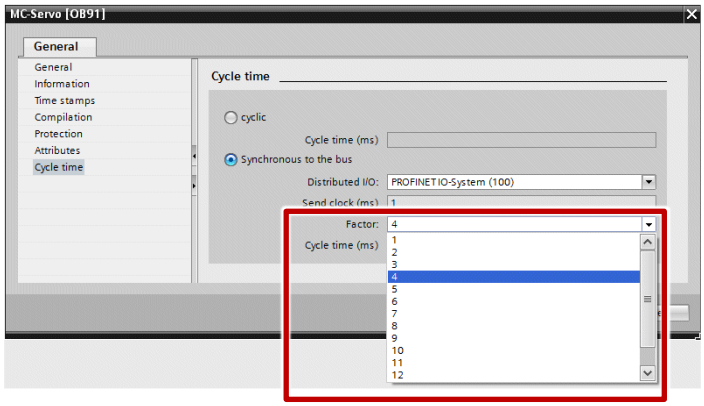
Figure 2-10 Setting of the scaling factor 4 and automatic delay



Scaling of the OB MC Servo

In the same way, in the Properties of the organization block MC Servo – which you reach via the pop-up menu of the block in the tree view – you can set processing of the block for integrated Motion Control of the SIMATIC CPU to be synchronous with the data bus.

Figure 2-11 Setting of the scaling factor on the OB MC Servo



Note However, the scaling factor of the organization block MC Servo also influences the axis control of the technology object Axis.
For example, if any interference on the axis can no longer be compensated satisfactorily, this might be due to a scaling factor that is set too high.

Simultaneous Utilization of both Synchronous OBs

If in an application both the isochronous processing of signals in the organization block Synchronous Cycle and the function of the integrated Motion Control of the SIMATIC CPU are used, you can likewise reduce the CPU load by setting a scaling of the synchronous OBs. Here you must set the scaling to the same factor on all the participating synchronous OBs that are to use or exchange data with each other.

Note If multiple synchronous OBs are to use data together, the scaling on all synchronous OBs must be set to the same value.

3 The Sync Domain

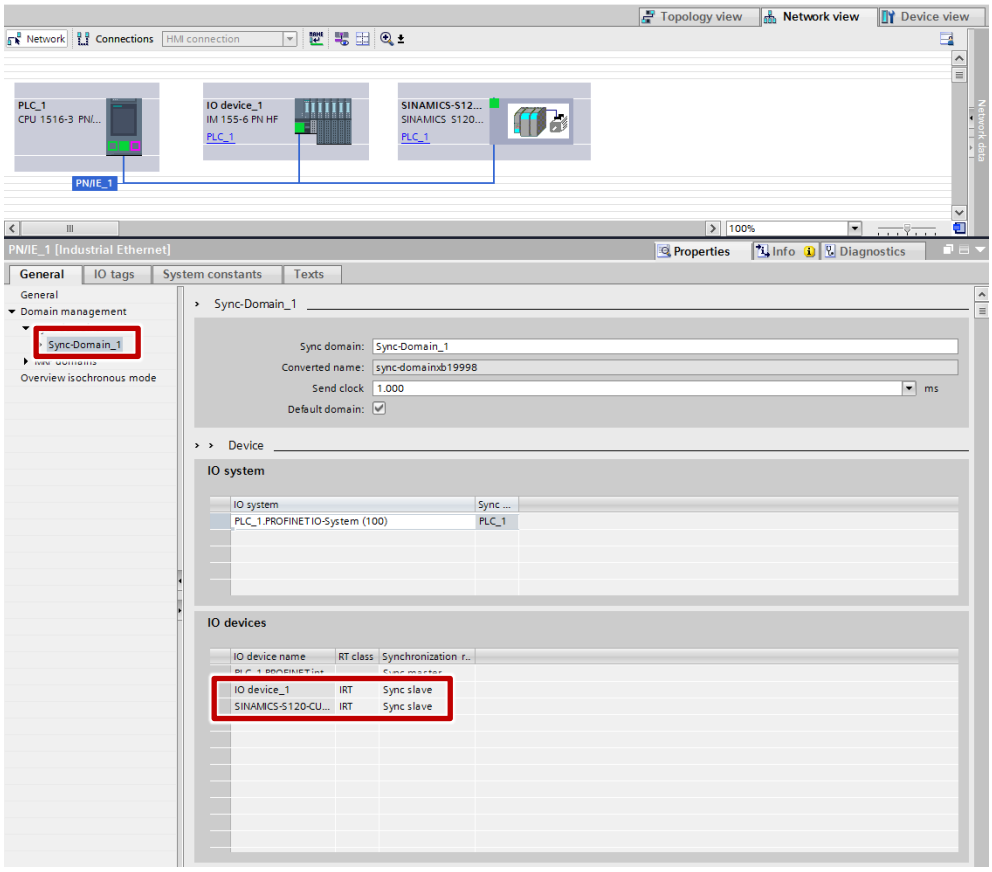
3.1 Setting a Sync Domain in the Network

Via the sync domain you assign the single distributed IO stations (sync slaves) to a SIMATIC CPU (sync master) with reference to the isochronous data exchange.

You get to the Properties of the sync domain in the Network View by double-clicking the required network.

In the display of the IO devices you can then define the sync master and the sync slaves assigned to the sync master.

Figure 3-1 Overview of the properties of a sync domain

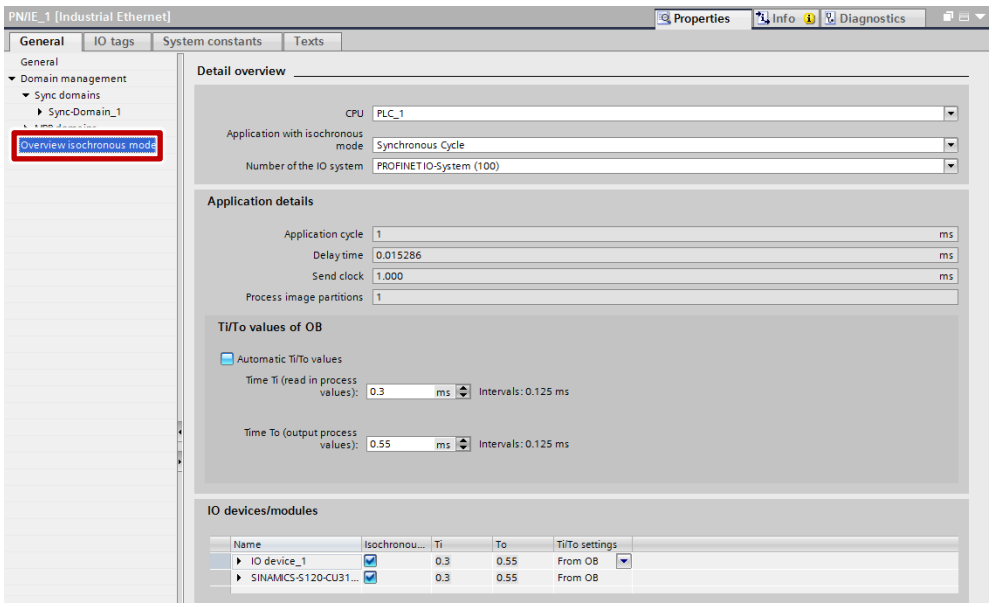


Note For isochronous data exchange IRT must always be selected for RT Class on the sync slaves.

3.2 Overview of the Clock Synchronicity Settings

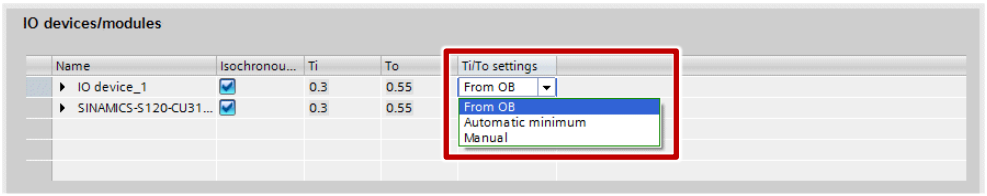
In the same Properties dialog you also have an overview of the clock synchronicity settings. Via this overview you can also have the settings made displayed. Here, however, cross-over settings in the sync domain are also possible.

Figure 3-2 Overview of the clock synchronicity settings



In the overview of the IO devices you can select one of the settings shown below in the Ti/To settings column.

Figure 3-3 Selecting the Ti/To setting

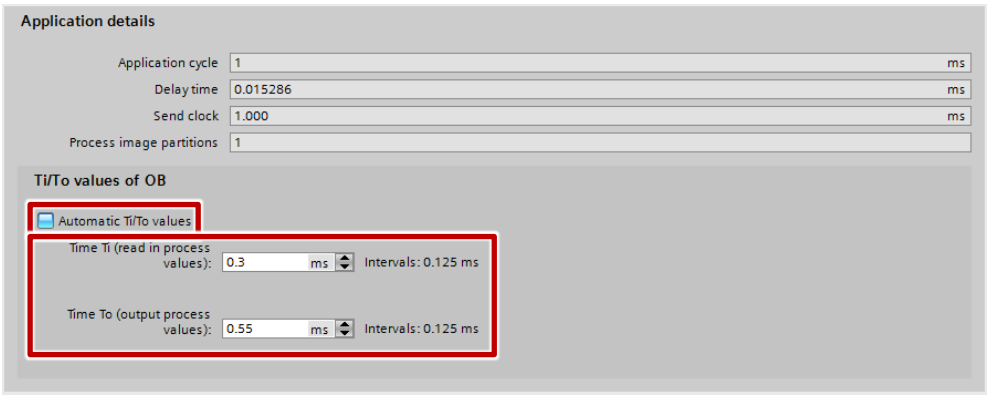


The meanings of the settings are as follows:

- Automatic Minimum:
  This is the default setting. The minimum time is selected each for the times Ti and To. The times are different in this setting for each distributed IO station and can be changed accordingly in the hardware configuration.
  If the times Ti and To are to be used in the user program, you must make sure that you use the correct Ti and To times assigned to the corresponding signal.
- Manual:
  In this setting you can specify the times Ti and To manually in the overview. The specified times must be greater than or equal to the possible minimum and must also be changed if necessary when changes are made in the hardware configuration.
- From OB:
  With this setting times Ti and To can be specified centrally for all the distributed IO stations.
  This setting is recommended when the times Ti and To are to be used in the user program as setting parameters or fixed values, because then the same times Ti and To are valid for each signal.

Central setting of the times Ti and To is done via the setting mask above the overview of the IO devices.

Figure 3-4 Central setting of times Ti and To



If the values are specified manually as well, for utilization of times Ti and To in the user program you can select an appropriate, easy-to-handle value (with few places after the decimal point).

4 Programming in the Synchronous OB

4.1 The SYNC_PI and SYNC_PO Functions

The system functions SYNC_PI and SYNC_PO can only be called in the synchronous OBs and are responsible for isochronous and consistent updating of the process image partition assigned to the synchronous OB.

You can precisely define the updating of the process image partition via the arrangement of the calls of the two system functions in the synchronous OB.

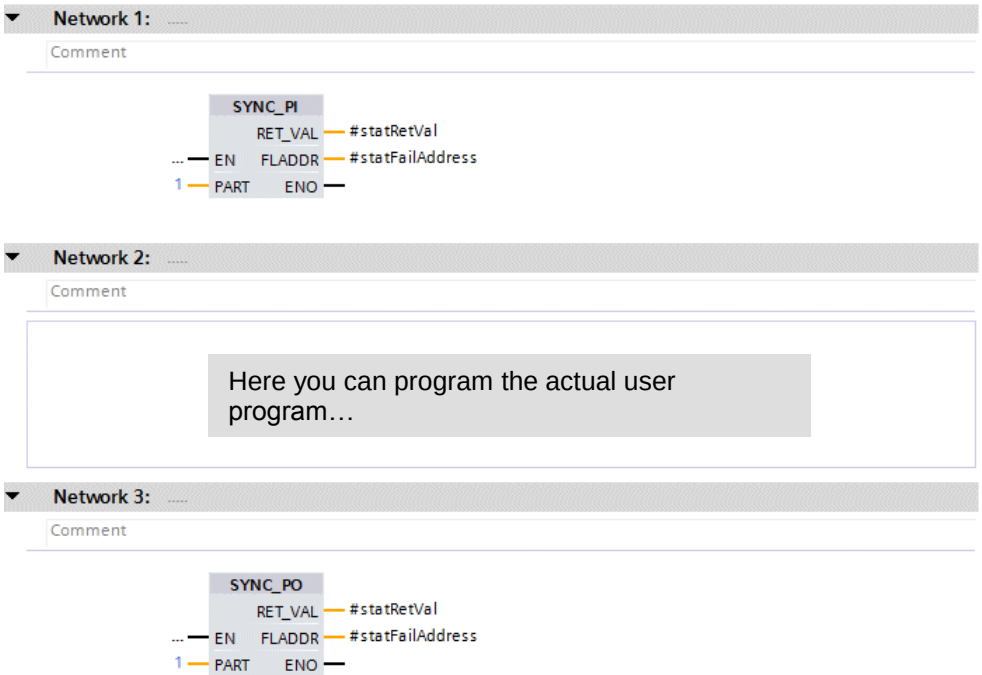
4.2 The IPO Model

Input (I):
With the IPO model, first in Network 1 the process image partition of the synchronous OB is input. In our example this is the process image partition 1 (PIP 1) as defined at the PART input of the SYNC_PI block. In this way the inputs of the process image partition are input and made available in the synchronous OB.

Processing (P):
In Network 2 the actual user program of the synchronous OB can be programmed.

Output (O):
In Network 3 the data changed via the user program is then output via the outputs of the process image partition 1 (PIP 1).

Figure 4-1 Programming the IPO model in the synchronous OB



Note

You should only use the IPO model if you are sure that the runtime of the synchronous OB is always clearly shorter than the Send clock of the communication network. The output data of the synchronous OB can thus always be transferred with the next bus cycle.

If you are working with scaling between application cycle (call clock of the synchronous OB) and bus cycle of the communication network, the above-mentioned demand can usually not be met. In this case you should use the OIP model.

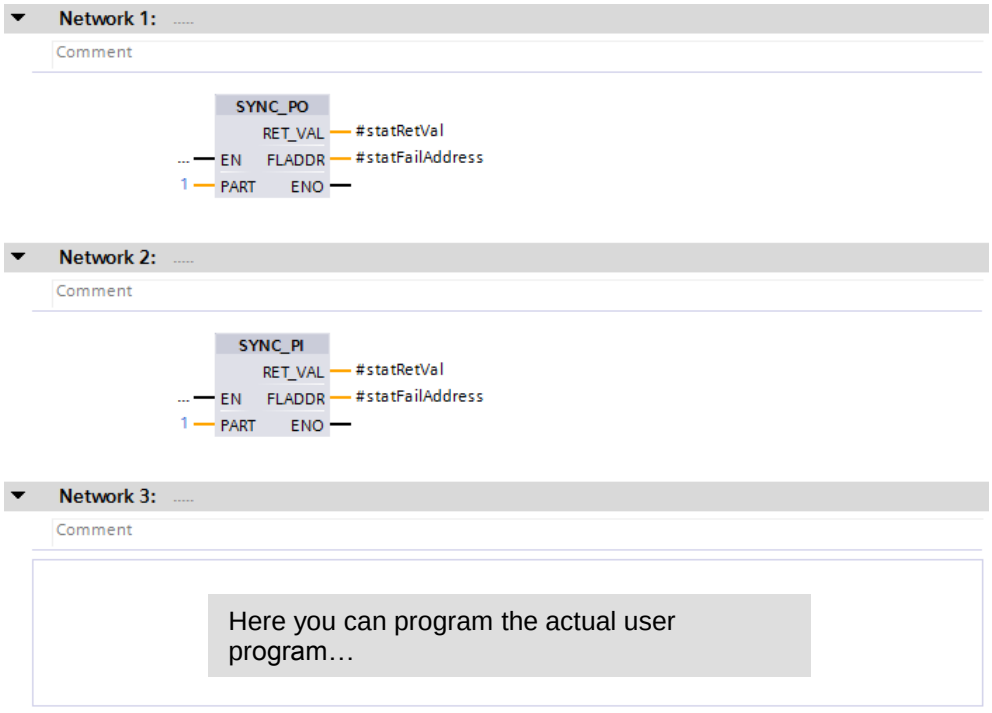
4.3 The OIP Model

Output (O):
With the OIP model, first in Network 1 the data changed via the user program in the previous cycle is output via the outputs of the process image partition 1 (PIP 1).

Input (I):
In Network 2 the inputs of the process image partition of the current cycle of the synchronous OB are input.

Processing (P):
In Network 3 the actual user program of the synchronous OB can be programmed.

Figure 4-2 Programming the OPI model in the synchronous OB



Note

If you are working with scaling between application cycle (call clock of the synchronous OB) and bus cycle of the communication network, utilization of the OIP model provides advantages.

With the OIP model, writing of the outputs and reading of the inputs of the process image partition in the synchronous OB is always done in the same bus cycle of the communication network. The data exchange with the process is therefore always deterministic, this means at a precisely fixed time with reference to execution of the synchronous OB.

4.4 Examples of the Models

We now give two examples to illustrate once again the choice of appropriate data processing model in an application.

4.4.1 Requirements

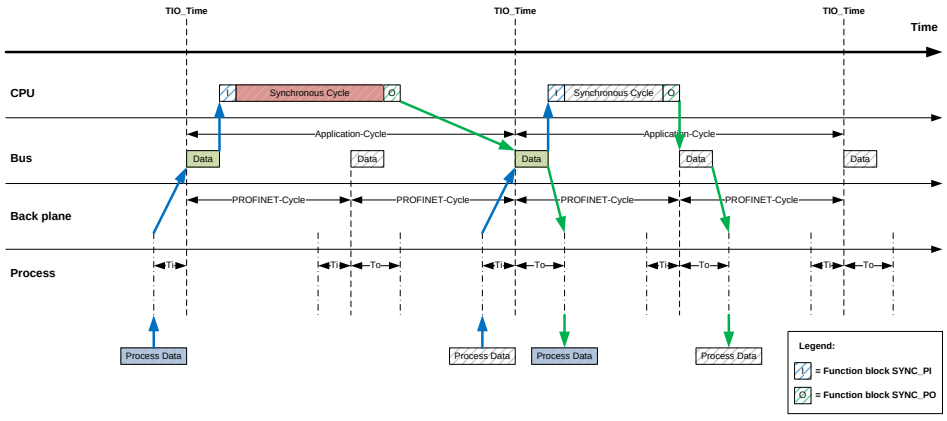
In the two examples given here are based on the following behavior of the applications and the following settings in the project:

- Between the application cycle (call clock of the synchronous OB) and the bus cycle of the communication network a scaling of 1:2 is set, which means that during the possible runtime of the application cycle there can be two data exchanges via the communication network.
- The user program in the application cycle is configured so that there might be great differences in the program runtimes in the synchronous OB.

4.4.2 Utilization of the IPO Model

The data flow and data processing for one application cycle is marked in color in the diagram below.

Figure 4-3 Data exchange in the synchronous cycle according to the OPI model



With this model the data exchange is made as close as possible to processing. The data is input, processed in the synchronous OB and then output again to the process in the next possible bus cycle.

However, if you compare the two consecutive synchronous OBs of different runtime lengths, you find that deterministic data exchange with the process is not possible with this choice of data processing model. If the runtime of the synchronous OB is long, there are two bus cycles between input of the process signals and output of the processed signals to the process. With a shorter synchronous OB runtime this process can also only require one bus cycle.

Conclusion

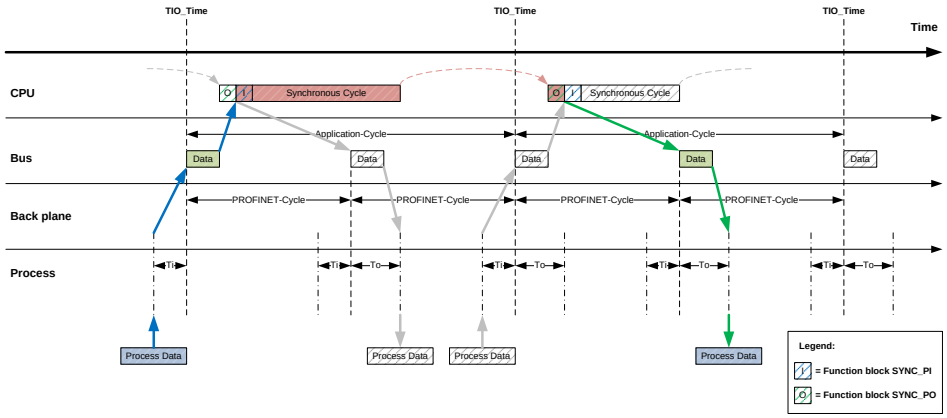
If scaling is used between the application cycle and the bus cycle and the execution time of the synchronous OB fluctuates due to the user program, then it is not recommended to use the IPO model.

Therefore the IPO model should only be used if the execution time of the synchronous OB is always constant or no scaling is set. Then this model is faster in processing process signals.

4.4.3 Utilization of the OIP Model

The data flow and data processing for one application cycle is also marked in color in the diagram below.

Figure 4-4 Data exchange in the synchronous cycle according to the OIP model



In the figure you immediately see the long time of three bus cycles between input of the process signals and output of the processed signals to the process. Signal exchange with the OIP model with scaling set is therefore slower than with the IPO model.

However, the deterministic behavior of this model is immediately noticeable through the fixed output of the signals from the previous synchronous OB run before input of the input signals and processing in the synchronous OB. Independent of the synchronous OB runtime, the signals are input and output respectively from and to the process always at the same time.

Conclusion

Despite its slower data exchange compared with the IPO model, the OIP model is best for data processing using scaling and for synchronous OB runtimes of different lengths.