STEP 7 Block library "Plant Communication Concept V3.0"

User documentation

Warranty and liability

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.

If there are any deviations between the recommendations provided in these Application Examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document. Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of the Siemens AG.

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks. In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens’ products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens’ guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens’ products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer’s exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity.
# Table of Contents

<table>
<thead>
<tr>
<th>Warranty and liability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
</tr>
<tr>
<td>1.1 Overview</td>
</tr>
<tr>
<td>1.2 System requirements</td>
</tr>
<tr>
<td>1.3 Supported communication protocols</td>
</tr>
<tr>
<td>1.3.1 Open Ethernet TCP communication (=OC)</td>
</tr>
<tr>
<td>1.3.2 S7 Communication (=S7C)</td>
</tr>
<tr>
<td>1.3.3 PROFINET IO communication (=IDC)</td>
</tr>
<tr>
<td>2 Library</td>
</tr>
<tr>
<td>2.1 Versioning and compatibility</td>
</tr>
<tr>
<td>2.2 PCC block structure</td>
</tr>
<tr>
<td>2.3 Block nesting</td>
</tr>
<tr>
<td>2.4 Block protection</td>
</tr>
<tr>
<td>2.5 Data flow</td>
</tr>
<tr>
<td>2.6 Message frame structure</td>
</tr>
<tr>
<td>3 Blocks</td>
</tr>
<tr>
<td>3.1 Data structure “UDT_PCC”</td>
</tr>
<tr>
<td>3.1.1 Configuration area</td>
</tr>
<tr>
<td>3.1.2 Diagnostics area</td>
</tr>
<tr>
<td>3.1.3 “SUB_CALL_INTERFACE” range</td>
</tr>
<tr>
<td>3.1.4 “BUFFER_SEND_DATA” range</td>
</tr>
<tr>
<td>3.1.5 “BUFFER_TEMP_DATA” range</td>
</tr>
<tr>
<td>3.1.6 “BUFFER_RCV_DATA” range</td>
</tr>
<tr>
<td>3.1.7 “BUFFER_RCV_SORTED” range</td>
</tr>
<tr>
<td>3.2 “FB_PCC_MAIN” block</td>
</tr>
<tr>
<td>3.3 Block “FB_PCC_SUB_CALL”</td>
</tr>
<tr>
<td>3.3.1 Open Ethernet TCP communication with T-blocks (OCT)</td>
</tr>
<tr>
<td>3.3.2 Open Ethernet TCP communication with send/receive modules (OCSR)</td>
</tr>
<tr>
<td>3.3.3 S7 communication (S7C)</td>
</tr>
<tr>
<td>3.3.4 PROFINET IO communication with direct IO access (IDC)</td>
</tr>
<tr>
<td>3.3.5 PROFINET IO communication with transfer blocks for CP (IDC_CP)</td>
</tr>
<tr>
<td>3.4 “FB_PCC_MAIN_CALL” block</td>
</tr>
<tr>
<td>4 Application in the project:</td>
</tr>
<tr>
<td>5 Configuration sample</td>
</tr>
<tr>
<td>5.1 Data interface (PDI)</td>
</tr>
<tr>
<td>5.2 Communication (PCC)</td>
</tr>
<tr>
<td>5.3 Parameter assignment</td>
</tr>
<tr>
<td>5.4 Configuration of the connections</td>
</tr>
<tr>
<td>5.5 Creating a TCP connection</td>
</tr>
<tr>
<td>5.6 Create S7 connection</td>
</tr>
<tr>
<td>5.7 Hardware configuration I-device</td>
</tr>
<tr>
<td>5.8 Error messages used</td>
</tr>
<tr>
<td>5.9 Configuration error</td>
</tr>
<tr>
<td>5.10 Runtime error</td>
</tr>
<tr>
<td>6 Appendix</td>
</tr>
<tr>
<td>6.1 Service and Support</td>
</tr>
<tr>
<td>6.2 Links and Literature</td>
</tr>
<tr>
<td>6.3 Change documentation</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Overview

The PLC modules, which are structured under the title "Plant Communication Concept (= PCC)”, are used for the unified communication of the "Plant Data Interface (PDI)" for the "Line Integration Concept (= LIS)" of the food and beverage industry.

The "Plant Data Interface (PDI)" defines specified content that complies with international standards such as those of the OMAC User Group or the Weihenstephan Standard, and transforms these into concrete data structures.

When doing this, it is important to transfer content reliably and to supply the interface with the correct information from the application. Based on the information in the interface, evaluations of the reliability of the machine and, for example, the frequency of specific fault messages, are frequently carried out. The information from the interface is connected in production mode with time models, for example, to calculate the availability of machines.

These unified data structures make it possible to combine individual machines of a production or packaging line in a uniform and interchangeable manner with a central administration / monitoring structure.

To keep the data exchange as efficient, safe and easy as possible, the "Plant Communication Concept (= PCC)" was developed. These components help the machine manufacturer (= OEM) to uniformly make the different data structures available on various combinations of hardware and interfaces.

Benefits

A uniform solution, which facilitates understanding and handling, has been created for all standard SIMATIC control series. Data consistency, transmission security, stability and functional monitoring are ensured.

In the current version 3.0, different protocols and different transmission media are also supported.

The library was created for minimal resource consumption and high flexibility.

To make the integration into the control program as sustainable as possible, only the absolutely necessary processing is carried out and, if possible, it is distributed over several CPU cycles. Furthermore, the memory consumption was reduced as much as possible through the scalability of the data volume and effective programming.

1.2 System requirements

The PLC blocks are available in a STEP 7 V5.5 and in another STEP 7 V14 library and can run on S7-300/400/1200/1500 (or compatible) controllers. All use-relevant program parts of the STEP 7 V5.5 library were created with the programming language "AWL" and can therefore be used without optional additional programs from STEP 7. For the STEP 7 V14 version, the programming language "SCL" was used uniformly. All blocks are intended for "optimized block access". Blocks are available in both libraries for the S7-300/400 control series.

Due to the block structure, the actual transmission can be adjusted relatively easily to unpredictable requirements (e.g. new firmware or newer communication blocks). The PCC library is therefore not subject to any hardware limitations. Due to the use of special functions in STEP 7 V14, however, the blocks can only be used as of CPU firmware version 4.2.
1 Introduction

The transfer does not require and has no influence on the CPU system time. In the example project, an NTP synchronization was parameterized only to illustrate the general requirement.

Required resources for the control:

- **Cycle time 0-x ms** (asynchronous machining)
  A possible maximum CPU load depends on the scope and type of configuration. In typical use, however, this is less than 1 ms.

- **Work memory** (Dependent on the type and version)
  PDI data: 0.3 KByte
  Base program including calls and data: approx. 11 KByte
  System blocks: 3.5 KByte (depending on the protocol)

- Average **communication load** (PDI basic configuration)
  of the machine to line control: 265 bytes/sec.
  From line control to the machine: 31 bytes/sec.

- One **communication interface** (preferably PROFINET)

1.3 Supported communication protocols

Due to the different technical requirements, 3 protocols were added to the standard library. Due to the block structure, it can also be supplemented with additional protocols or blocks.

Decision-making aids

- Most flexible application: Open TCP communication
- Network independence: S7 communication
- Quickest communication: PROFINET IO communication

Explanation of the type of connection parameters

- For “parameterized connections”, the communication partners are specified during engineering and corresponding parameters are set in specific dialogs of the hardware or network configuration. The connection is handled by the firmware.
- For “connections without parameterization”, the connection setup is also handled in the user program. All parameters (e.g. IP address of the partner) can be changed in the program.

1.3.1 Open Ethernet TCP communication (=OC)

Features

- Open default (communication also possible with external controllers)
- Communication via CPU interface, CM or via CP
- Use even without configured connections

Further distinction

For S7-300 / 400, a distinction must also be made:

- Use of the integrated CPU interface (= OCT)
  So-called “T-blocks” are used
1 Introduction

- Use of communications processors (=OCSR)
  Communication takes place using send/receive blocks and a configured connection

1.3.2 S7 Communication (=S7C)

Features
- Network-independent user interface: Identical handling for PN/IE, PB and MPI
- Communication via CPU interface or via CP(CM)
- Communication via configured connections (S7 connection)
- Conditionally routable between the bus systems
- Acknowledgment by the remote application of the receipt of the data

Constraint
Protocol is not supported by all controllers (e.g. S7-1200).

1.3.3 PROFINET IO communication (=IDC)

Features
- Parameterized cyclic communication in the hardware configuration
- Transmission is ensured by the firmware - user program copies the data

Constraint
All network components used (including switches) must support the protocol.
For each process data (PDI) interface to be transferred, one communication instance must be called up on each side. If several PDI interfaces are to be exchanged with a remote station (= machine), then several communication instances have to be implemented. One of these instances, present as a copy template "FB_PCC_MAIN_CALL", thus transmits exactly one complete "UDT_PDI" instance (= copy template "DB_PDI").

Example (One line corresponds to one communication instance):

Table 2-1

<table>
<thead>
<tr>
<th>Machine (OEM)</th>
<th>Data</th>
<th>Block calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erector</td>
<td>DB_PDI</td>
<td>FB_PCC_MAIN_CALL</td>
</tr>
<tr>
<td>Packaging</td>
<td>DB_PDI_PREPARE</td>
<td>FB_PCC_MAIN_CALL_PREPARE</td>
</tr>
<tr>
<td></td>
<td>DB_PDI_PACK</td>
<td>FB_PCC_MAIN_CALL_PACK</td>
</tr>
<tr>
<td></td>
<td>DB_PDI_CLOSE</td>
<td>FB_PCC_MAIN_CALL_CLOSE</td>
</tr>
<tr>
<td>Palletizers</td>
<td>DB_PDI</td>
<td>FB_PCC_MAIN_CALL</td>
</tr>
</tbody>
</table>

Table 2-2

<table>
<thead>
<tr>
<th>Line control</th>
<th>Block calls</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erector</td>
<td>FB_PCC_MAIN_CALL_M101</td>
<td>DB_PDI_M101</td>
</tr>
<tr>
<td>Packaging</td>
<td>FB_PCC_MAIN_CALL_M102</td>
<td>DB_PDI_M102</td>
</tr>
<tr>
<td></td>
<td>FB_PCC_MAIN_CALL_M103</td>
<td>DB_PDI_M103</td>
</tr>
<tr>
<td></td>
<td>FB_PCC_MAIN_CALL_M104</td>
<td>DB_PDI_M104</td>
</tr>
<tr>
<td>Palletizers</td>
<td>FB_PCC_MAIN_CALL_M105</td>
<td>DB_PDI_M105</td>
</tr>
</tbody>
</table>

All PCC blocks support the option "optimized block access" and do not have to be retentive. Only the PDI data should be stored retentively.

To save the CPU resources, the processing may be divided over several call cycles depending on the necessity of the function.

The internal processing of the communication blocks is also distributed over several cycles. All the block calls should therefore be incorporated in the main program processing cycle (= OB1) as shown in the configuration example. This prevents the receipt of data from hindering the sender in the optimal communication process, and the data buffers are used optimally.

### 2.1 Versioning and compatibility

Each block of the library has a version identifier in the metadata. This may also be different from the version of the PCC library. For details, see also the block comments with the "version history"

The communication itself also distinguishes the version of the PDI data as well as the version of the communication message frame. These versions are partially checked in the program and give rise to corresponding error messages in the event of incompatibility.

The completely revised communication message frame from PCC V3 is not compatible with previous PCC versions. Due to the simplified application, replacing the communication blocks is recommended.
2.2 PCC block structure

The PCC blocks are identical for both communication endpoints. They are divided into the following areas:

Data block of the process data interface

This is instantiated in the copy template "DB_PDI" as data type "UDT_PDI". The user data is stored in this data area for transmission. These vary according to the standard used (OMAC or Weihenstephan).

The freely selectable structure is divided primarily into:
1. Basic data (BASIC) - minimum amount of data for connection
2. Control data (LCU) – data used for optional control of the machine
3. Energy data (PEC) - contains programmable energy measuring points at runtime
4. Parameter data (PARA) - measured values or additional user data
5. Plant-specific data - flexible data area for additional plant-specific data

For details on the structure and the data content, please refer to the documentation of the corresponding standard.

Communication data block

This is instantiated in the copy template “DB_PCC” as data type “UDT_PCC". This area contains configuration, diagnostics, and data caching areas for a communication instance. The arrays defined in this area are used dynamically and can be resized according to the PDI setup.

Blocks for data processing

The block “FB_PCC_MAIN" copies the data and controls the communication. Likewise, the communication connection is monitored and possibly also controlled depending on the protocol.

Blocks for calling the communication

Depending on the protocol used and the hardware used, the corresponding subroutines are called in “FB_PCC_SUB_CALL". For special cases, the received raw data can still be edited or changed (e.g.: Integration of external controllers or versions)

Communication system blocks

To ensure a consistent library, all necessary Siemens system components (e.g.: "FC_TSEND", "FB_BSEND", "FC_PNIO_SEND", ...) are also included. Depending on the application, these can also be exchanged individually for more up-to-date modules.

Calling the blocks

The copy / template “FB_PCC_MAIN_CALL" is used for configuration and contains a multi-instance call of the PCC program parts.
2.3 Block nesting

The following representation illustrates the nesting of the blocks as used in the example project. The call is not mandatory, but it is recommended that the blocks be called in the same execution cycle.

The block “FB_PCC_SUB_CALL” must be selected protocol-specific or depending on the hardware used. Due to the nesting selected, all block numbers can be individually adapted in STEP 7 V5.5 as well.

Table 2-3

<table>
<thead>
<tr>
<th>OB_CYCLE_EXEC (OB1)</th>
<th>FB_PCC_MAIN_CALL (Know-how-protected)</th>
<th>FB_PCC_SUB_CALL</th>
<th>FC_SERIALIZE*)</th>
<th>FC_DESERIALIZE*)</th>
<th>FX_MOVE_BLK_VARIANT*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OB_CYCLE_EXEC (OB1)</td>
<td>FB_PCC_MAIN_CALL (Know-how-protected)</td>
<td>FB_PCC_SUB_CALL</td>
<td>FC_SERIALIZE*)</td>
<td>FC_DESERIALIZE*)</td>
<td>FX_MOVE_BLK_VARIANT*)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FB_TCON</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FB_TDISCON</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FB_TSEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FB_TRCV</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) Only STEP 7 TIA Portal

2.4 Block protection

The block protection for “FB_PCC_MAIN” is only used for version or change protection of the block and to ensure that the planned use is adhered to. If compilation is necessary for the hardware used (= error message when loading the program), then you can cancel this protection with the password “Siemens!F&B”.

2.5 Data flow

The payload data length to be transmitted depends on the actual extent required. Reserve areas are not transferred. The communication block detects from its interconnection whether PDI areas should be transmitted or not. If connections are not interconnected or are “zeroed”, the data is regrouped accordingly.

The length of the data is checked for plausibility and compared on the receiving side. A different interconnection of the blocks leads to an error message.

The connection between preparation and transmission is illustrated below. The transfer direction of the individual PDI areas is also displayed.
2.6 Message frame structure

Regardless of the protocol, the PDI data to be transmitted is still packed into a message frame which ensures the following:

- Version control
- Prevention of different parameterization
- Completeness of data
- Optimization of the temporal monitoring
- Monitoring of data transmission in both directions
- Message frame consistency for small PDU sizes of the transmission path

For this purpose, additional “administrative data” are transmitted at the beginning and at the end of the user data as follows:

Table 2-4

<table>
<thead>
<tr>
<th>Message frame</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header (8 bytes)</strong></td>
<td>Version (1 byte)</td>
</tr>
<tr>
<td></td>
<td>Type (1 byte – 16#FE)</td>
</tr>
<tr>
<td></td>
<td>Count M to LC (1 byte)</td>
</tr>
<tr>
<td></td>
<td>Count LC to M (1 byte)</td>
</tr>
<tr>
<td></td>
<td>Length (2 bytes)</td>
</tr>
<tr>
<td></td>
<td>Max send time (2 bytes)</td>
</tr>
<tr>
<td><strong>Used PDI data (88 – x byte)</strong></td>
<td>Basic</td>
</tr>
<tr>
<td></td>
<td>LC</td>
</tr>
<tr>
<td></td>
<td>Parameter</td>
</tr>
<tr>
<td></td>
<td>PEC</td>
</tr>
<tr>
<td></td>
<td>Plant-specific</td>
</tr>
<tr>
<td></td>
<td>End Char (1 Byte – 16#FC)</td>
</tr>
</tbody>
</table>

**Version (1 byte)**

Specifies the version number of the message frame structure and ensures the correct evaluation of the transmitted data structure.
Message frame type (1 byte)

Used to identify the type of user data contained in the message frame. There is currently only one type of message frame. A constant hexadecimal "FE" (= decimal 254) is transmitted.

Message frame counter machine to line control (1 byte)

Circulating transmission counter which indicates the message frames sent by the machine control. This is copied unchanged by the line control, after a validity check of the received message frame, into the send message frame. The communication is thus recognized on both sides as faulty, even if only one transmission direction does not work. The validity of the message frame is recognized by the correct length as well as by the content of the "header".

Message frame counter line control to the machine (1 byte)

Circulating transmission counter which indicates the message frames sent by the line control. This is copied unchanged by the machine control, after a validity check of the received message frame, into the send message frame. The communication is thus recognized on both sides as faulty, even if only one transmission direction does not work. The validity of the message frame is recognized by the correct length as well as by the content of the "header".

Length (2 bytes)

Specifies the number of bytes sent and is used, among other things, to check that the block has been connected to the same PDI structure on both sides.

Maximum transmission time (2 bytes)

Specifies the maximum transmission pause (in seconds) that there can be between 2 transmission cycles. This value is needed for an optimization of the communication monitoring, because the parameterization and protocol selection can result in very different transmission speeds.

End character (1 byte)

To ensure complete transmission of the data, a constant hexadecimal "FC" (= decimal 252) is sent at the end.
3 Blocks

3.1 Data structure “UDT_PCC”

This data structure defines a communication instance (= communication for a PDI interface). This data type can be stored with the “optimized block access” option and the data does not have to be retained.

3.1.1 Configuration area

This area contains all the parameters that determine the mode of transmission.

Note

Changes to communication parameters are sometimes only accepted by the respective blocks after the connection has been restarted or the “instance data” (or CPU restart) has been loaded. For additional information, refer to the online help of the communication blocks used in “FB_PCC_SUB_CALL”.

- “Enable” (default: “True”)
  Enable establishment of connection and communication. If this data point is “False”, the communication is also cleared during active connection control (with OCT).

- “Reset” (default: “False”)
  This data point interrupts the connection. All diagnostic counters and internally stored information are reset.

- “ControllerSide” (default: “False”)
  This indicates whether the block is used on the machine side (= interconnection “False”) or on the side of the line control (= interconnection “True”).

- “EnableCheckChange” (default: “True”)
  The transmission is basically cyclical, but if important information changes in the PDI data area (e.g.: commands), the transmission is triggered immediately. Monitoring for changes can be enabled or disabled with this data point if required.

- “EnableCCPlantCmd” (default: “True”)
  For the area of plant-specific payload data from the line control to the machine, due to the undefined content, the change monitoring for the immediate transmission can be set separately.

- “EnableCCPlantStat” (default: “False”)
  For the area of plant-specific payload data from the machine to the line control, due to the undefined content, the change monitoring for the immediate transmission can be set separately.

- “PDI_Type” (default: 1=OMAC)
  To check the plausibility of the interconnection or to enable change monitoring, the PDI type must be specified:
  - OMAC PDI Interface Version 2
  - Weihenstephan PDI Interface Version 2
- **“ConnectionId”** (default: 16)
  Identification number of the communication connection. For configured communication connections (OCSR or S7C), the number of the configured connection must be entered (Reference 5.2.2). If the connection is controlled by the user program (= PCC) (OCT), then a CPU-wide unique, but freely assignable number must be specified. The possible values can be found in the technical specification of the CPU used.

- **“MinPauseSendMs”** (default: 500 ms)
  The minimum pause between 2 send message frames in milliseconds is specified here. This pause time also prevents continuous transmission in the event of active data change monitoring in case of unexpected constant changes.

- **“MaxPauseSendMs”** (default: 1500 ms)
  The maximum pause between 2 send message frames in milliseconds is specified here. This pause time determines how often the data is cyclically transmitted.

**Note**

Interaction of the transmission rates as an example:

A new “Start” command is detected by change monitoring and transmitted immediately. However, the minimum pause time “MinPauseSendMs” is used to ensure that pauses between the message frames are also observed so as not to block the network.

A current measurement of energy consumption, however, is not monitored for change due to the constant fluctuations. This is transmitted periodically after the maximum pause time “MaxPauseSendMs” has expired.

- **“OPEN_COM.PartnerIP”** (default: 192.168.0.0)
  Specifies the IPv4 address of the partner station when using open TCP communication.

- **“OPEN_COM.PortLineCSide” / “OPEN_COM_T.PortLineCSide”** (default: 2200)
  Specifies the TCP port number of the line control when using the open Ethernet TCP communication (OCT).

- **“OPEN_COM.PortMachineSide” / “OPEN_COM_T.PortMachineSide”** (default: 2100)
  Specifies the TCP port number of the machine control when using the open Ethernet TCP communication (OCT).

- **“OPEN_COM.HwIdentifier”** (default: 64)
  Specifies the hardware identifier interfaces when using the open Ethernet TCP communication (OCT) (not to be confused with the hardware ID of the single port). A symbolic assignment can also be made using the system variables.

- **“OPEN_COM_T.LocalDeviceld”** (default: 2 corresponds to S7-300 CPU interface)
  Specifies the device ID of the hardware when using the open Ethernet TCP communication (OCT). Reference STEP 7 Online help of the block “FB_TCON” or at https://support.industry.siemens.com/cs/ww/en/view/51339682.

- **“OPEN_COM_SR.LADDR”** (default: 0)
  Specification of the access address of the communication processor when using the open Ethernet TCP communication (OCSR) with send/receive blocks.
• “S7_COM.SR_PairID.CtM” (default: 1)
  When using S7 communication, several send/receive blocks can be used over
  a single communication connection. However, an identical and unique number
  must be specified at both communication endpoints for this. At this point, the
  identification number for the transmission from the line control to the machine
  must be specified.

• “S7_COM.SR_PairID.MtC” (default: 2)
  As explained above, when using the S7 communication, an identification
  number for the transmission from the machine to the line control must be
  specified here.

• “ID_COM.FirstInByte” (default: 0)
  For PROFINET IO communication (IDC), the peripheral start address (number
  of the 1st byte) is specified for the input range here.

• “ID_COM.FirstOutByte” (default: 0)
  For PROFINET IO communication (IDC), the peripheral start address (number
  of the 1st byte) is specified for the output range here.

• “ID_CP_COM.LADDR” (default: 0)
  Specification of the access address of the communication processor when
  using the open PROFINET IO communication (IDC_CP) via CP.

3.1.2 Diagnostics area

This area contains all the information needed to correct or search for causes of a
communication error.

• “ComSumError”
  If an error is detected during communication or parameterization, this is also
  output in binary form. For short-term disconnections such as a restart or the
  like, no communication error is reported.

• “Connected”
  An actively established connection to the remote station is displayed at this
data point.
  This is only relevant if the connection is controlled by the user program (=
PCC) (OCT), otherwise “True” is always displayed.

• “ConfigError”
  The currently detected configuration error is output here (see also 5.4.1):

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>8028</td>
<td>Invalid value for FB parameter “PDI_TYPE”</td>
</tr>
<tr>
<td>8032</td>
<td>Interconnection to the buffer of the transmit data is not a valid DB link</td>
</tr>
<tr>
<td>8034</td>
<td>Interconnection to the buffer of the received data is not a valid DB link</td>
</tr>
<tr>
<td>8036</td>
<td>Interconnection to PDI_Basic is not a valid DB link</td>
</tr>
<tr>
<td>8037</td>
<td>Interconnection to PDI_LCU_STAT is not a valid DB link</td>
</tr>
<tr>
<td>8038</td>
<td>Interconnection to PDI_LCU_CMD is not a valid DB link</td>
</tr>
<tr>
<td>8039</td>
<td>Interconnection to PDI_PARA is not a valid DB link</td>
</tr>
<tr>
<td>803A</td>
<td>Interconnection to PDI_PEC is not a valid DB link</td>
</tr>
<tr>
<td>803B</td>
<td>Interconnection to PDI_PLANT_STAT is not a valid DB link</td>
</tr>
<tr>
<td>803C</td>
<td>Interconnection to PDI_PLANT_CMD is not a valid DB link</td>
</tr>
<tr>
<td>8044</td>
<td>Length of the output range for the transmission data is invalid</td>
</tr>
<tr>
<td>8045</td>
<td>Length of the input range of the received data is invalid</td>
</tr>
<tr>
<td>8046</td>
<td>Length of the temporary data buffer is invalid</td>
</tr>
<tr>
<td>8047</td>
<td>Length of the transmit data buffer is invalid</td>
</tr>
<tr>
<td>8048</td>
<td>Length of the raw data receive buffer is invalid</td>
</tr>
<tr>
<td>8049</td>
<td>Length of sorted receive data buffer is invalid</td>
</tr>
<tr>
<td>Code</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>8050</td>
<td>Interconnection to OMAC PDI Basic has an invalid data length</td>
</tr>
<tr>
<td>8051</td>
<td>Interconnection to OMAC PDI LCU_CMD has an invalid data length</td>
</tr>
<tr>
<td>8052</td>
<td>Interconnection to OMAC PDI LCU_STAT has an invalid data length</td>
</tr>
<tr>
<td>8053</td>
<td>Interconnection to OMAC PDI PARA has an invalid data length</td>
</tr>
<tr>
<td>8054</td>
<td>Interconnection to OMAC PDI PEC has an invalid data length</td>
</tr>
<tr>
<td>8055</td>
<td>Interconnection to WS PDI Basic has an invalid data length</td>
</tr>
<tr>
<td>8056</td>
<td>Interconnection to WS PDI LCU_CMD has an invalid data length</td>
</tr>
<tr>
<td>8057</td>
<td>Interconnection to WS PDI LCU_STAT has an invalid data length</td>
</tr>
<tr>
<td>8058</td>
<td>Interconnection to WS PDI PARA has an invalid data length</td>
</tr>
<tr>
<td>8059</td>
<td>Interconnection to WS PDI PEC has an invalid data length</td>
</tr>
<tr>
<td>805A</td>
<td>Length of the temporary data buffer is too small</td>
</tr>
<tr>
<td>805B</td>
<td>Length of the transmit data buffer is too small</td>
</tr>
<tr>
<td>805C</td>
<td>Length of the raw data receive buffer is too small</td>
</tr>
<tr>
<td>805D</td>
<td>Length of the output range for the transmission data is too small</td>
</tr>
<tr>
<td>805E</td>
<td>Length of the input range of the received data is too small</td>
</tr>
<tr>
<td>805F</td>
<td>Length of sorted receive data buffer is too small</td>
</tr>
<tr>
<td>8064</td>
<td>Transmit buffer write pointer has a wrong end position</td>
</tr>
<tr>
<td>8065</td>
<td>Receive buffer read pointer has a wrong end position</td>
</tr>
<tr>
<td>806F</td>
<td>Invalid configuration for open communication with T functions</td>
</tr>
<tr>
<td>8070</td>
<td>Invalid configuration for open communication with S/R functions</td>
</tr>
<tr>
<td>8071</td>
<td>Invalid configuration for S7 protocol communication</td>
</tr>
<tr>
<td>8072</td>
<td>Invalid configuration for direct PROFINET IO communication</td>
</tr>
<tr>
<td>8073</td>
<td>Invalid configuration for PROFINET IO communication with CP</td>
</tr>
<tr>
<td>8074</td>
<td>Invalid value for the communication protocol to be used</td>
</tr>
<tr>
<td>8079</td>
<td>Length of the received data does not match what is expected</td>
</tr>
<tr>
<td>807A</td>
<td>Received data has a difference in length in the header and in the receive block</td>
</tr>
<tr>
<td>807B</td>
<td>Received data has an invalid version number in the header (no compatible PCC version)</td>
</tr>
<tr>
<td>807C</td>
<td>Received data has an invalid message frame identifier (no PDI message frame received)</td>
</tr>
<tr>
<td>807D</td>
<td>Received data has an invalid identifier at the end (no complete message frame received)</td>
</tr>
<tr>
<td>8081</td>
<td>Error while copying the data from the send buffer to the output area</td>
</tr>
<tr>
<td>8082</td>
<td>Error copying data from the input area to the receive buffer</td>
</tr>
<tr>
<td>8083</td>
<td>Error while deserializing the message frame header from the received data</td>
</tr>
<tr>
<td>8084</td>
<td>Error while deserializing the PDI Basic range from the received data</td>
</tr>
<tr>
<td>8085</td>
<td>Error while deserializing the PDI LCU_STAT range from the received data</td>
</tr>
<tr>
<td>8086</td>
<td>Error while deserializing the PDI PARA range from the received data</td>
</tr>
<tr>
<td>8087</td>
<td>Error while deserializing the PDI PEC range from the received data</td>
</tr>
<tr>
<td>8088</td>
<td>Error while deserializing the PDI PLANT_STAT range from the received data</td>
</tr>
<tr>
<td>8089</td>
<td>Error while deserializing the PDI LCU_CMD range from the received data</td>
</tr>
<tr>
<td>808A</td>
<td>Error while deserializing the PDI PLANT_CMD range from the received data</td>
</tr>
<tr>
<td>808B</td>
<td>Error while deserializing the message frame header for sending</td>
</tr>
<tr>
<td>808E</td>
<td>Error while serializing the PDI Basic range into the buffer</td>
</tr>
<tr>
<td>808F</td>
<td>Error while serializing the PDI LCU_STAT range into the buffer</td>
</tr>
<tr>
<td>8090</td>
<td>Error while serializing the PDI PARA range into the buffer</td>
</tr>
<tr>
<td>8091</td>
<td>Error while serializing the PDI PEC range into the buffer</td>
</tr>
<tr>
<td>8092</td>
<td>Error while serializing the PDI PLANT_STAT range into the buffer</td>
</tr>
<tr>
<td>8093</td>
<td>Error while serializing the PDI LCU_CMD range into the buffer</td>
</tr>
<tr>
<td>8094</td>
<td>Error while serializing the PDI PLANT_CMD range into the buffer</td>
</tr>
</tbody>
</table>
“TimeoutError”
Errors resulting from a runtime error are output here (see 5.4.2). These errors are determined by transient circumstances and may occur simultaneously as a result of a configuration error.

Table 3-2

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>8101</td>
<td>Time-out when connecting to the partner station -&gt; Repeat after a pause</td>
</tr>
<tr>
<td>8102</td>
<td>Time-out when sending the data to the partner station -&gt; New connection after pause</td>
</tr>
<tr>
<td>8103</td>
<td>Time-out when disconnecting</td>
</tr>
<tr>
<td>810E</td>
<td>Undefined step number of the communication process</td>
</tr>
<tr>
<td>8119</td>
<td>No change of the received data within a period (= runtime monitoring)</td>
</tr>
</tbody>
</table>

“LastErrorConnect, LastErrorDisconnect, LastErrorSend and LastErrorReceive”
Status of the last called block for the communication. The value is deleted after a successful repetition of the operation. The evaluation of the status code can be looked up in the online help of the communication block used in “FB_PCC_SUB_CALL”.

Note
Example of “LastErrorConnect” = 80C4 with use of the open TCP communication:
→ “The connection cannot be established at the moment”

“UsedComType”
This data point is described by the “FB_PCC_SUB_CALL” block used and indicates the communication type.

Table 3-3

<table>
<thead>
<tr>
<th>Communication type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Open communication with T-blocks (OCT)</td>
</tr>
<tr>
<td>2. Open communication with send/receive blocks (OCSR)</td>
</tr>
<tr>
<td>3. S7 communication (S7C)</td>
</tr>
<tr>
<td>4. PROFINET IO communication with direct IO access (IDC)</td>
</tr>
<tr>
<td>5. PROFINET IO communication with transfer blocks for CP (IDC_CP)</td>
</tr>
</tbody>
</table>

“ComPhase”
Communication phase in the following subdivisions:

Table 3-4

<table>
<thead>
<tr>
<th>Communication phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Standstill or reset of the communication</td>
</tr>
<tr>
<td>2. Evaluation of the block interconnection</td>
</tr>
<tr>
<td>3. Editing of the transmission data</td>
</tr>
<tr>
<td>4. Transmission data review</td>
</tr>
<tr>
<td>5. Checking the request to send the data</td>
</tr>
<tr>
<td>6. Preparation for establishing a connection</td>
</tr>
</tbody>
</table>
### Communication phase

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.</td>
<td>Waiting for successful connection</td>
</tr>
<tr>
<td>8.</td>
<td>Sending of data</td>
</tr>
<tr>
<td>9.</td>
<td>Pause after sending the data</td>
</tr>
<tr>
<td>10.</td>
<td>Preparation for terminating a connection</td>
</tr>
<tr>
<td>11.</td>
<td>Waiting for successful connection</td>
</tr>
</tbody>
</table>

- **“CountConnect”**
  Counter of the attempts to establish a communication connection. If the connection is not controlled by the user program (= PCC) (OCT), this value has no meaning.

- **CountSent**
  Counter of the successfully sent message frames.

**Note**

A successfully sent message frame does not necessarily mean that it has been accepted on the other side.

- **“CountReceived”**
  Counter of the successfully received message frames.

- **“CountByteToSend”**
  Specifies the number of bytes that are sent depending on the PDI interconnection and configuration (OMAC or Weihenstephan). The send buffers and, in the case of PROFINET IO communication, the I-device range to be transferred must be adapted according to this value.

- **“CountByteToReceive”**
  Specifies the number of bytes received, depending on the PDI interconnection and configuration (OMAC or Weihenstephan). The receive buffers and, in the case of PROFINET IO communication, the I-device range to be received must be adapted according to this value.

### 3.1.3 “SUB_CALL_INTERFACE” range

In this range, all necessary information is exchanged between the generic communication block "FB_PCC_MAIN" and the protocol and hardware-specific part "FB_PCC_SUB_CALL".

To ensure error-free communication, this data must not be affected.

The identifiers are based on the identifiers of the communication modules that are primarily used. If a special communication block is required, the meaning of the connections can be looked up in the online help of the communication blocks used in "FB_PCC_SUB_CALL".
3 Blocks

3.1.4 “BUFFER_SEND_DATA” range
This buffer contained the complete transmission message frame at the time of transmission. To save storage space, the size can be adapted to the actual extent required.
The required length is calculated at runtime, depending on the configuration and wiring, and output in the diagnostics area under "DB_PCC".PCC.CONFIG_DIAG.DIAG.CountByteToSend.
You can also determine the required length yourself (see 2.6).
Example for the machine side including reserves: Array [1..450] of bytes

3.1.5 “BUFFER_TEMP_DATA” range
This buffer is needed to prepare the send data. To save storage space, the size can be adapted to that of the send data buffer "BUFFER_SEND_DATA".
Example for the machine side including reserves: Array [1..450] of bytes

3.1.6 “BUFFER_RCV_DATA” range
This buffer contains the received message frame raw data after confirmation by the receive block. Depending on the size of the PDI, the message frame can be received divided into several parts.
The required length is calculated at runtime, depending on the configuration and wiring, and output in the diagnostics area under "DB_PCC".PCC.CONFIG_DIAG.DIAG.CountByteToReceive.
You can also determine the required length yourself (see 2.6).
Example for the machine side including reserves: Array[1..100] of bytes

3.1.7 “BUFFER_RCV_SORTED” range
This buffer contains the complete receive message frame at the end of the message frame reception. To save memory space, the size of the receive data buffer "BUFFER_RCV_DATA" can be adjusted.
Example for the machine side including reserves: Array[1..100] of bytes

3.2 “FB_PCC_MAIN” block
This block is the heart of the communication. It copies the data and controls the communication for a PDI instance. The communication connection is monitored and possibly also controlled depending on the protocol.

Interconnection of the block:
- **CallCycle** (integer as input)
  For correct time processing, this input must be assigned with the call interval in milliseconds. This corresponds to the pause between 2 calls. For a cyclic call in OB1, the input must be interconnected with "OB1_PREV_CYCLE" from the temporary data area of the OB.
  For S7-1200 / 1500, this time is determined in the block itself, but this was not implemented in the S7-300 / 400 for performance reasons.
- **COM_ERROR** (binary output)
  If an error is detected during communication or parameterization, this is also output in binary form.
  For message frame repeats and temporary disconnections as restart or the
like, no communication error is reported. This error message can therefore also
be reported as an alarm to a visualization, since it is actually a longer-lasting
interruption.
No acknowledgment of this message is required.

- **EDG_NDR** (binary output)
  For each successfully received message frame, a signal for evaluating the data
  is output for the length of one program cycle.

- **CONFIG_DIAG** (structure as “InOut” link)
  Interconnection to the "UDT_PCC.CONFIG_DIAG" data structure with the
  configuration and diagnostics data

- **SUB_CALL_INTERFACE** (structure as “InOut” link)
  Interconnection to the data structure "UDT_PCC.SUB_CALL_INTERFACE" for internal data connection to "FB_PCC_SUB_CALL"

- **BUFFER_SEND_DATA, BUFFER_RCV_DATA, BUFFER_TEMP_DATA and BUFFER_RCV_SORTED** (one array as “InOut” link for each)
  Interconnection to the respective buffers of the data structure "UDT_PCC"

- **PDI_BASIC, PDI_LCU_CMD, PDI_LCU_STAT, PDI_PARA and PDI_PEC** (one variant as “InOut” link for each)
  Interconnection to the respective PDI areas.
  If a PDI area is not used or needed, it must not be connected (in STEP 7 V14 it
  must be assigned "NULL"). The interconnection must be identical for both
  communication partners. After changing the interconnection, with STEP 7 V5.5
  it may be necessary to reload or initialize the instance DB of the call block.

- **PDI_PLANT_CMD** (variant as “InOut” link)
  Any data range, which is to be transferred system-specifically from the line
  control to the machine, as a structure. For further interconnection instructions
  see 5.1.1.

- **PDI_PLANT_STAT** (variant as “InOut” link)
  Any data range, which is to be transferred system-specifically from the
  machine to the line control, as a structure. For further interconnection
  instructions see 5.1.1.

### 3.3 Block “FB_PCC_SUB_CALL”

The "FB_PCC_SUB_CALL" block is used to transfer the send and receive buffers
to the respective communication blocks. These differ depending on the protocol
and the modules used. It may therefore happen that these blocks are not identical
on the send and receive sides.

Depending on the selected variant, it may be necessary to parameterize a
communication connection which defines the connection between the line control
and the machine. See 5.2.

Depending on the STEP 7 version used, the following interconnections are
implemented directly or via the "InOut" link:

1. Range of send data from the data structure "UDT_PCC"
   .BUFFER_SEND_DATA

2. Range of receive data in the data structure "UDT_PCC"
   .BUFFER_RCV_DATA

3. Range of the internal handshake with "FB_PCC_MAIN" via the data structure
   "UDT_PCC". SUB_CALL_INTERFACE
The following variants of the block are included:

3.3.1 **Open Ethernet TCP communication with T-blocks (OCT)**

This block exchanges the PDI data with the partner over a TCP connection. The parameterization and handling of the communication connection are handled exclusively in the user program. It is thus possible to set all the necessary parameters at a local operating option. The communication can be set in operation without a programming device or software.

The T-blocks used can be used with and without parameterized connections. However, the “FB_PCC_SUB_CALL” block included with PCC only works without parameterized connections.

**Note**

For S7-300/400 and related controllers, use is only possible with the CPU-integrated Ethernet interface.

Internally, the following system blocks are called:

- **FB_TSEND** - Sending the data to the communication partner
- **FB_TRCV** - Receipt of the data from the communication partner
- **FB_TCON** - Establishing a connection
- **FB_TDISCON** - Connection termination

Necessary parameterization (see "UDT_PCC" - configuration area):

1. IPv4 address of the communication partner
2. TCP port of the communication partner
3. TCP Port of your own side

3.3.2 **Open Ethernet TCP communication with send/receive modules (OCSR)**

This block exchanges the PDI data with the partner via an Ethernet CP using a parameterized TCP connection.

**Use**

This variant was integrated for S7-300/400 and related controllers for use of an additional communications processor (CP).

Internally, the following system blocks are called (differentiating according to control series):

<table>
<thead>
<tr>
<th>Block</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC_AG_SEND/FC_AG_LSEND</td>
<td>Sending the data to the communication partner</td>
</tr>
<tr>
<td>FC_AG_RECV/FC_AG_LRECV</td>
<td>Receipt of the data from the communication partner</td>
</tr>
</tbody>
</table>

**Establishment of a TCP connection under NetPro or device configuration connections**

1. IPv4 address of the communication partner
2. TCP port of the communication partner
3 Blocks

3. TCP Port of your own side
4. Active connection always on the part of the line control

3.3.3 S7 communication (S7C)

This block exchanges the PDI data with other Siemens “S7 family” devices.

Use

This variant was implemented because of its particular reliability as well as its flexibility. (S7 communication supports connections via PN/IE, PB and MPI.) Internally, the following system blocks are called (differentiating according to control series):

Table 3-6

<table>
<thead>
<tr>
<th>Block</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BSEND/SFB_BSEND</td>
<td>Sending the data to the communication partner</td>
</tr>
<tr>
<td>FB_BRCV/SFB_BRCV</td>
<td>Receipt of the data from the communication partner</td>
</tr>
</tbody>
</table>

Establishment of an S7 connection under NetPro or device configuration> connections

1. Ipv4 address of the communication partner
2. TSAP ID of the communication partner
3. TSAP ID of your own side
4. Synchronization S7 subnet ID
5. Active connection (is always done by the line control)

3.3.4 PROFINET IO communication with direct IO access (IDC)

This block exchanges the PDI data via cyclic PROFINET IO communication. In this case, the line control of the IO controller and the machine is the I-device.

Note

All network components (switches, etc.) must support PROFINET IO.

Use

This variant was chosen because of the particularly fast communication as well as the resource-saving implementation in the user program. The configured PROFINET IO range must be in the active process image of the controller. Internally, the following system blocks are called:

Table 3-7

<table>
<thead>
<tr>
<th>Block</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFC_BLKMOV</td>
<td>Copying of the data into the process image of the controller</td>
</tr>
</tbody>
</table>
3 Blocks

Required parameter assignment

The machine must export a GSD file, which is used for integration in the line control, from the hardware configuration.

3.3.5 PROFINET IO communication with transfer blocks for CP (IDC_CP)

This block exchanges the PDI data with a CP, which in turn exchanges it with the partner via cyclic PROFINET IO communication. In this case, the line control of the IO controller and the machine is the I-device.

Note

All network components (switches, etc.) must support PROFINET IO.

Use

This variant was integrated for S7-300 for the use of an additional communication processor (CP) 343-1 Lean.

Internally, the following system blocks are called:

Table 3-8

<table>
<thead>
<tr>
<th>Block</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC_PNIO_SEND</td>
<td>Transfer of the data to the CP</td>
</tr>
<tr>
<td>Transfer of the data to the CP</td>
<td>Receipt of the data from the CP</td>
</tr>
</tbody>
</table>

Required parameter assignment

The machine must export a GSD file, which is used for integration in the line control, from the hardware configuration.

3.4 “FB_PCC_MAIN_CALL” block

This block is used as a template for the configuration and multi-instance call of the PCC program components for the transmission of a PDI interface. It must be called more often by the line control according to the number of PDI instances.

It can be called directly in the cyclic program (OB1) and contains a commented sample for the parameterization. For the respective use, unnecessary parameters (e.g. parameters of other protocols) can be marked as a comment or deleted.

Connection Description for „COM_ERROR“, „EDG_NDR“ as well as „CallCycle“ reference „FB_PCC_MAIN“. 
The block is structured by regions into:

**Configuration (copy template corresponds to OCT communication)**

- **Control**
  Describes the data points for controlling the communication.

- **Use**
  Determines the basic usage. The template is used by the machine with OMAC V2 PDI interface.

- **Parameters of connection-based protocols (OCT, OCSR and S7C)**
  Parameters for the timing of the transmission, as well as the connection identifier.

- **“Open Ethernet TCP communication with T-blocks (OCT)” parameters**
  - Specification of the partner Ethernet IPv4 address
  - Specification of the local and remote TCP port number
  - Setting the hardware interface In STEP 7 TIA, the system variable for the symbolic hardware identifier can also be used here.

- **“Open communication with send/receive blocks (OCSR)” parameter**
  Specification of the hardware address of the communication processor (CP)

- **Parameter “S7 communication (S7C)”**
  S7 protocol-typical parameters for the block handshake.

- **“PROFINET IO communication with direct IO access (IDC)” parameter**
  - STEP 7 V5.5: Periphery start addresses for input and output range set.
  - STEP 7 V14: Symbolic transfer of the range at the "FB_PCC_SUB_CALL" call.

- **“PROFINET IO communication with transfer blocks for CP (IDC_CP)” parameter**
  Specification of the hardware address of the communication processor (CP) that serves as an I device.

**Calling communication functions**

Here, "FB_PCC_MAIN" and "FB_PCC_SUB_CALL" are called as a multi-instance.

- The interconnection of the PDI interface area must be adapted to the usage.

- For STEP 7 V14, the periphery area for the PROFINET IO input/output areas is symbolically transferred at the "FB_PCC_SUB_CALL" call.
4 Application in the project:

As an alternative to using the blocks of the respective library, program parts from the configuration sample can also be copied.

Plant data Interface (PDI Library)
1. Open the library or integrate it into the project (see online documentation of the respective STEP 7 version).
2. Depending on the standard used, you transfer the required PDI groups "BASIC", "LCU", "PARA" and "PEC" as data types (UDT) to the project library or project.
3. Transfer the plant-specific data type "UDT_PDI" from the template to the project and adapt the PDI group usage and the plant-specific area according to the current project.
4. Integrate the "UDT_PDI" in a data block or in this case transfer the copy template "DB_PDI" to the project.
5. Optionally, the observation template "VAT_PDI" can also be transferred to the project and adapted to its use.

Plant communication (PCC Library)
1. Open the library or integrate it into the project (see online documentation of the respective STEP 7 version).
2. Depending on the requirements, configure a connection or the hardware for the protocol used (see the documentation for the individual protocols in chapter 3.3).
3. Transfer the data type "UDT_PCC" to the project library or project.
4. Apply the protocol as well as the hardware-specific "FB_PCC_SUB_CALL" to the project. (Optionally also available as AWL source in STEP 7 V5.5.)
5. Apply the hardware-specific block "FB_PCC_MAIN" to the project.
6. Integrate the "UDT_PCC" in a data block or in this case transfer the finished copy template "DB_PCC" to the project.
7. Create the call block in which the two blocks "FB_PCC_MAIN" and "FB_PCC_SUB_CALL" are called. Alternatively, a finished copy template "FB_PCC_MAIN_CALL" can be added to the project. (Optionally also available as AWL source in STEP 7 V5.5.)
8. Make sure that the configuration area in the DB (UDT_PCC) is preassigned as required or is described during startup or cyclically.
9. Make sure that the project-specific use of the PDI interface also corresponds to the interconnection on the "FB_PCC_MAIN" call.
10. Optionally, the observation template "VAT_PCC" can also be transferred to the project and adapted to its use.
5 Configuration sample

5.1.1 Data interface (PDI)

Just like in a real project, first of all the PDI usage must be defined. The basis for this is the customer request, which represents the information content.

The structure can also differentiate for each machine. The plant-specific part should be defined by means of a UDT, which is integrated into the respective projects.

<table>
<thead>
<tr>
<th>Library</th>
<th>Length (bytes)</th>
<th>Project</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M to LC</td>
<td>LC to M</td>
<td>M to LC</td>
</tr>
<tr>
<td>WS_V2_BASIC</td>
<td>88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS_V2_LCU</td>
<td>44</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>VS_V2_PARA</td>
<td>140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS_V2_PEC</td>
<td>124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OMAC_V2_Basic</td>
<td>88</td>
<td>OMAC_Basic</td>
<td>88</td>
</tr>
<tr>
<td>OMAC_V2_LCU</td>
<td>40</td>
<td>18</td>
<td>40</td>
</tr>
<tr>
<td>OMAC_V2_PARA</td>
<td>142</td>
<td>OMAC_PEC</td>
<td>124</td>
</tr>
<tr>
<td>OMAC_V2_PEC</td>
<td>124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLANT_SPEC</td>
<td>?</td>
<td>PLANT_SPEC</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Longer user data</td>
<td></td>
<td>256</td>
<td>22</td>
</tr>
<tr>
<td>Message frame length</td>
<td></td>
<td>265</td>
<td>31</td>
</tr>
</tbody>
</table>

5.1.2 Communication (PCC)

The following information must be exchanged to set up the communication:

1. Plant-specific network (example: Usable areas, address of the machine, NTP server, remote maintenance addresses, ...)
   - When using open TCP communication (OCT and OCSR):
     a. Ipv4 address of the partner
     b. TCP port of machine and line control
   - When using the S7 communication (S7C):
     a. Ethernet Subnet ID
     b. TSAP port of machine and line control
     c. Message frame Identification (ID) of the blocks for the communication from machine to line control and vice versa
   - When using PROFINET IO communication (IDC and IDC_CP):
     a. Unique assignment of PROFINET names
     b. Transfer of the GSD file from the machine manufacturer to the line integrator
### Plant Communication Concept V3

**Entry ID:** 98278624, V1.0, 02/2018

![Image](image_url)

© Siemens AG 2018 All rights reserved

#### Key:

Colored information must be agreed on between the communication partners.

<table>
<thead>
<tr>
<th>Station</th>
<th>Step</th>
<th>Hardware Interface</th>
<th>Comm.ID Hex (VAM64...)</th>
<th>IPv4 Address</th>
<th>Port</th>
<th>TSNAP</th>
<th>HW ID / IADDR / Device ID</th>
<th>PROFINET Name</th>
<th>GUID File</th>
<th>IN Address Range</th>
<th>OUT Address Range</th>
<th>Ethernet SubnetID</th>
<th>Com ID CC to M</th>
<th>Com ID M to CC</th>
<th>IPv4 Address</th>
<th>Comm.ID Hex (VAM64...)</th>
<th>Port</th>
<th>TSNAP</th>
<th>HW ID / IADDR / Device ID</th>
<th>IN Address Range</th>
<th>OUT Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.013</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M2.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.014</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M3.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.015</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M4.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.016</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M5.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.017</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M6.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.018</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M7.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.019</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M8.5</td>
<td>7</td>
<td>CPU</td>
<td>500</td>
<td>192.368.004.020</td>
<td>2100</td>
<td>64</td>
<td>PROFINET Name</td>
<td>GUID File</td>
<td>IN Address Range</td>
<td>OUT Address Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Machine side**

- **Station:** MTP Server
- **Step:** 7
- **Hardware Interface:** CPU
- **Comm.ID Hex (VAM64...):** PROFINET Name
- **IPv4 Address:** GUID File
- **Port:** IN Address Range
- **TSNAP:** OUT Address Range
- **HW ID / IADDR / Device ID:** PROFINET Name
- **PROFINET Name:** GUID File
- **GUID File:** IN Address Range
- **IN Address Range:** PROFINET Name
- **OUT Address Range:** GUID File
- **Ethernet SubnetID:** MTP Server
- **Com ID CC to M:** PROFINET Name
- **Com ID M to CC:** GUID File
- **IPv4 Address:** PROFINET Name
- **Comm.ID Hex (VAM64...):** GUID File
- **Port:** PROFINET Name
- **TSNAP:** GUID File
- **HW ID / IADDR / Device ID:** PROFINET Name
- **IN Address Range:** GUID File
- **OUT Address Range:** PROFINET Name

---

**Controller side**

- **Station:** PROFINET Name
- **Step:** 7
- **Hardware Interface:** CPU
- **Comm.ID Hex (VAM64...):** PROFINET Name
- **IPv4 Address:** GUID File
- **Port:** IN Address Range
- **TSNAP:** OUT Address Range
- **HW ID / IADDR / Device ID:** PROFINET Name
- **PROFINET Name:** GUID File
- **GUID File:** IN Address Range
- **IN Address Range:** PROFINET Name
- **OUT Address Range:** GUID File
- **Ethernet SubnetID:** MTP Server
- **Com ID CC to M:** PROFINET Name
- **Com ID M to CC:** GUID File
- **IPv4 Address:** PROFINET Name
- **Comm.ID Hex (VAM64...):** GUID File
- **Port:** PROFINET Name
- **TSNAP:** GUID File
- **HW ID / IADDR / Device ID:** PROFINET Name
- **IN Address Range:** GUID File
- **OUT Address Range:** PROFINET Name

---

<table>
<thead>
<tr>
<th>Station</th>
<th>Step</th>
<th>Hardware Interface</th>
<th>Comm.ID Hex (VAM64...)</th>
<th>IPv4 Address</th>
<th>Port</th>
<th>TSNAP</th>
<th>HW ID / IADDR / Device ID</th>
<th>PROFINET Name</th>
<th>GUID File</th>
<th>IN Address Range</th>
<th>OUT Address Range</th>
<th>Ethernet SubnetID</th>
<th>Com ID CC to M</th>
<th>Com ID M to CC</th>
<th>IPv4 Address</th>
<th>Comm.ID Hex (VAM64...)</th>
<th>Port</th>
<th>TSNAP</th>
<th>HW ID / IADDR / Device ID</th>
<th>IN Address Range</th>
<th>OUT Address Range</th>
</tr>
</thead>
</table>
5.1.3 Parameter assignment

As part of the application sample, you download two projects for STEP 7 V5.5 or STEP 7 (TIA Portal) V14.

The CPUs are configured in three groups:

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPU family</th>
<th>Configuring</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-18</td>
<td>S7-1200/1500</td>
<td>STEP 7 (TIA Portal) V14</td>
</tr>
<tr>
<td>21-28</td>
<td>S7-300/400</td>
<td>STEP 7 (TIA Portal) V14</td>
</tr>
<tr>
<td>31-38</td>
<td>S7-300/400</td>
<td>STEP 7 V5.5:</td>
</tr>
</tbody>
</table>

**STEP 7 (TIA Portal)-Project**

Table 5-3

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPU</th>
<th>Interface</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>S7-1200</td>
<td>PROFINET (integrated)</td>
<td>TCP with T-blocks (OCT)</td>
</tr>
<tr>
<td>12</td>
<td>S7-1200</td>
<td>PROFINET (integrated)</td>
<td>I-device with direct IO access (IDC)</td>
</tr>
<tr>
<td>13</td>
<td>S7-1500</td>
<td>PROFINET (integrated)</td>
<td>TCP with T-blocks (OCT)</td>
</tr>
<tr>
<td>14</td>
<td>S7-1500</td>
<td>CP</td>
<td>S7 communication (S7C) and parameterized connection²</td>
</tr>
<tr>
<td>15</td>
<td>S7-1500</td>
<td>PROFINET (integrated)</td>
<td>I-device with direct IO access (IDC)</td>
</tr>
<tr>
<td>16</td>
<td>S7-1200</td>
<td>PROFINET (integrated)</td>
<td>Resource-optimized I-device with direct IO access (SIDC)³</td>
</tr>
<tr>
<td>17</td>
<td>S7-1500</td>
<td>CM</td>
<td>TCP with T-blocks (OCT)</td>
</tr>
<tr>
<td>18</td>
<td>S7-1500</td>
<td>CP</td>
<td>TCP with T-blocks (OCT)</td>
</tr>
<tr>
<td>21</td>
<td>S7-300</td>
<td>PROFINET (integrated)</td>
<td>TCP with T-blocks (OCT)</td>
</tr>
<tr>
<td>22</td>
<td>S7-300</td>
<td>CP343-1 Lean</td>
<td>TCP communication with send/receive modules (OCSR) and parameterized connection</td>
</tr>
<tr>
<td>23</td>
<td>S7-300</td>
<td>PROFINET (integrated)</td>
<td>S7 communication (S7C) and parameterized connection</td>
</tr>
<tr>
<td>24</td>
<td>S7-300</td>
<td>PROFINET (integrated)</td>
<td>I-device with direct IO access (IDC)</td>
</tr>
<tr>
<td>25</td>
<td>S7-300</td>
<td>CP343-1 Lean</td>
<td>I-device with transfer blocks for CP (IDC_CP)</td>
</tr>
<tr>
<td>26</td>
<td>S7-400</td>
<td>CP</td>
<td>TCP communication with send/receive modules (OCSR) and parameterized connection</td>
</tr>
<tr>
<td>27</td>
<td>S7-400</td>
<td>CP</td>
<td>S7 communication (S7C) and parameterized connection</td>
</tr>
<tr>
<td>28</td>
<td>S7-300</td>
<td>PROFINET (integrated)</td>
<td>Resource-optimized I-device with direct IO access (SIDC)⁴</td>
</tr>
</tbody>
</table>

¹ CP = communication processor, CM = communication module

² Automatic determination of TSAP addresses using SIMATIC-ACC (SIMATIC Application Controlled Communication)

³ Uses optimized blocks, which are only available for I-device communication on the machine side

⁴ Uses optimized blocks, which are only available for I-device communication on the machine side
### 5.2 Configuration of the connections

#### 5.2.1 Creating a TCP connection

**Requirements**

- Configured modules (e.g.: Rack, CPU, CP,…)
- Configuration of the Ethernet interface (e.g.: Subnet, IP address, subnet mask, router, …)

---

5. CP = communication processor, CM = communication module

6. Uses optimized blocks, which are only available for I-device communication on the machine side
Creating a TCP connection with STEP 7 V5.5

1. Open "NetPro" by selecting the network and "Open object (Ctrl + Alt + O)".
2. Select the CPU module of the station.
3. In the context menu, select "Insert new connection (Ctrl-N)"

Figure 5-1

4. In the dialog that opens, select "unspecified" if the control program of the partner controller is in another project (1).
5. Set the type of connection to "TCP connection" (2) and confirm with OK.
6. Enter a connection name (1) and select a free ID.
7. Put a checkmark next to “Check active connection setup” (2).
8. Note the connection ID and LADDR (3). These must be specified later in the user program.

9. Enter the IPv4 address of the communication partner in the "Addresses" tab (1).
10. Enter your own (local) TCP port (2).
11. Enter the partner (remote) TCP port (3).

Creating a TCP connection with STEP 7 V14
1. Open "Devices & Networks" by double-clicking on the entry in the project tree (1).

---

7 For PCC, the endpoint of the line control is always the active participant
2. Select the representation of the “connections” (2).
3. Select the CPU module of the station (3).
4. Select “Add new connection” from the context menu (4).

5. Select “TCP Connection” as the connection type (1).
6. Select “unspecified” if the control program of the partner controller is located in another project (2).
7. Select a free connection ID (3).
8. Put a checkmark next to Check active connection setup (4).
9. Confirm the creation of the new connection.

For PCC, the endpoint of the line control is always the active participant.
5.2.2 Create S7 connection

Requirements
- Configured modules (e.g.: Rack, CPU, CP, ...)
- Configuration of the Ethernet interface (e.g.: Subnet, IP address, subnet mask, router, ...)

Creating an S7 connection with STEP 7 V5.5
1. Open "NetPro" by selecting the network and "Open object (Ctrl + Alt + O)".
2. Select the CPU module of the station.
3. Select "Insert new connection (Ctrl-N)" from the context menu.

Figure 5-9

4. In the dialog that opens, select "unspecified" if the control program of the partner controller is in another project (1).

5. Set the type of connection to "S7 connection" (2) and confirm with OK.

Figure 5-10

6. Put a checkmark next to "Check active connection setup" (1).  
7. Enter a free connection ID (2).  
8. Enter the Ipv4 address of the partner (3).

9 For PCC, the endpoint of the line control is always the active participant.
9. Open the address-detail dialog (4).

Figure 5-11

10. Determine the TSAP (Transport Service Access Point) ID by specifying free connection resources and the rack and slot of the CPU module (1). For S7 connections, the network subnet ID (Fig. 5-12, 1) must be set identically in the project of both communication partners. This can be controlled in the dialog of the connection or set in the properties of the IE network.

Fig. 5-12

Creating an S7 connection with STEP 7 V14

1. Open "Devices & Networks" by "double-clicking" on the entry in the project tree.
2. Select the representation of the “connections” (1).
3. Select the CPU module of the station (2).
4. In the shortcut menu, select “Add new connection”.
5. Select “S7 Connection” as the connection type (3).
6. Select “unspecified” if the control program of the partner controller is located in another project (4).
7. Select a free connection ID (5).
8. Confirm the creation of the new connection.

9. Enter a connection name of your choice (1).
10. Enter the Ipv4 address of the partner (2).
5.3 Hardware configuration I-device

Handling

The procedure described assumes that the I-device and the IO controller are located in different automation projects, as is often the case with PCC. If both are in the same project, then no export/import is necessary, but only a corresponding creation of the transfer areas.

Requirements

- Configured modules (e.g.: Rack, CPU, CP,...)
- Configuration of the PROFINET interface (e.g.: Subnet, IP address, subnet mask, router, ...)

10 For PCC, the endpoint of the line control is always the active participant
Configuration I-device with STEP 7 V5.5

1. Open "HW Config" by selecting "Station>" Hardware and "Open Object (Ctrl + Alt + O)".

Fig. 5-17

2. Assign a unique device name in the properties of the PROFINET interface (1). This name identifies the device on the network by the controller.

Figure 5-18

3. Put a checkmark next to next to the I-device function (1).
4. In the table of transfer areas, enter the data lengths to be transmitted for sending and receiving (2). The required lengths are calculated at runtime depending on the configuration and interconnection and output in the diagnostics area:

"DB_PCC".PCC.CONFIG_DIAG.DIAG.CountByteToSend = Length of the output data
"DB_PCC".PCC.CONFIG_DIAG.DIAG.CountByteToReceive = Length of the input data

For certain modules, the length is limited for each individual entry. In this case, 2 consecutive entries with the corresponding total length must be created.

You can also determine the lengths yourself (see 2.6).

**Note**

If you work with direct IO access, the transfer area must be in the process image of the controller. Depending on the CPU type, this can be controlled or adjusted in the properties of the CPU under “Cycle/size of process image”.

5. In the hardware configurator, under the menu item "Extras"> "Create GSD file for I-device ..." (1) create a GSD file of the station and export it. The GSD file is imported in the line control project and contains all necessary information for communication with the I-device.

**Configuration I-device with STEP 7 V14**

1. Open “Device configuration” by “double-clicking” on the entry in the project tree.
2. Assign a unique device name in the properties of the PROFINET interface (1). This name identifies the device on the network by the controller.
3. Mark the interface (1) and then under "Properties" select the menu item "Operating mode" (2). Put a checkmark next to next to the I-device function (3).

4. In the table of transfer areas, enter the data lengths to be transmitted for sending and receiving (4). The required lengths are calculated at runtime depending on the configuration and interconnection and output in the diagnostics area:
   
   "DB_PCC".PCC.CONFIG_DIAG.DIAG.CountByteToSend = Length of the output data
   
   "DB_PCC".PCC.CONFIG_DIAG.DIAG.CountByteToReceive = Length of the input data

   For certain modules, the length is limited for each individual entry. In this case, 2 consecutive entries with the corresponding total length must be created.

   You can also determine the lengths yourself (see 2.6).

**Note**

If you work with direct IO access, the transfer area must be in the process image of the controller. Depending on the CPU type, this can be controlled or adjusted in the properties of the CPU under "Cycle/size of process image".

5. Under "Export device description (GSD)" (point 5), export the GSD file.

The GSD file is imported in the line control project and contains all necessary information for communication with the I-device.

### Configuration IO controller with STEP 7 V14

1. Open “Device configuration” by “double-clicking” on the entry in the project tree.

   **Figure 5-22**

   2. Import the I-device GSD files into the automation project under the menu item "Tools"> "Manage device description files (GSD)" (1).
Figure 5-23

3. In the catalog under "Other field devices"> "PROFINET IO"> "PLCs & CPs", select the previously imported I-device (1) and drag it to the corresponding network in your project.

4. Determine the desired IO controller by selecting the "Link" of the added station (2).

Figure 5-24

5. Now check the transfer area after selecting the I-device (1) in the "Network view" tab with the "Network" view in the "Network overview" tab (2). The assignment to the I/O area of the IO controller can now be adapted directly in the table (3) or in the properties (4).

The I/O areas set for the individual stations must be specified later in the user program.
5.4 Error messages used

In general, all errors are not evaluated in every library version (STEP 7 V5.5 or V14). This list only contains the sum of all error codes.

5.4.1 Configuration error

Invalid value for FB parameter "PDI_TYPE" (8028)
The value of the configuration parameter "PDI_Type" could not be assigned a corresponding type.

Interconnection to <Range> is not a valid DB link (8032-803C)
Only interconnections to data blocks with a byte length> 0 are permitted.

Length <range> is invalid DB-Link (8044-8049)
When specifying dynamic arrays, only positive ranges with a length> 0 are allowed.

Interconnection to <range> has an invalid data length (8050-8059)
Depending on the type and version (parameter "PDI_Type"), lengths of the individual areas are stored. If structures with wrong lengths are interconnected, the corresponding error message appears.

The length of the <range> is too short (805A-805F)
If the length of the interconnected dynamic array is too short, the corresponding error message appears. The required length is determined by the interconnection of the PDI areas.

Transmit buffer write pointer has a wrong end position (8064)
After the formation of the transmission data, the total length does not give the expected value. This can occur due to incorrect connection of the PDI data or due to an error in the "serialization" of optimized data areas.

Receive buffer read pointer has a wrong end position (8065)
After evaluation of the received data, the total length does not give the expected value. This can occur due to incorrect connection of the PDI data or due to an error in the "deserialization" in optimized data areas.

Invalid configuration for <protocol> (806F-8073)
Implausible parameters have been detected for the respective protocol ("ComType" described in "FB_PCC_SUB_CALL")

Invalid value for the communication protocol to be used (8074)
The value of the "ComType" parameter (described by "FB_PCC_SUB_CALL") could not be assigned to a corresponding communication protocol.

Length of the received data does not match what is expected (8079)
The sum of the received message frame segments does not correspond to the expected length. The length of the expected data is calculated from the sum of the interconnected PDI structures. The interconnection of the PDI structures must be identical on both endpoints of the communication!
This error can also occur sporadically if individual message frame segments are missing. The cause can also be an extremely slow or unstable communication.
5 Configuration sample

Received data has a difference in length in the header and in the receive block (807A)

  The sum of the received message frame segments does not correspond to the length of the transmitted data described in the header. The cause may be a faulty use or interconnection of the receive blocks in the “FB_PCC_SUB_CALL” block. This error can also occur sporadically if individual message frame segments are missing. The cause can also be an extremely slow or unstable communication.

Received data has an invalid version number in the header (no compatible PCC version) (807B)

  The PCC blocks check the version of the protocol used by transferring an identifier in the header. Both communication endpoints must use identical versions of the blocks.

Received data has an invalid message frame identifier (no PDI message frame received) (807C)

  The received data does not correspond to the message frame identifier valid for PCC. The cause can be, inter alia, multiple connections. Check "ConnectionId" for uniqueness!

Received data has an invalid identifier at the end (no complete message frame received) (807D)

  The received data is missing the "end" identifier of the message frame. This can occur with PROFINET IO communication if the data lengths parameterized in the hardware configuration do not correspond to the complete data volume.

Error copying data from <range> to <range> (8081-8082)

  The copy function has reported an error. This can occur, among other things, in read-only data areas.

Error deserializing the <range> from the received data (8083-808A)

  The function for deserializing the data has reported an error. This can occur, for example, in read-only data areas as well as with invalid PDI interconnection.

Error serializing the <range> in the buffer (808E-8094)

  The function for serializing the data has reported an error. This can occur, for example, with invalid PDI interconnection.

5.4.2 Runtime error

Time-out when connecting to the partner station -> Repeat after a pause (8101)

  The connection to the remote station took longer than expected. After a short break, the connection is restarted.

Time-out when sending the data to the partner station -> New connection after a pause (8102)

  Sending the data to the remote station took longer than expected. After a short pause, a connection is restarted.

Time-out when disconnecting (8103)

  The connection to the remote station could not be successfully terminated. The disconnection is aborted and, if necessary, a new connection is started.
5 Configuration sample

Undefined step number of the communication process (810E)

An internal error has occurred during the communication process. A connection is restarted.

No change of the received data within a period (= runtime monitoring) (8119)

No new data was received for a set period of time. This message is the result of all communications problems and the basis for monitoring the successful transmission.
6 Appendix

6.1 Service and Support

Industry Online Support
Do you have any questions or need assistance?
Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.
The Industry Online Support is the central address for information about our products, solutions and services.
Product information, manuals, downloads, FAQs, and application examples – all the information you need is accessible with just a few mouse clicks at:
https://support.industry.siemens.com

Technical Support
The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.
You send queries to Technical Support via Web form:
https://support.industry.siemens.com/My/ww/en/requests

Service offer
Our range of services includes, inter alia, the following:
- Product trainings
- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts
You can find detailed information on our range of services in the service catalog:
https://support.industry.siemens.com/cs/sc

Industry Online Support app
You will receive optimum support wherever you are with the “Siemens Industry Online Support” app. The app is available for Apple iOS, Android and Windows Phone:
https://support.industry.siemens.com/cs/ww/en/sc/2067
6.2 Links and Literature

Table 6-1

<table>
<thead>
<tr>
<th>No.</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Siemens Industry Online Support</td>
</tr>
<tr>
<td></td>
<td><a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Entry page of the application sample</td>
</tr>
<tr>
<td>3</td>
<td>Which &quot;local_device_id&quot; do you parameterize in order to establish a connection with FB65 &quot;TCON&quot; for Open User Communication (OUC) via Industrial Ethernet?</td>
</tr>
</tbody>
</table>

6.3 Change documentation

Table 6-2

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>02/2018</td>
<td>First version</td>
</tr>
</tbody>
</table>