# Documentation

# Profinet connection
# S7-1200/1500 ↔ SE-7xx

**fixed data mapping**

**via Hilscher NetJack 100 (IO device)**
**using the S7 Profinet interface**


**Documentation: 11th May 2021**
**valid for: Version 1.1**

Author: Lukas Jolbej

In this documentation the S7 Profinet connection (fixed data mapping) to the Stange SE-7xx device with Hilscher NetJack 100 module is explained.

Used devices:

- Stange SE-702
- Siemens S7-1212C DC/DC/DC (6ES7212-1AE40-0XB0) as IO Controller
- Hilscher NetJack 100 (NJ 100DN-RE/PNS) as IO Device

connected via a 100 MBit/s switch

Used software:

- Siemens TIA Portal V15
- Windows 7 SP1
- Device version 7.0.2.10 for the SE-702
- Firmware version 4.1 for the S7-1212C
- Firmware version 3.12.0.2 for the Hilscher NetJack 100

Requirements:

- The feature must be licensed in the SE-7xx
- The Profinet name of the SE-7xx (Hilscher NetJack 100) and the IP addresses for both networking interfaces must be known
- SE-7xx (Hilscher NetJack 100) and S7 are located in the same network subnet

corresponding TIA Portal templates:

- se7xx-1200-1500-scl-pn (template project), version 1.1
- se7xx-1200-1500-scl-pn-library (library), version 1.1

**Please note that, depending on the S7 firmware version, a current version of TIA Portal may be required.**
**The template project and the library were created with TIA Portal V15,**
**but can be upgraded to a current version if required.**

# Table of contents

## Getting started

### Function overview

#### Functionality

The S7 Profinet interface of the SE-7xx series via a Hilscher NetJack 100 is a licensable extension to the two Modbus interfaces at port 502 and 21303 (➔ S7 Modbus interface). These interfaces allow read and write access to data in the SE-7xx via Modbus TCP or Profinet. Differences to the Modbus interface at port 502 are in the amount of data as well as the data preparation (S7-optimized byte order). Therefore, the blocks can only be used for the S7-Profinet interface. Not all data available at port 502 are also available via Profinet.

Basically, the behavior of the SE-7xx changes by switching on the S7 Profinet interface in such a way that the digital control signals of the S7 have priority over corresponding entries in the PLC instruction list of the SE-7xx. Therefore, the entire digital logic must be implemented in the S7. All data areas are written or overwritten in the SE-7xx.
For the actual values, please note that values of the CAN IO (SIOS or CAN base) always have priority.

The S7 Profinet interface (variant with fixed data mapping) is identical to the S7 Modbus interface in terms of data scope and data structure. In both cases the enlarged data quantities (from device version 7.0.3.x onwards) are not supported.
The S7 Profinet interface and the S7 Modbus interface (port 21303) must not be activated at the same time, because they use the same data area to write and they would block each other. Both interfaces each require a license.

➔ see document: **SE-7xx – S7 interface data structure.pdf**

#### What are the features?

- Control and query programmer
- Control and query control zone
- Query setvalue and its status
- Control and query alarm handler
- Generate alarm and query alarm status
- Control and query datalogger
- Query process step status
- Query digital track status
- Query tolerance status, enable tolerance (if configured as external activatable)
- Query limit status
- Set digital variable in SE-7xx (FI 2000-2199)
- Query digital variable from SE-7xx (FO 2000-2199)
- Set analog variable in SE-7xx (values 41-80)
- Query analog variable from SE-7xx (values 1-40)
- Set actual value in SE-7xx, force Overflow/Underflow/Break status
- Query actual value from SE-7xx, query error status of actual value

#### What data is transferred from the SE-7xx to the S7 (status data)?

- Boolean data ➔ Programmer status, Control zone status, Setvalue status, Actual value status, Tolerance status, Limit status, Alarm status, Alarm handler status, Process step status, Digital track status, Digital output variables (FO 2000-2199), Datalogger status
- 32 Bit floating-point values (REAL) ➔ Control zone outputs (Y values), Setvalues, Analog variables 1-40, Actual values

#### What data is transferred from the S7 to the SE-7xx (control data)?

- Boolean data ➔ Programmer control, Control zone control, Tolerance enabling, Digital input variables (FI 2000-2199), Alarm handler control, Alarm inputs, Datalogger control
- 32 Bit floating-point values (REAL) ➔ Analog variables 41-80, Actual values

### Check licensing status

The correct licensing status of the S7 Profinet connection can be checked in the SE-7xx.
***Configuration*** > ***Hardware Test*** > ***License Information*** > ***Profinet IO-Device*** will show the current status.
In case of a missing license this entry shows “No” and a license alarm will be activated.

### Activating the S7 Profinet interface in the SE-7xx

The S7 Profinet interface must be enabled in the SE-7xx device. This takes place under ***Configuration*** > ***Hardware*** > ***Hardware options***. Change the setting ***Hardware module*** to ***Profinet IO-Device***. Now the submenu ***Profinet IO-Device*** can be opened. There you can select ***PARAM.*** on the left side and configure the ***Profinet name***, by default ***nj100repns***. The setting ***Data mapping*** must be set to ***Predefined data blocks***.

The "variable mapping" is another operation mode of the S7 Profinet interface and is described in a separate documentation.

Only a 1:1 data transfer is possible. This means that an SE-7xx cannot be connected to several S7s and exchange data. Likewise, an S7 can only exchange data with one SE-7xx.[1]

### Limitations for configuring the Profinet Device Name

Please consider the rules of the Profinet naming convention. The following rules must be respected when configuring the Profinet device name so that the communication can take place: (Source: https://support.industry.siemens.com)

- Limitation to a total length of 127 characters (letters “a” to “z”, digits “0” to “9”, hyphen or dot)
- A name component within the device name, i.e. a string between two dots, must not be longer than 63 characters
- No special characters like umlauts, parantheses, underscores, slashes, blank spaces
  The hyphen is the only allowed special character
- The device name may not contain any capital letters
- The device name may not start or end with the characters “-“ or “.”
- The device name may not start with digits
- The device name may not have the format n.n.n.n (n = 0…999)
- The device name may not start with the string “port-xyz-“ (x, y, z = 0…9)

### Datalogger configuration (PLC statement list)

For proper functionality of the datalogger when enabling the S7 Profinet interface, the following two lines must be present in the SE-7xx PLC statement list:

```
L FO 768
R FO 1311
```

The PLC statement list can be found at ***Configuration*** > ***Functions*** > ***PLC statement list***. After adding those two lines apply the changes by selecting ***Apply*** (***Take Over***) and then save the changes with ***Back***.

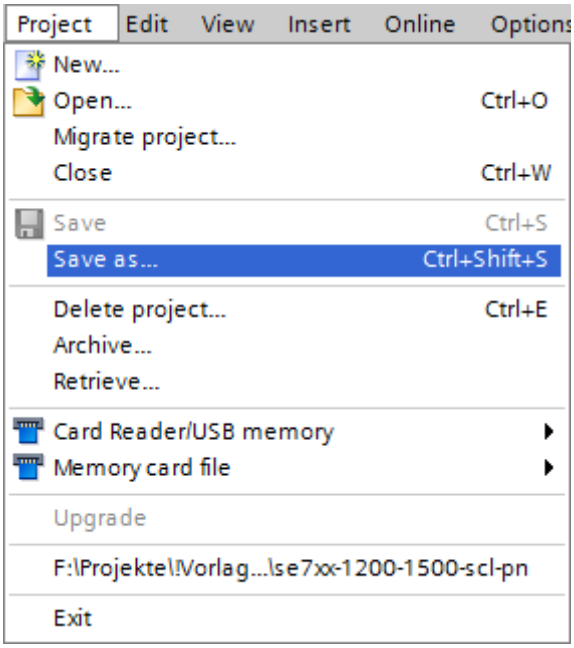
For the basic operation of the S7 Profinet interface only these two lines are required. The factory lines can therefore be deleted, provided that no customer or plant-specific configuration has been carried out by the plant manufacturer or by Stange.
If such a configuration has been carried out, the existing instruction lines must not be deleted. It must then only be ensured that the two lines mentioned above are present.

For more information on configuring the datalogger see the corresponding documentation.

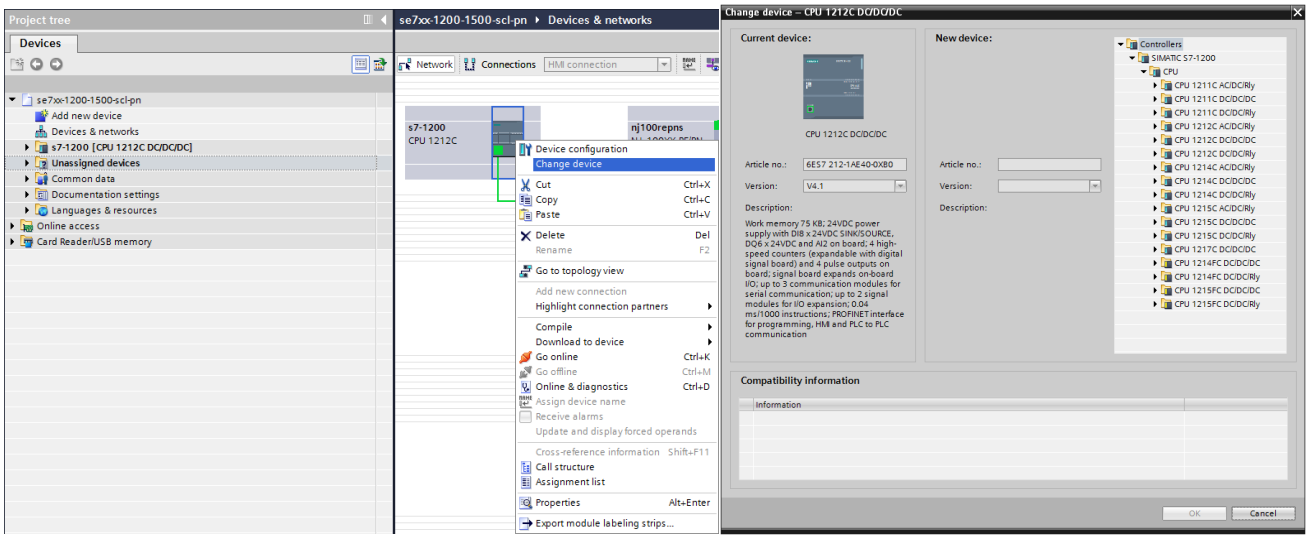
[1] It would be technically possible to create a copy of the blocks and adapt them so that a second SE-7xx can also be controlled. However, this is not supported by Stange and is therefore not discussed in detail here. For example, it should also be noted that the peripheral address range for the modules is limited for the S7-1200.

## Using the project as a template

The project ***se7xx-1200-1500-scl-pn*** can be used as a template. It is loaded into TIA Portal and can be saved directly via ***Project > Save as*** as a copy with a new name. This enables to use the template again.




In the template project, there are projected one S7-1212C DC/DC/DC and one S7-1513-1 PN, respectively. Both contain the same modules. The project must be adapted if another PLC than S7-1212C DC/DC/DC or S7-1513-1 PN is used: under ***Devices & networks*** the S7 must be selected by right mouse click and ***Change device*** opens the Change device dialog. Now the used device can be selected. The S7 not in use can be deleted from the project by right mouse click.



### Configuring the IP addresses in TIA Portal

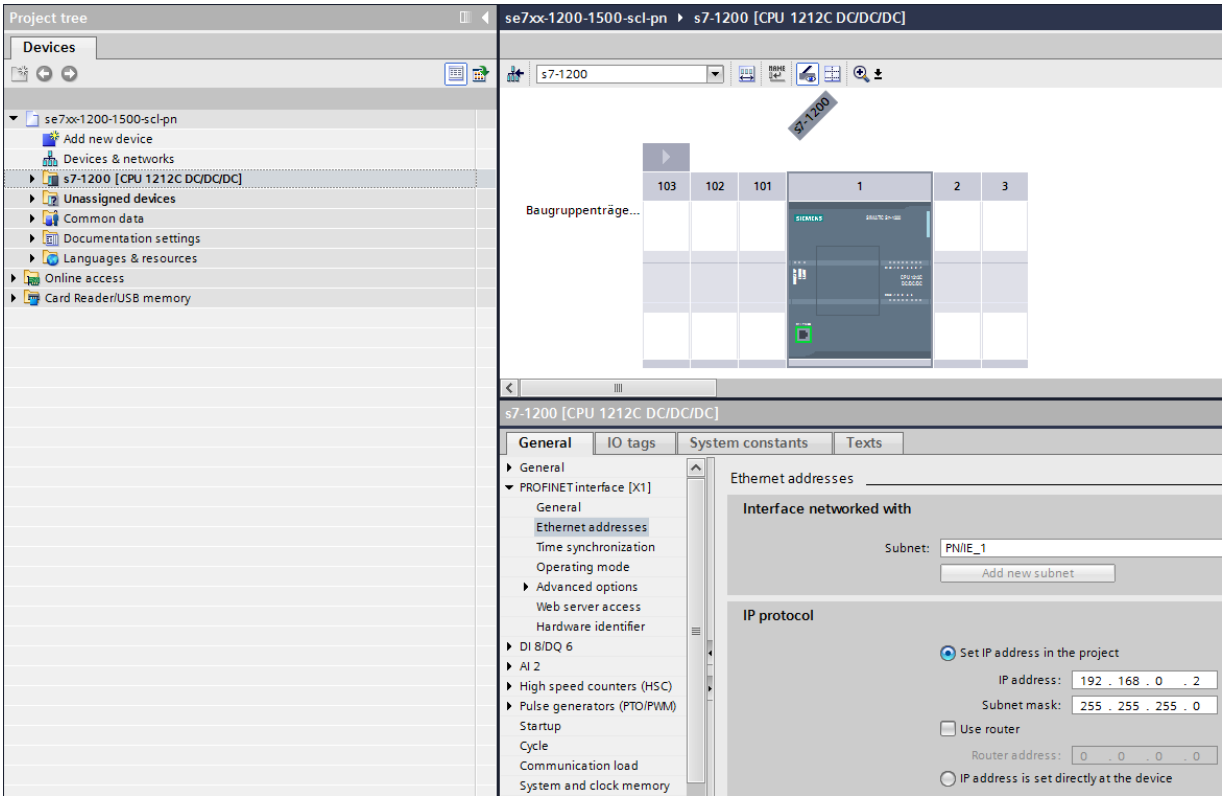
To change the IP address of the S7, select ***Devices & networks*** and double-click on the S7. Then double-click on the S7 again. Under the category ***PROFINET interface*** the IP address, Subnet mask and Router address (if needed) of the S7 can be set.

Then the S7 can be assigned to an existing “Subnet” by selecting the correct entry or you can create a new subnet by clicking “Add new subnet”. Either way, this step of assigning a subnet is necessary so TIA Portal can connect to the S7.

Likewise, a separate IP address must be assigned to the SE-7xx (Hilscher NetJack 100) in the TIA Portal.
**Please note that the IP address configured here does not correspond to the IP address configured in the SE-7xx. These are two different network interfaces, which therefore require two separate IP addresses.**
**Please set the IP address for Profinet only in the project - analog to the IP address of the S7; in particular, do not use the “Assign IP address” function under “Online & Diagnostics” for the SE-7xx (Hilscher NetJack 100).**

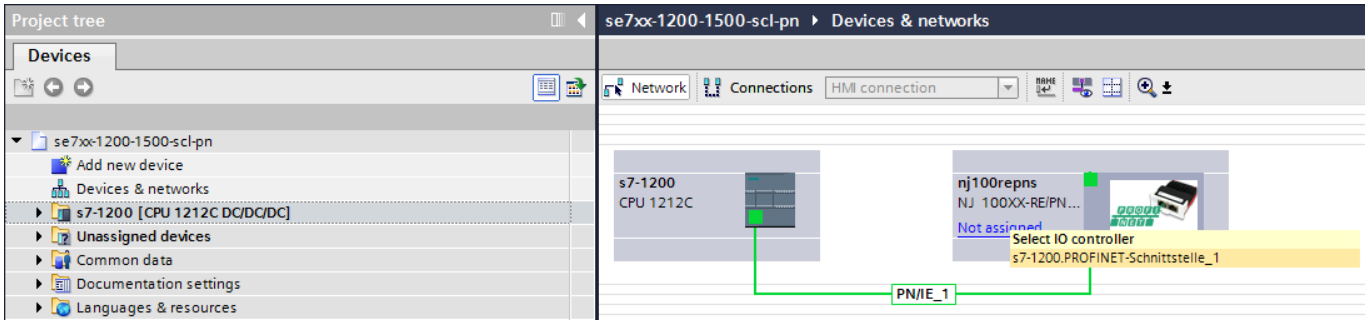
The SE-7xx (Hilscher NetJack 100) and the S7 must be in the same subnet (subnet mask), otherwise communication errors may occur.

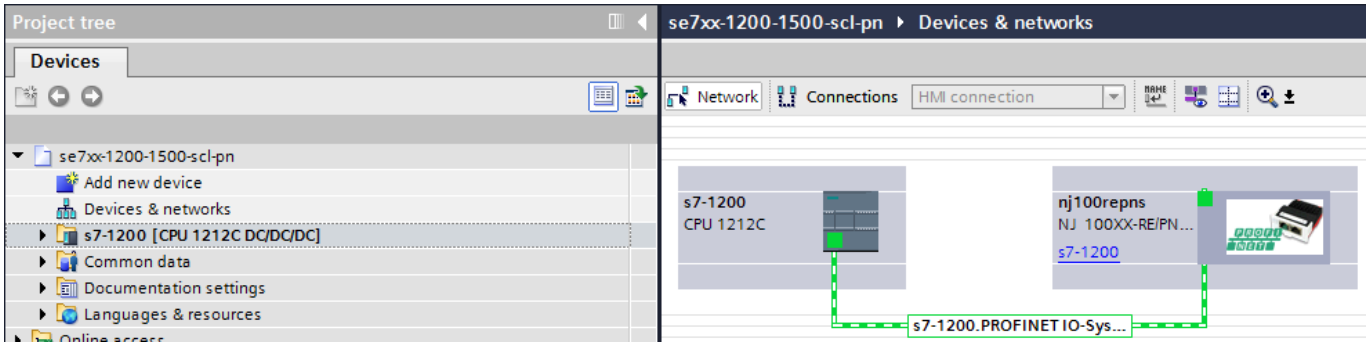
Both ports of the Hilscher NetJack 100 are equivalent; only one of them has to be connected to a Profinet compatible switch. The Hilscher NetJack 100 is only responsible for the Profinet functionality. For all other network functions of the SE-7xx, it must be connected to the switch via a separate cable (via the Ethernet socket directly on the device).



## Selecting the Profinet IO Controller (Master) in TIA Portal

Finally, the Hilscher NetJack 100 must be assigned a Profinet IO Controller (Master) in order to establish communication. Choose ***Devices & networks*** and then ***Network view***. Now the IO Controller can be selected by clicking on "*Not assigned*" at the Hilscher NetJack 100, i.e., the used S7. Here it is the S7-1212C.



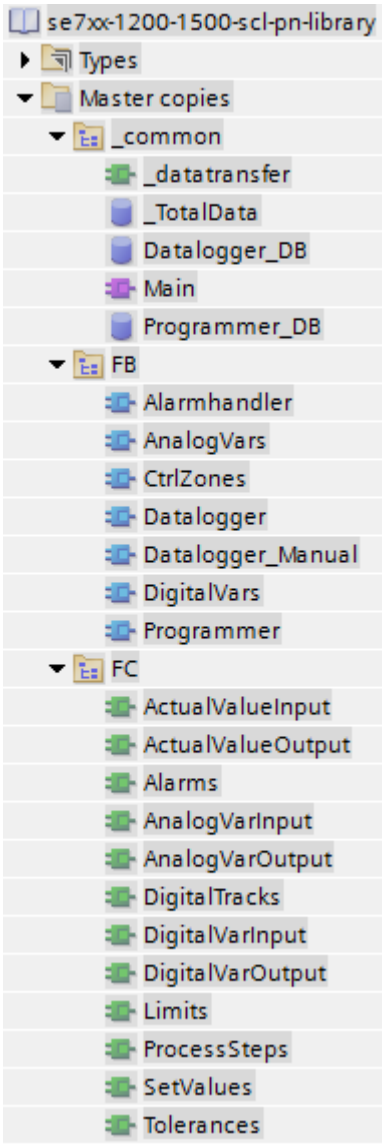

Now the projected S7 must be compiled and the hardware configuration and the blocks have to be downloaded to the PLC. In the case of an error the red Error LED of the S7 and the red BF LED of the Hilscher NetJack 100 are blinking.

The most common error cause is the Profinet name set in the SE-7xx which does not correspond to the projected Profinet name of the Hilscher NetJack 100 module in TIA Portal. In the template project, the default name ***nj100repns*** is set. If necessary, this must be changed in the Properties so that the name configured in the SE-7xx and the name configured in the project are equal. Finally recompile and reload the project to the PLC.

## Using the library modules in an existing project

The provided library ***se7xx-1200-1500-scl-pn-library*** can be used if an S7 project already exists in TIA Portal and only the Profinet communication modules shall be added to the project.
These blocks are the same as in the template project.

The following screenshot shows an overview of the contained modules:



The following modules are needed at least in order that the Profinet communication works. They must be copied into the project (***Program blocks***):

- **_datatransfer**
- **_TotalData**
- **Datalogger** [FB114] and ***Datalogger_DB*** (when using the Datalogger)

The remaining FC/FB can be integrated to the project as required. But they only work if the above-described modules are available in the project. ***_datatransfer*** is the module that performs the actual communication of both devices.
It must be integrated via OB1. An example OB1 can be found in the library.

If the datalogger is configured as active in the SE-7xx, the FB ***Datalogger*** must be called via OB1 and be served with an IDB. This step is mandatory, because the SE-7xx datalogger relies on external control and does not work otherwise.
For full flexibility, the FB ***Datalogger_Manual*** can be used as an alternative (cf. corresponding chapter).

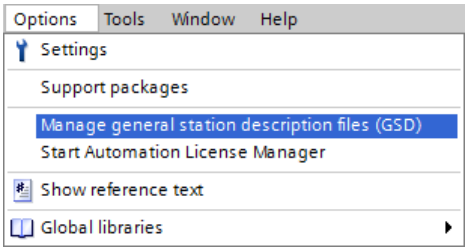
If both the Programmer block and the Datalogger block are used in the project, the input ***ProgStart*** of the ***Programmer*** block must be connected with the bit ***Start_Programmer*** of the Datalogger IDB by an OR operation, otherwise the programmer will not start. If ***ProgStart*** is not connected at all, no changes are needed.

To avoid problems when using the Programmer block and the Datalogger block at the same time, the Programmer block shall be called before the Datalogger block is called. Otherwise, the programmer of the SE-7xx may not start. It is not advised to insert ***Programmer*** and ***Datalogger*** more than one time each. To avoid further problems, only one of the two Datalogger FBs should be used.

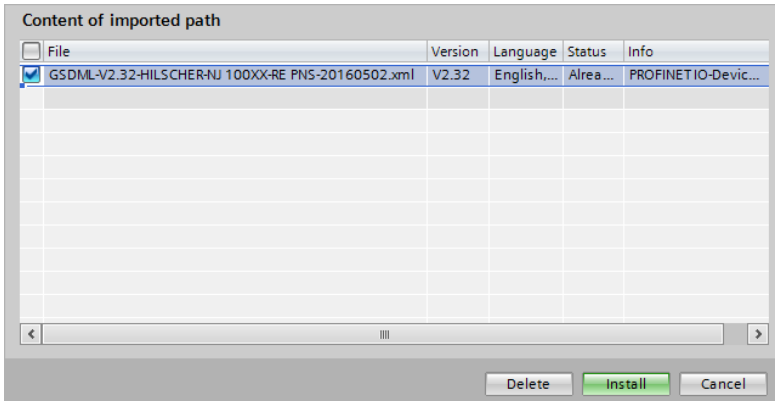
If the datalogger is not used, the FB ***Datalogger*** (or ***Datalogger_Manual***) do not have to be imported into the project.

### Installing the GSDML file in TIA Portal

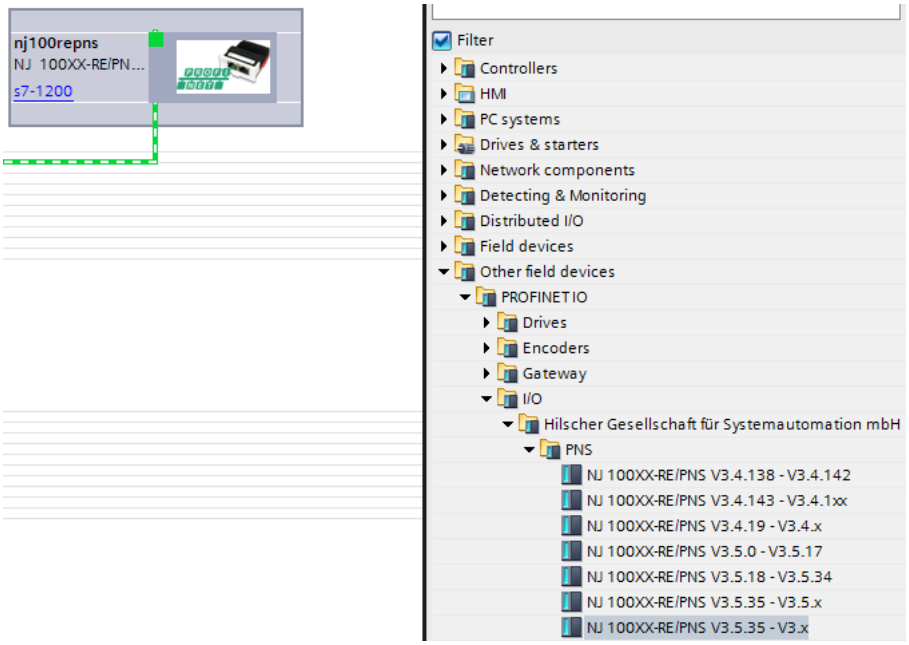
To use the Hilscher NetJack 100 in the project, the corresponding GSDML file must be imported. This happens in *Options* and *Manage general station description files (GSD)*:




Now select the storage path and install the GSDML file (the GSDML file is included in the template package):




After installation, the window can be closed. Now the Hilscher NetJack 100 is available in the hardware catalogue.
Please use the entry **NJ 100XX-RE/PNS V3.5.35 – V3.x**.

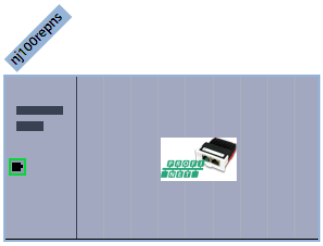



Using Drag & Drop connect the green boxes of the S7 and the Hilscher NetJack 100, alternatively the subnet can be chosen in the properties of the Hilscher NetJack 100. There the IP address can also be configured.
Finally, the Hilscher NetJack 100 must be assigned a Profinet IO Controller (Master) (see corresponding section).
Please observe the notes on assigning the IP addresses in chapter *Configuring the IP addresses*.

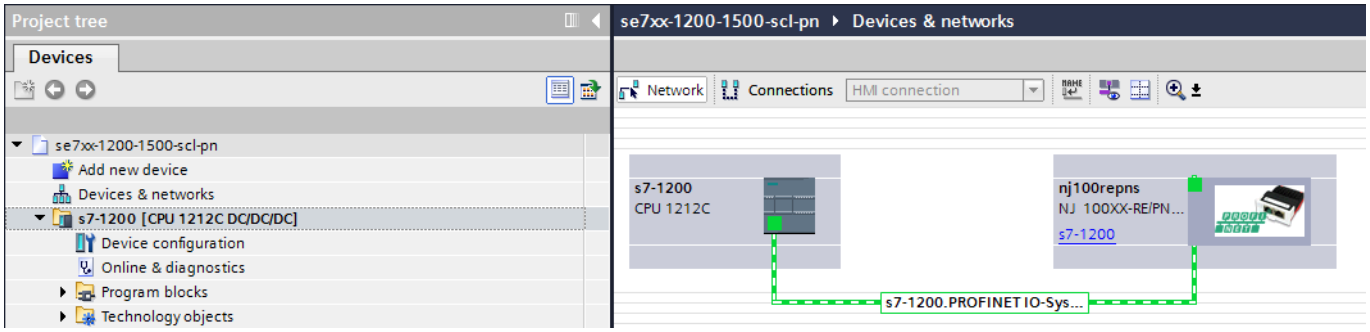
## Module configuration in TIA Portal

For the data exchange between the SE-7xx (Hilscher NetJack 100) and the S7, Profinet modules have to be configured. Therefore, you need 13 Input modules with 64 Bytes each and 9 Output modules with 64 Bytes each. These are provided by the imported GSDML file. This was already done in the template project.

The mapping of the modules to the SE-7xx data happens automatically with a fixed structure.
Only 64 Byte modules can be used.




| Module | ... | Rack | Slot | I address | Q address | Type |
|---|---|---|---|---|---|---|
| nj100repns | | 0 | 0 | | | NJ 100XX-RE/PN... |
| PN-IO | | 0 | 0 X1 | | | nj100repns |
| 64 Bytes Input_1 | | 0 | 1 | 128...191 | | 64 Bytes Input |
| 64 Bytes Input_2 | | 0 | 2 | 192...255 | | 64 Bytes Input |
| 64 Bytes Input_3 | | 0 | 3 | 256...319 | | 64 Bytes Input |
| 64 Bytes Input_4 | | 0 | 4 | 320...383 | | 64 Bytes Input |
| 64 Bytes Input_5 | | 0 | 5 | 384...447 | | 64 Bytes Input |
| 64 Bytes Input_6 | | 0 | 6 | 448...511 | | 64 Bytes Input |
| 64 Bytes Input_7 | | 0 | 7 | 512...575 | | 64 Bytes Input |
| 64 Bytes Input_8 | | 0 | 8 | 576...639 | | 64 Bytes Input |
| 64 Bytes Input_9 | | 0 | 9 | 640...703 | | 64 Bytes Input |
| 64 Bytes Input_10 | | 0 | 10 | 704...767 | | 64 Bytes Input |
| 64 Bytes Input_11 | | 0 | 11 | 768...831 | | 64 Bytes Input |
| 64 Bytes Input_12 | | 0 | 12 | 832...895 | | 64 Bytes Input |
| 64 Bytes Input_13 | | 0 | 13 | 896...959 | | 64 Bytes Input |
| 64 Bytes Output_1 | | 0 | 14 | | 128...191 | 64 Bytes Output |
| 64 Bytes Output_2 | | 0 | 15 | | 192...255 | 64 Bytes Output |
| 64 Bytes Output_3 | | 0 | 16 | | 256...319 | 64 Bytes Output |
| 64 Bytes Output_4 | | 0 | 17 | | 320...383 | 64 Bytes Output |
| 64 Bytes Output_5 | | 0 | 18 | | 384...447 | 64 Bytes Output |
| 64 Bytes Output_6 | | 0 | 19 | | 448...511 | 64 Bytes Output |
| 64 Bytes Output_7 | | 0 | 20 | | 512...575 | 64 Bytes Output |
| 64 Bytes Output_8 | | 0 | 21 | | 576...639 | 64 Bytes Output |
| 64 Bytes Output_9 | | 0 | 22 | | 640...703 | 64 Bytes Output |




In the above example only the Profinet modules have been projected which are needed for the data exchange between the SE-7xx (Hilscher NetJack 100) and the S7. The input modules and the output modules respectively must be projected without spaces to guarantee a proper data exchange.

It is possible to move the input modules and the output modules to other start addresses. For example, the 13 input modules could start at address 125.0 (***pn_inputoffset*** = 125) and the 9 output modules at address 210.0 (***pn_outputoffset*** = 210). These offsets are needed in the next step.

The S7 hardware configuration and the blocks must then be compiled and downloaded to the S7. The best way to do this is to select the S7 PLC in the Project tree (left) and to click on the buttons “Compile” and then “Download to device”. This ensures that both the hardware configuration and the software are processed.

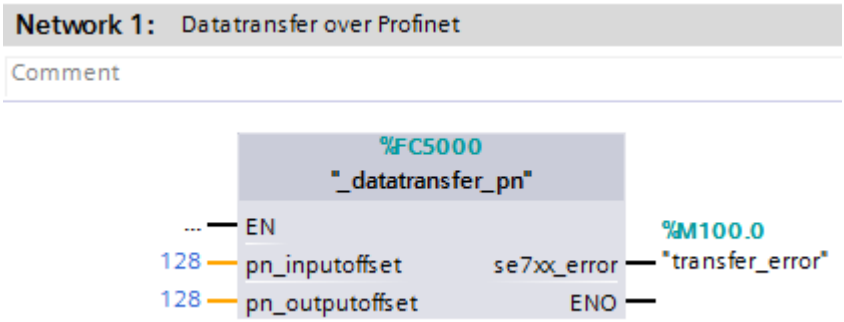
When the error “Network 1: Tag “se7xx_error” not defined” occurs, then open the block ***Main*** [OB1], go to Network 1 (***_datatransfer***) and right-click on ***se7xx_error***. Now define a tag and recompile and download the project again.

## Description of the functionality

### General

The most important module is ***Main*** [OB1]. The content is cyclically called and contains at least ***_datatransfer*** [FC5000] and when using the datalogger ***Datalogger*** [FB114] or ***Datalogger_Manual*** [FB115]. ***_datatransfer*** controls the general data exchange between both the S7 and the SE-7xx. Without this FC no Profinet communication is possible.

| _datatransfer [FC5000] | |
|---|---|
| **Parameter** | **Description** |
| pn_inputoffset | Byte offset of the Profinet input modules (status) |
| pn_outputoffset | Byte offset of the Profinet output modules (control) |
| se7xx_error | Error output of the watchdog |




The input parameters ***pn_inputoffset*** and ***pn_outputoffset*** set the offsets of the input/output modules (cf. section *Module configuration*). In the above example, the input modules for the Hilscher NetJack 100 start at I128.0 and the output modules start at Q128.0. This information is important to read and write data properly. The input modules transfer data from the SE-7xx to the S7, the output modules do the same in the opposite direction.

The output ***se7xx_error*** is set when the connection watchdog triggers. It can be connected to a flag or a DB bool variable.

The respective blocks correspond to the components of the SE-7xx. They can be easily dragged into OB1 or a self-created FC/FB. Then they are integrated by their inputs/outputs into the program sequence.
InstanceNo describes the number of the instance of the function; for example, digital track 4 or limit value 2. The value is checked for limits; for an InstanceNo outside of the valid range (for instance setvalue 23 in case of a maximum of 20 possible values) the value is set to the maximum possible instance; for values equal to or less than 0 instance 1 is selected.

The number of insertable blocks is not limited. For each inserted FB, a separate IDB (Instance DB) is created. Not used inputs/outputs of FCs can be set to an unused flag or variable in a DB. The sequence of instances of a FC/FB does not make any difference; however, each new call of a block with an already used instance number overwrites each previous call of this block with this instance.

## Overview FCs/FBs

| Name | Block | Function |
|---|---|---|
| SetValues | FC101 | Read setvalue and setvalue status |
| Alarms | FC103 | Generate alarm,<br>read alarm status |
| ProcessSteps | FC104 | Read process step status |
| DigitalTracks | FC105 | Read digital track status |
| Tolerances | FC106 | Read tolerance status,<br>external tolerance activation |
| Limits | FC107 | Read limit status |
| DigitalVarInput | FC108 | Set digital input variable in SE-7xx<br>(FI 2000-2199) |
| DigitalVarOutput | FC109 | Read digital output variable from SE-7xx<br>(FO 2000-2199) |
| AnalogVarInput | FC110 | Set analog input variable in SE-7xx<br>(values 41-80) |
| AnalogVarOutput | FC111 | Read analog output variable from SE-7xx<br>(values 1-40) |
| ActualValueInput | FC112 | Set actual value in SE-7xx,<br>force Overflow/Underflow/Break status |
| ActualValueOutput | FC113 | Read actual value from SE-7xx,<br>read actual value error status |
| | | |
| Programmer | FB100 | Control Programmer and get status |
| CtrlZones | FB102 | Control Control zone and get status |
| Alarmhandler | FB103 | Control Alarmhandler and get status |
| DigitalVars | FB108 | Write and read multiple digital variables |
| AnalogVars | FB110 | Write and read multiple analog variables |
| Datalogger | FB114 | Control Datalogger and get status<br>(automatic mode) |
| Datalogger_Manual | FB115 | Control Datalogger and get status<br>(manual mode) |

## General structure of the FCs/FBs

Inputs: InstanceNo [instance number] and respective function inputs

Outputs: Function outputs

Temp: instno_tmp: Copy of InstanceNo; used for limit check

Constant: entries: contains maximum number of instances; used for limit check

### FC5000: _datatransfer and DB5000: _TotalData

The FC5000 *_datatransfer* is responsible for sending and receiving data over Profinet. For each data area POKE_BLK is called for the actual data transfer between the I/O modules and *_TotalData*. Also, a watchdog monitors the connection. After 10 cycles without answer the error output is activated.

Only a 1:1 data transfer is possible. This means that an SE-7xx cannot be connected to several S7s and exchange data. Likewise, an S7 can only exchange data with one SE-7xx.[2]

```
// System data prolog
FOR #for_counter := 0 TO 3 DO ... END_FOR;

FOR #for_counter := 0 TO 7 DO ... END_FOR;

"_TotalData".System.control.InterfaceVersion := #InterfaceVersion;

Strg_TO_Chars(Strg := #FrameName, ...);

FOR #for_counter := 0 TO 43 DO ... END_FOR;

FOR #for_counter := 0 TO 3 DO ... END_FOR;

// TX data transfer
//
// System.tx
POKE_BLK(area_src := 16#84, dbNumber_src := 5000, byteOffset_src := #offset_control_systemdata_db, area_dest := 16#82, dbNumber_dest := 0, byteOffset_dest := #offset_control_systemdata + #pn_outputoffset, c

// Booldata.tx
POKE_BLK(area_src := 16#84, dbNumber_src := 5000, byteOffset_src := #offset_control_booldata_db, area_dest := 16#82, dbNumber_dest := 0, byteOffset_dest := #offset_control_booldata + #pn_outputoffset, count

// ActualValues.tx
POKE_BLK(area_src := 16#84, dbNumber_src := 5000, byteOffset_src := #offset_control_actualvalues_db, area_dest := 16#82, dbNumber_dest := 0, byteOffset_dest := #offset_control_actualvalues + #pn_outputoffse

// AnalogVars.tx
POKE_BLK(area_src := 16#84, dbNumber_src := 5000, byteOffset_src := #offset_control_analogvars_db, area_dest := 16#82, dbNumber_dest := 0, byteOffset_dest := #offset_control_analogvars + #pn_outputoffset, c


// TX completed
// now RX transfer
//
// System.rx
POKE_BLK(area_src := 16#81, dbNumber_src := 0, byteOffset_src := #offset_status_systemdata + #pn_inputoffset, area_dest := 16#84, dbNumber_dest := 5000, byteOffset_dest := #offset_status_systemdata_db, coun

// Booldata.rx
POKE_BLK(area_src := 16#81, dbNumber_src := 0, byteOffset_src := #offset_status_booldata + #pn_inputoffset, area_dest := 16#84, dbNumber_dest := 5000, byteOffset_dest := #offset_status_booldata_db, count :=

// Yvalues.rx
POKE_BLK(area_src := 16#81, dbNumber_src := 0, byteOffset_src := #offset_status_yvalues + #pn_inputoffset, area_dest := 16#84, dbNumber_dest := 5000, byteOffset_dest := #offset_status_yvalues_db, count := 8

// SetValues.rx
POKE_BLK(area_src := 16#81, dbNumber_src := 0, byteOffset_src := #offset_status_setvalues + #pn_inputoffset, area_dest := 16#84, dbNumber_dest := 5000, byteOffset_dest := #offset_status_setvalues_db, count

// AnalogVars.rx
POKE_BLK(area_src := 16#81, dbNumber_src := 0, byteOffset_src := #offset_status_analogvars + #pn_inputoffset, area_dest := 16#84, dbNumber_dest := 5000, byteOffset_dest := #offset_status_analogvars_db, count

// ActualValues.rx
POKE_BLK(area_src := 16#81, dbNumber_src := 0, byteOffset_src := #offset_status_actualvalues + #pn_inputoffset, area_dest := 16#84, dbNumber_dest := 5000, byteOffset_dest := #offset_status_actualvalues_db, c


// Watchdog handling
IF (("_TotalData".System.control.echobyte_s7 - "_TotalData".System.status.echobyte_s7) > #cyclewait) ... THEN ... END_IF;

```

[2] It would be technically possible to create a copy of the blocks and adapt them so that a second SE-7xx can also be controlled. However, this is not supported by Stange and is therefore not discussed in detail here. For example, it should also be noted that the peripheral address range for the modules is limited for the S7-1200.

Locally the data is stored in the DB *_TotalData*. All the FC/FB just access this DB when reading or writing data. Data from *control* are be sent to the SE-7xx, data from the SE-7xx are stored in *status*. It is wise to not use any direct connections with entries from *_TotalData*, but to use the interface of the FC/FB.

The internal assignment of the individual Profinet addresses to the functions of the SE-7xx is fully automatic, so that no setting or configuration is necessary here.

_TotalData

| | Name | Data type | Offset |
|---|---|---|---|
| 1 | Static | | |
| 2 | control | Struct | 0.0 |
| 3 | booldata | Array[0..1119] of B... | 0.0 |
| 4 | actvalues | Array[0..47] of Real | 140.0 |
| 5 | analogvars | Array[0..39] of Real | 332.0 |
| 6 | status | Struct | 492.0 |
| 7 | booldata | Array[0..1375] of B... | 0.0 |
| 8 | actvalues | Array[0..47] of Real | 172.0 |
| 9 | setvalues | Array[0..29] of Real | 364.0 |
| 10 | yvalues | Array[0..19] of Real | 484.0 |
| 11 | analogvars | Array[0..39] of Real | 564.0 |
| 12 | System | Struct | 1216.0 |
| 13 | control | Struct | 0.0 |
| 14 | status | Struct | 64.0 |
| 15 | cyclecounter | Int | 128.0 |

**_TotalData.control.booldata**

Array [0..1119] of Bool.

Contains all Bools which should be sent to the SE-7xx. These bits are set by the FC/FB.

**_TotalData.status.booldata**

Array [0..1375] of Bool.

Contains all Bools received from the SE-7xx. These bits are read by the FC/FB.

**_TotalData.control.actvalues/analogvars**

and

**_TotalData.status.actvalues/setvalues/yvalues/analogvars**

Array [0..47/39] of Real und Array [0..47/29/19/39] of Real.

Contains 32-Bit actual values(/setvalues/Y values)/analogue variables.

When sending actual values they must not be configured as “unassigned” in the SE-7xx.
The sent analogue variables 1-40 are mapped in the SE-7xx as analog variables 41-80.

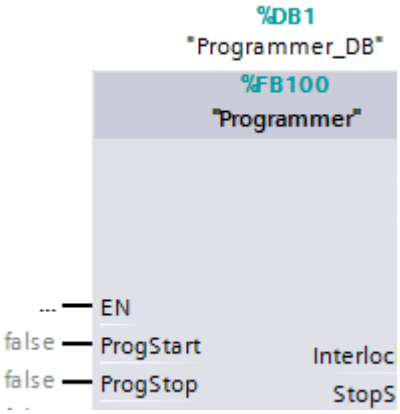
### FB100: Programmer and FB114: Datalogger

When the S7 interface is activated, the datalogger only works if the ***Datalogger*** block is programmed into the program sequence via OB1. This ***Datalogger*** block contains the logic for the job control of the datalogger. The inputs and outputs of the ***Datalogger*** FB may be wired in the project, but this is not necessary.

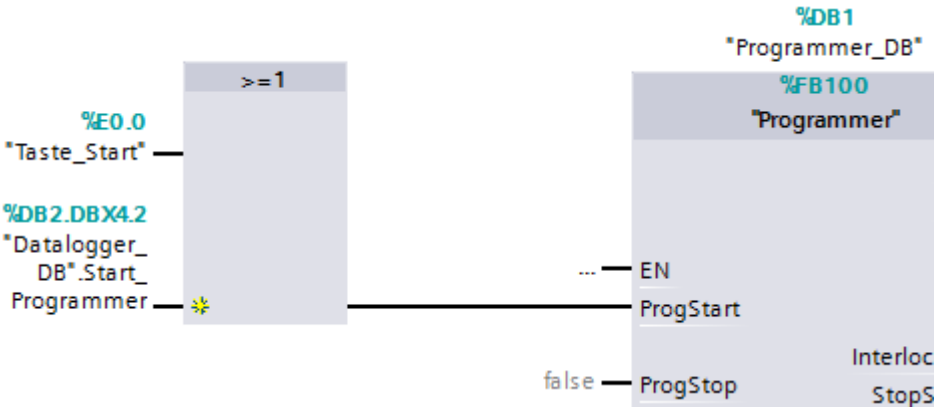
To start the datalogger (and the programmer) via the S7, set an impulse to the input ***ProcessStart***. After five seconds, the programmer will be started automatically. As soon as the program has reached the end or the operator selected END or RESET, the logger stops automatically. The recorded log data can be viewed via the log list of the datalogger in the SE-7xx.
When the logging is started from the S7, the user “plc” will be displayed in the charge details, otherwise the name of the currently logged in user.

To only start the programmer without the datalogger, you can set an impulse to the input ***ProgStart*** of ***Programmer***.

Because a ***Programmer*** block may overwrite the programmer start event, its input (***ProgStart***) must be supplied with the bit ***Start_Programmer*** of ***Datalogger_DB*** by an OR block. This step is not necessary if there is no wiring at ***ProgStart*** at all.




**Programmer without wiring at ProgStart**




**Programmer with wiring at ProgStart**

## FB114: Datalogger and FB115: Datalogger_Manual

The FB *Datalogger* works in “automatic mode”. This means it contains the logic to detect when the user wants to start the programmer via the graphical interface of the SE-7xx and then to finally start the datalogger and the programmer. This logic is necessary since most of the PLC statement list lines got obsolete with the S7 Profinet connection.
Therefore, the FB must just be called via OB1; connections to its inputs/outputs are not necessary. When needed, the datalogger and the programmer can also be started from the S7. Normally this functionality is adequate for most use cases.

But when the user wants full flexibility in controlling the datalogger, *Datalogger_Manual* can be used. It contains no logic, but offers no limits in processing the control and status signals.
To avoid problems, only one of both FB should be used in the program.

When the datalogger is enabled in the SE-7xx configuration, the Start button on the Programmer page only creates a process start event (and does not start the programmer yet). This event is displayed on the output *ProcessstartActive*. Also, the input *ProcessStart* creates a process start event. The event mainly sets the batch data in the SE-7xx.

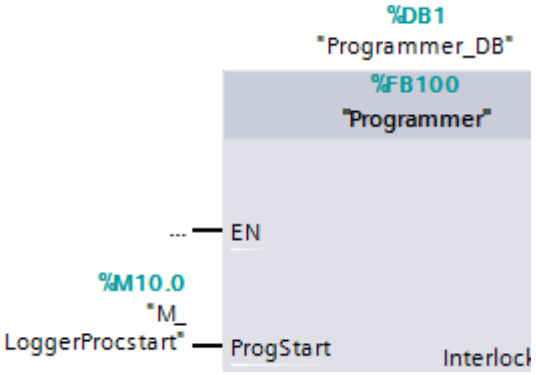
*ProcessstartActive* can be used to trigger *LogStart* to start the datalogger. Also, the input *ProgStart* of *Programmer* shall get the signal to start the programmer. *ProcessstartActive* will be reset automatically when the programmer is running (these are the two lines which must be inserted into the SE-7xx PLC statement list).

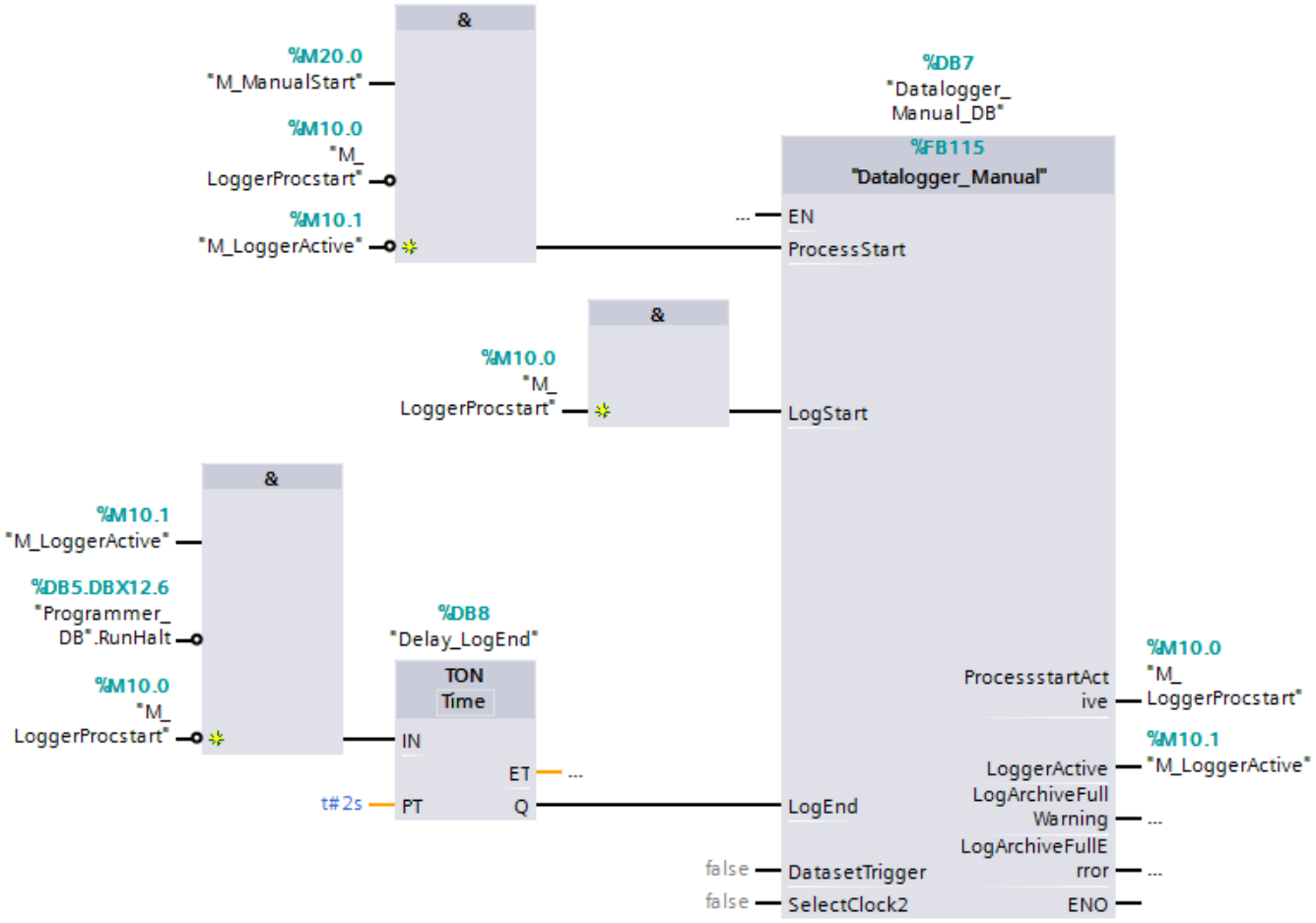
*LoggerActive* shows that the datalogger is recording. Using *LogEnd* the recording will be finished.

On the following page there is an example of using *Datalogger_Manual*.

The process start event can either be created locally (*M_ManualStart*); under the condition that no other start event exists and the datalogger is not active. Or the event can be created by the SE-7xx (Programmer).
*ProcessstartActive* outputs a signal through the event. This signal (stored in *M_LoggerProcstart*) can then be used to start the datalogger recording using *LogStart*. The flag *M_LoggerActive* then shows the active status of the datalogger.
This flag will then be used to end the datalogger recording as soon as the programmer reaches the program end or RESET.

Starting of the datalogger and the programmer can be achieved by just applying an impulse to *M_ManualStart*. The status outputs can be processed in the program if necessary.
As an alternative, the FB *Datalogger* can be used. It already contains all the logic.



### FC108/FC109, FB108: DigitalVarInput, DigitalVarOutput, DigitalVars (Digital variables)

Digital input variables of the SE-7xx can be set from the S7 with the blocks ***DigitalVarInput*** and ***DigitalVars***. These will be mapped to Function inputs (FI) 2000 to 2199 of the SE-7xx and can be used for status displays in the Visualisation, for example. These variables can only be written and not be read by the S7.

Digital output variables of the SE-7xx can be read from the S7 with the blocks ***DigitalVarOutput*** and ***DigitalVars***. These are mapped to Function outputs (FO) 2000 to 2199 of the SE-7xx and can be used for buttons in the Visualisation, for example. These variables can only be read and not be written by the S7.

The parameter ***Shift10*** at ***DigitalVars*** specifies which ten digital variables are to be read and written. With a value of 0, the first ten variables are addressed, with a value of 1, the next ten, and so on.

### FC110/FC111, FB110: AnalogVarInput, AnalogVarOutput, AnalogVars (Analog variables)

Analog input variables of the SE-7xx can be set from the S7 with the blocks ***AnalogVarInput*** and ***AnalogVars***. These will be mapped to analog variables 41-80 of the SE-7xx and can be used for status displays in the Visualisation or as substituting control zone setvalues, for example.
These variables can only be written and not be read by the S7.

Analog output variables of the SE-7xx can be read from the S7 with the blocks ***AnalogVarOutput*** and ***AnalogVars***. These are mapped to analog variables 1-40 of the SE-7xx and can be used for input fields in the Visualisation, for example.
These variables can only be read and not be written by the S7.

Both ***AnalogVarInput*** and ***AnalogVarOutput*** expect an ***InstanceNo*** in the range 1-40.

Analog variables must be configured as IEEE-Float before using them (configuration in the SE-7xx).
Other variable types are not supported.

### FC112/FC113: ActualValueInput, ActualValueOutput (Actual values)

Actual values of the SE-7xx can be set by the S7 with the block ***ActualValueInput***. These can be used there as a controller actual value, for example. They can also be configured with correction points, mean values, etc.

When sending actual values, they must not be configured as “unassigned” in the SE-7xx, but at least “linear”.

Special values according to IEEE 754 can trigger an actual value alarm in the SE-7xx:

| Actual value | IEEE 754 description | Display on SE-7xx |
|---|---|---|
| 0x7F800000 | positive infinity | Overflow |
| 0xFF800000 | negative infinity | Underflow |
| 0x7F800001 ff. | signalling NaN | Break |
| 0xFF800001 ff. | signalling NaN | Break |
| 0x7FC00000 ff. | quiet NaN | Break |
| 0xFFC00000 ff. | quiet NaN | Break |

The respective status can be generated in the SE-7xx with the inputs ***ForceOverflow***/***ForceUnderflow***/***ForceBreak***.

Actual values of the SE-7xx can be read from the S7 with the block ***ActualValueOutput***.
The output ***ActValueError*** will change to true if the actual value has an error (for example break).

**How To**

## External setvalue supply by S7

Although the SE-7xx has the ability to store programm recipes which result into setvalue definitions over time, setvalues also may be gathered remotely by an S7. This makes the programmed setvalue definition worthless as it will not come into effect. The SE-7xx will then calculate Y values with these external setvalues ignoring setvalues coming directly from the programmer.

A direct modification of the internal programmer setvalues is not possible due to technical reasons. However, there is a way to control which setvalue a control zone gets. The idea is to provide the control zone a substituting setvalue. This substituting setvalue is supplied by an analog variable from the S7. On the S7 this is simply done by inserting a block providing the desired setvalue as an input parameter. Finally, the control zone will be configured to activate the substituting setvalue permanently.

This needs a one-time configuration in the SE-7xx. In the S7 Profinet interface the 80 analog variables of the SE-7xx are divided into 40 read values (1-40) and 40 write values (41-80). Therefore, there must be at least 41 analog variables configured so the S7 can at least write one analog variable to the SE-7xx.

➔ *Configuration* > *Functions* > *Analog Variables* > *PARAM.* (button on the left side)
Please configure at least **41** values. After touching *Back*, the variables list is shown again.
For example, analog variable 41 will be configured.
After selecting variable 41 and *Edit*, the configuration page is displayed. A meaningful description will help finding the variable later. The variable type must be configured to *IEEE Float*. The display format allows setting the decimal places (the more decimal places, the less digits left of the decimal point). A maximum of two decimal places is advised. Below the low and high limit of the analog variable – the external setvalue – can be changed. You can just set them to their maximum, **-99999.9** or **99999.9**, respectively.
The control system address is not applicable here. The initialization mode changes the behavior of the analog variable after a reset. It can be changed if needed. The default value is *None*.

The configuration of the analog variable is now done.

➔ *Configuration* > *Functions* > *Control Zones*
Here all the control zones are listed. Select the respective control zone which shall be supplied with the external setvalue and choose *Edit*. Scroll down to *Subst. SV Type* and select *Variable*.
The substituting setvalue number corresponds to the number of the analog variable which will contain the external setvalue, for example: **41**. The description of the analog variable is shown in parantheses.

The configuration can now be saved by exiting. This completes the configuration in the SE-7xx.

In the S7, the *AnalogVarInput* block can be used to send the analog variable to the SE-7xx. At the input parameter *Value*, the setvalue to be sent is specified in REAL format (float). *InstanceNo* = **1** then corresponds to analog variable 41 in the SE-7xx, which was configured above as a substitute setvalue. Similarly, other alternative setvalues can also be defined, which are then transferred to the SE-7xx as analog variables 42, etc.

The *CtrlZones* block is used to enable the substitute setvalue at the selected control zone. *InstanceNo* specifies the number of the control zone (e. g. **1**). By setting **true** at the *EnableSubstSV* input, the alternative setvalue configured in the SE-7xx is finally activated. The current setvalue can then be displayed on the Controller page.

The SE-7xx controls independently of the status of the programmer. Using the *Disable* input of *CtrlZones*, the control zone can be deactivated if necessary, i. e. the Y controller output can be set to 0.0.

To obtain a Start/Stop signal from the SE-7xx, a button can be defined in the visualization, whose status can be read out via *DigitalVarOutput*. The other possibility would be to create a pseudo-program recipe in order to start and stop the programmer as usual.

## Description of the interface (FC/FB)

### _datatransfer [FC5000]

Data transfer between datablock _TotalData and SE-7xx (Hilscher NetJack 100).

| Input | Format | Function |
|---|---|---|
| pn_inputoffset | Int | Offset for Profinet modules from SE-7xx to S7 (input modules) |
| pn_outputoffset | Int | Offset for Profinet modules from S7 to SE-7xx (output modules) |

| Output | Format | Function |
|---|---|---|
| se7xx_error | Bool | Error output of watchdog |

### ActualValueInput [FC112]

Sends a float value as an actual value to the SE-7xx. The configured actual value must not be “unassigned”.
InstanceNo: 1..48

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of Actualvalue |
| Input | Real | Actualvalue input |
| ForceOverflow | Bool | Force Overflow signal on this Actualvalue |
| ForceUnderflow | Bool | Force Underflow signal on this Actualvalue |
| ForceBreak | Bool | Force Break signal on this Actualvalue |

### ActualValueOutput [FC113]

Reads an actual value and its error condition from the SE-7xx.
InstanceNo: 1..48

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of Actualvalue |

| Output | Format | Function |
|---|---|---|
| Value | Real | Actualvalue output (value) |
| ActValueError | Bool | Actualvalue has an error |

### Alarms [FC103]

Generates an alarm in SE-7xx (1-200) and reads current alarm status (1-240) from SE-7xx.
System alarms (201-240) can only be read and AlarmInput will be ignored.
InstanceNo: 1..240

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of alarm |
| AlarmInput | Bool | Generate selected alarm |

| Output | Format | Function |
|---|---|---|
| AlarmOutput | Bool | Current alarm status |

### AnalogVarInput [FC110]

Sends a float value as an analog variable to the SE-7xx (analog variables 41-80).
Analog input value 1 will be mapped as analog variable 41.
InstanceNo: 1..40

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of analog variable input |
| Value | Real | Value of analog variable input |

### AnalogVarOutput [FC111]

Reads an analog variable from the SE-7xx (analog variables 1-40).
InstanceNo: 1..40

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of analog variable output |

| Output | Format | Function |
|---|---|---|
| Value | Real | Value of analog variable output |

### DigitalTracks [FC105]

Reads the current status of the selected digital track from the SE-7xx.
InstanceNo: 1..64

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of digital track |

| Output | Format | Function |
|---|---|---|
| State | Bool | Digital track active |

### DigitalVarInput [FC108]

Sends a digital variable to the SE-7xx. They are mapped as FI 2000-2199.
InstanceNo: 1..200

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of digital input |
| State | Bool | State of selected digital input |

### DigitalVarOutput [FC109]

Reads a digital variable from the SE-7xx. They are mapped as FO 2000-2199.
InstanceNo: 1..200

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of digital output |

| Output | Format | Function |
|---|---|---|
| State | Bool | State of selected digital output |

### Limits [FC107]

Reads the current status of the selected limit from the SE-7xx.
InstanceNo: 1..40

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of limit |

| Output | Format | Function |
|---|---|---|
| Crossed | Bool | Limit crossed |

### ProcessSteps [FC104]

Reads the current status of the selected process step from the SE-7xx.
InstanceNo: 1..50

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of process step |

| Output | Format | Function |
|---|---|---|
| State | Bool | Process step active |

### SetValues [FC101]

Returns the value and status of a setvalue from the SE-7xx.
InstanceNo: 1..30

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of setvalue |

| Output | Format | Function |
|---|---|---|
| Value | Real | Value of setvalue |
| ManualSVEnabled | Bool | Manual setvalue setting enabled |
| SVRising | Bool | Setvalue is rising |
| SVConst | Bool | Setvalue is constant |
| SVFalling | Bool | Setvalue is falling |
| SVRampsection | Bool | Setvalue is currently in ramp section |

### Tolerances [FC106]

Enables a tolerance (if configured as external) and returns status of the selected tolerance from the SE-7xx.
InstanceNo: 1..40

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of tolerance |
| EnableTol | Bool | Enable tolerance |

| Output | Format | Function |
|---|---|---|
| PlusTolCrossed | Bool | Upper tolerance crossed |
| MinusTolCrossed | Bool | Lower tolerance crossed |

### Alarmhandler [FB103]

Controls the alarmhandler of the SE-7xx and returns its status.

| Input | Format | Function |
|---|---|---|
| AckAcoustic | Bool | Acknowledge acoustic alarm |
| AckOptical | Bool | Acknowledge optical common alarm |
| AlarmComing_bcdbin | Bool | Alarm is coming; using BCD/binary notation for alarm selection |
| AlarmGoing_bcdbin | Bool | Alarm is going; using BCD/binary notation for alarm selection |
| ClearAll | Bool | Clear all alarms |
| Lock209 | Bool | Lock or unlock Alarm 209 (void actualvalues) |

| Output | Format | Function |
|---|---|---|
| AcousticAck | Bool | Acoustic alarm has been acknowledged |
| OpticalAck | Bool | Optical alarm has been acknowledged |
| AcousticOut | Bool | Acoustic alarm output |
| OpticalOut | Bool | Optical alarm output |
| CommonOut | Bool | Common alarm output |
| FeedbackCommonack | Bool | Feedback for common acknowledging (acknowledging all alarms) |
| FeedbackSingleack | Bool | Feedback for single acknowledging (acknowledging one alarm) |
| AlarmnrReceived_bcdbin | Bool | The alarm number in BCD/binary format has been received |
| Priority1 | Bool | Priority 1 alarm active |
| Priority2 | Bool | Priority 2 alarm active |
| Priority3 | Bool | Priority 3 alarm active |
| Priority4 | Bool | Priority 4 alarm active |
| Priority5 | Bool | Priority 5 alarm active |
| Priority6 | Bool | Priority 6 alarm active |
| Priority7 | Bool | Priority 7 alarm active |
| Priority8 | Bool | Priority 8 alarm active |

### AnalogVars [FB110]

Sends and reads multiple analog variables to/from the SE-7xx.
Analog input variables are written to analog variables 41-80 of the SE-7xx.
Analog output variables are read from analog variables 1-40 of the SE-7xx.

| Input | Format | Function |
|---|---|---|
| Input1 | Real | Value of analog variable input |
| Input2 | Real | Value of analog variable input |
| […] | […] | […] |
| Input40 | Real | Value of analog variable input |

| Output | Format | Function |
|---|---|---|
| Output1 | Real | Value of analog variable output |
| Output2 | Real | Value of analog variable output |
| […] | […] | […] |
| Output40 | Real | Value of analog variable output |

### CtrlZones [FB102]

Controls control zone settings of the SE-7xx and returns its status.
InstanceNo: 1..20

| Input | Format | Function |
|---|---|---|
| InstanceNo | Int | Number of controller |
| PIDselect | Int | Number of PID parameter set (1-8) |
| Disable | Bool | Disable controller |
| EnableYlimit | Bool | Enable Y limiter for controller |
| EnableSubstSV | Bool | Enable substituting setvalue for controller |
| EnableSubstAV | Bool | Enable substituting actual value for controller |
| EnableYhandConstVal | Bool | Enable Y-HAND constant value |
| EnableXtrack | Bool | Enable X-Tracking for controller |
| EnableYtrack | Bool | Enable Y-Tracking for controller |

| Output | Format | Function |
|---|---|---|
| Value | Real | Y-value for controller |
| Heating | Bool | Controller is heating |
| Cooling | Bool | Controller is cooling |
| AVVoidalarm | Bool | Alarm: Value is broken |
| AVTolerancealarm | Bool | Alarm: Value is out of tolerance |
| YhandActive | Bool | Y-HAND is active |
| XtrackAct | Bool | X-Tracking is active |
| YtrackAct | Bool | Y-Tracking is active |
| MinusTolCrossed | Bool | Value is lower than lower tolerance |
| PlusTolCrossed | Bool | Value is higher than upper tolerance |
| LowLimCrossed | Bool | Value is lower than lower limit |
| HighLimCrossed | Bool | Value is higher than upper limit |

### DigitalVars [FB108]

Sends and reads multiple digital variables to/from the SE-7xx.
Digital inputs are written to FI 2000-2199. Digital outputs are read from FO 2000-2199.
Shift10 can be used to set the focus on which values to write/read; e.g. if Shift10 is 5, digital variables 51-60 are written/read.
Shift10: 0..19

| Input | Format | Function |
|---|---|---|
| Shift10 | Int | Offset multiplicated by 10 to access all 200 inputs/outputs |
| Input1 | Bool | Digital input |
| Input2 | Bool | Digital input |
| […] | […] | […] |
| Input10 | Bool | Digital input |

| Output | Format | Function |
|---|---|---|
| Output1 | Bool | Digital output |
| Output2 | Bool | Digital output |
| […] | […] | […] |
| Output10 | Bool | Digital output |

### Programmer [FB100]

Controls the programmer of the SE-7xx and returns its status.

| Input | Format | Function |
|---|---|---|
| ProgStart | Bool | Program control: START program (without datalogger) |
| ProgStop | Bool | Program control: STOP program |
| ProgReset | Bool | Program control: RESET program |
| ProgInterlock | Bool | Program control: INTERLOCK program |
| JumpNextSect | Bool | Program control: Jump to next section |
| JumpProgEnd | Bool | Program control: Jump to program end |
| StopSectEndEnable | Bool | Program control: Stop at section end [static] |
| ContSectEnd | Bool | Program control: Continue (if section end reached) [impulse] |
| SetNoProg | Bool | Program control: Set current program to "no program" |
| SetProg | Bool | Program control: Select program using SetProgNr (integer) |
| SetProgNr | Int | Program control: Set program number |
| JumpAV | Bool | Program control: Feature "Jump to actual value" |
| JumpAVDest | Int | Program control: Selection of controlzone for feature “Jump to actual value" |

| Output | Format | Function |
|---|---|---|
| ProgNr | Int | Program status: Current program number |
| SectNr | Int | Program status: Current section number |
| Reset | Bool | Program status: RESET |
| Run | Bool | Program status: RUN |
| Stop | Bool | Program status: STOP |
| InterlockActive | Bool | Program status: INTERLOCK active |
| StopSectEnd | Bool | Program status: STOP after reaching section end |
| ProgEnd | Bool | Program status: Program END |
| RunHalt | Bool | Program status: Program in RUN or STOP |
| PwrFailStop | Bool | Program status: STOP after power failure |
| AVNotfound | Bool | Program status: Provided actual value not found |
| NewSectLoaded | Bool | Program status: New program section loaded |
| ProgSelected | Bool | Program status: Program selected |
| ProgNotfound | Bool | Program status: Program not found |
| CurrentProgChanged | Bool | Program status: Current program changed |
| StarttimeEnabled | Bool | Program status: Program start at specific time/date enabled |

### Datalogger [FB114]

Controls the SE-7xx datalogger and returns its status (“automatic mode”).
Must be inserted if the datalogger is used (after the Programmer block).
Do not use with **Datalogger_Manual [FB115]**.

| Input | Format | Function |
|---|---|---|
| ProcessStart | Bool | Starts the datalogger and after 5 seconds starts the programmer |
| DatasetTrigger | Bool | Trigger dataset |
| SelectClock2 | Bool | Select clock 2 instead of clock 1 for data logging |

| Output | Format | Function |
|---|---|---|
| ProcessstartActive | Bool | Received a local (PLC) or remote (SE-7xx) process start event |
| LoggerActive | Bool | Datalogger active |
| LogArchiveFullWarning | Bool | Log archive nearly full |
| LogArchiveFullError | Bool | Log archive completely full |

### Datalogger_Manual [FB115]

Controls the SE-7xx datalogger and returns its status (“manual mode”).
Must be inserted if the datalogger is used (after the Programmer block).
Do not use with **Datalogger [FB114]**.

| Input | Format | Function |
|---|---|---|
| ProcessStart | Bool | Generates a process start event |
| LogStart | Bool | Starts the current datalogger recording |
| LogEnd | Bool | Stops the current datalogger recording |
| DatasetTrigger | Bool | Trigger dataset |
| SelectClock2 | Bool | Select clock 2 instead of clock 1 for data logging |

| Output | Format | Function |
|---|---|---|
| ProcessstartActive | Bool | Received a local (PLC) or remote (SE-7xx) process start event |
| LoggerActive | Bool | Datalogger active |
| LogArchiveFullWarning | Bool | Log archive nearly full |
| LogArchiveFullError | Bool | Log archive completely full |