Documentation

Profinet connection
S7-300 ↔ SE-7xx
via Hilscher NetJack 100

Documentation: 2018-03-06
valid for: Version 1.1

Author: Lukas Jolbej

In this documentation the S7 Profinet connection to the STANGE SE-7xx device is explained.

Used devices:

- Stange SE-702
- Siemens S7-315-2 PN/DP (6ES7 315-2EH14-0AB0) as IO-Controller
- Hilscher NetJack 100 (NJ 100DN-RE/PNS) as IO-Device

connected via a 100 MBit/s switch

Used software:

- Siemens STEP7 5.5 SP4
- Windows 7 SP1
- Device version 7.0.1.11 for the SE-702
- Firmware version 3.2 for the S7-315-2 PN/DP
- Firmware version 3.5.26.0 for the Hilscher NetJack 100

Requirements:

- The feature must be licensed in the SE-7xx

corresponding STEP7 templates:

- se7xx-300-awl-pn (template project), version 1.1
- se7xx-300-awl-pn-library (library), version 1.1

Table of contents

Getting started

Function overview

What are the features?

- Control and query programmer
- Control and query control zone
- Read setvalue and its status
- Control and query alarmhandler
- Generate alarm and query alarm status
- Control and query datalogger
- Read process step status
- Read digital track status
- Read tolerance status, enable tolerance (if configured as external activatable)
- Read limit status
- Set digital variable in SE-7xx (starting at FI 2000)
- Read digital variable from SE-7xx (starting at FO 2000)
- Set analog variable in SE-7xx (values 41-80)
- Read analog variable from SE-7xx (values 1-40)
- Set actual value in SE-7xx, force Overflow/Underflow/Break status
- Read 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
  - ➔ Control zone status, Setvalue status, Actual value status, Tolerance status, Limit status, Programmer status, Process step status, Digital track status, Digital output variables, Alarmhandler status, Alarm status, 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, Digitale input variables, Alarmhandler control, Alarm inputs, Datalogger control
- 32 Bit floating point values (REAL)
  - ➔ Analog variables 41-80, Actual values (correction points, mean values etc. can be configured via SE-7xx)

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.

Configure hardware option in the SE-7xx

First the hardware option is 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 configure the ***Profinet name***, by default ***nj100repns***.

Datalogger configuration (PLC statement list)

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

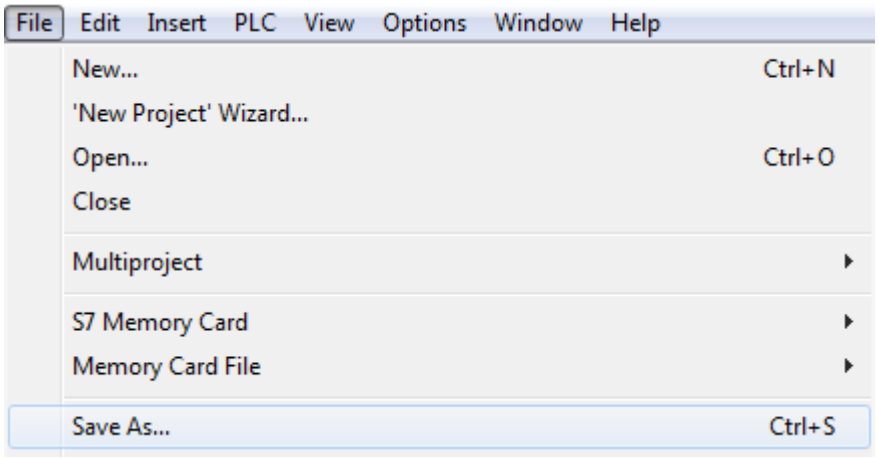
```
L FO 768
R FO 1311
```

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***.

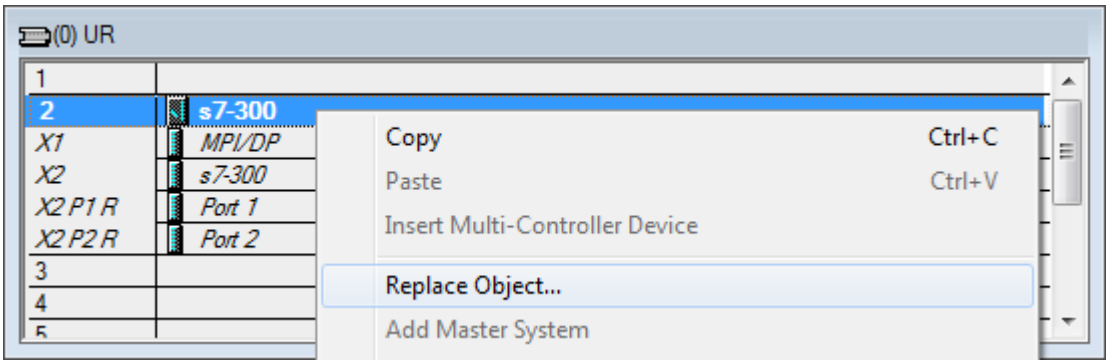
For more information on configuring the datalogger see the corresponding documentation.

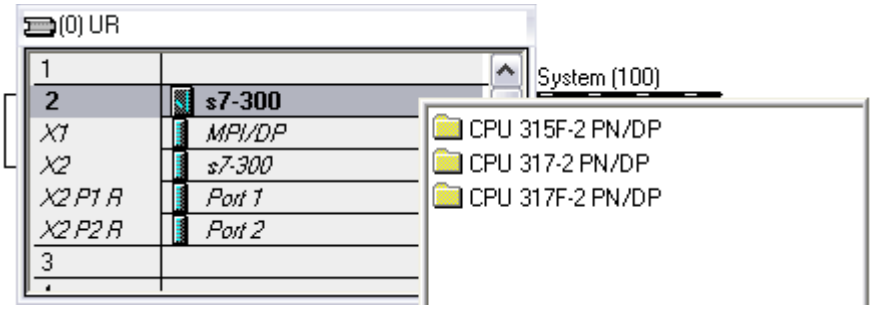
Using the project as a template

The project ***se7xx-300-awl-pn*** can be used as a template. It is loaded into STEP7 and can be saved directly via ***Project > Save as*** as a copy with a new name. This enables to use the template again.

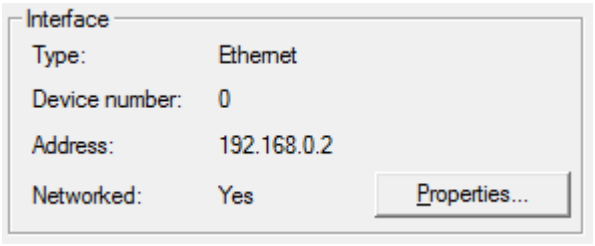


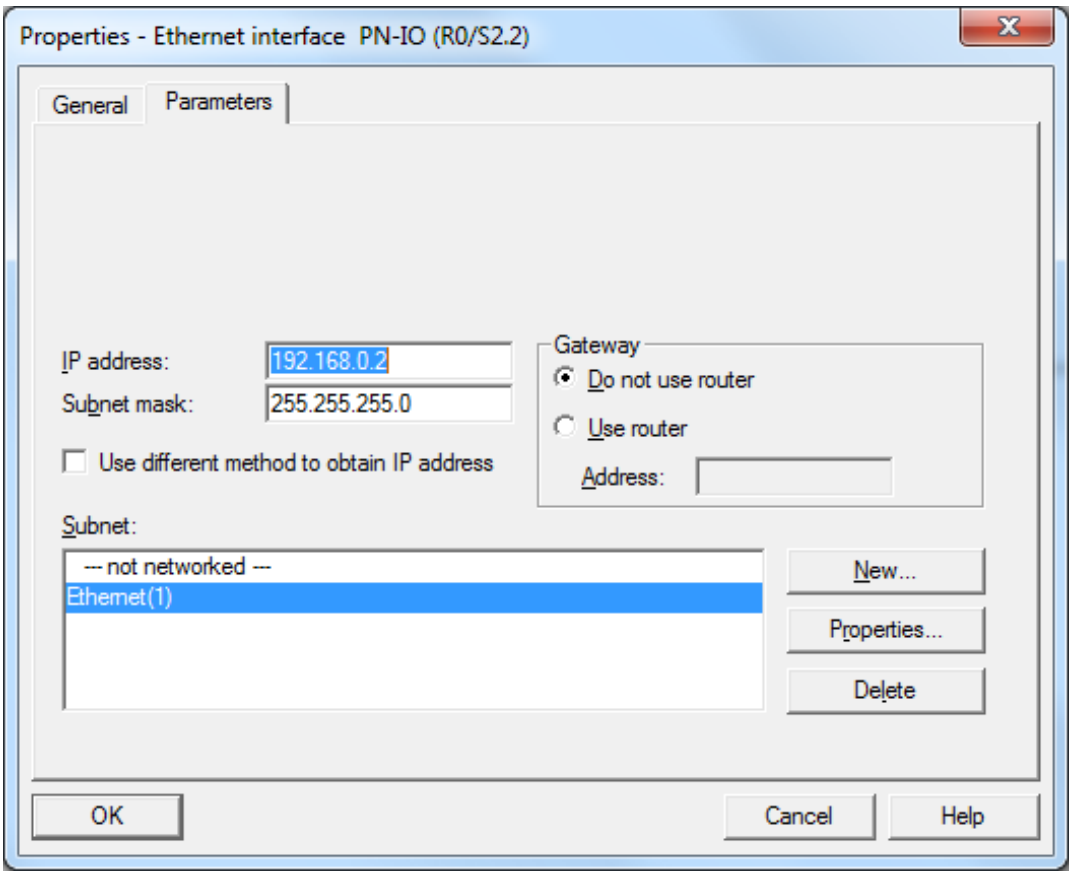
In the template project, there is a projected S7 315-2 PN/DP. The project must be adapted if another PLC than S7 315-2 PN/DP (S7 317-2 PN/DP, S7 315F-2 PN/DP, S7 317F-2 PN/DP) is used: in ***HW Config*** the S7 must be selected by right mouse click and ***Replace object*** shows a selection for the replacement hardware. Now the used device can be selected.





To change the IP address, subnet mask and/or gateway of the S7, doubleclick on the ***X2*** entry, and click ***Properties***.





Using the library modules in an existing project

The provided library ***se7xx-300-awl-pn-library*** can be used if a S7 project already exists in STEP7 and only the Profinet communication modules shall be added to the project. The following screenshot shows an overview of the contained modules:

OB1	Main	STL	150	Organization Block
OB86	RACK_FLT	FBD	38	Organization Block
FB1000	Programmer	STL	850	Function Block
FB1002	CtrlZones	STL	446	Function Block
FB1003	Alarmhandler	STL	250	Function Block
FB1008	DigitalVars	STL	342	Function Block
FB1010	AnalogVars	STL	710	Function Block
FB1014	Datalogger	STL	494	Function Block
FB1015	Datalogger_Manual	STL	166	Function Block
FC1001	Setvalues	STL	220	Function
FC1003	Alarms	STL	200	Function
FC1004	ProcessSteps	STL	148	Function
FC1005	DigitalTracks	STL	148	Function
FC1006	Tolerances	STL	200	Function
FC1007	Limits	STL	148	Function
FC1008	DigitalVarInput	STL	148	Function
FC1009	DigitalVarOutput	STL	148	Function
FC1010	AnalogVarInput	STL	154	Function
FC1011	AnalogVarOutput	STL	154	Function
FC1012	ActualValueInput	STL	222	Function
FC1013	ActualValueOutput	STL	182	Function
FC5000	_datatransfer	STL	1582	Function
DB1	Programmer_DB	DB	50	Instance data block for FB 1000
DB2	Datalogger_DB	DB	130	Instance data block for FB 1014
DB5000	_TotalData	DB	1382	Data Block
SFB3	TP	STL	---	System function block
SFB4	TON	STL	---	System function block
SFC20	BLKMOV	STL	---	System function

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

- **_datatransfer**
- **_TotalData**
- **Datalogger** [FB1014] 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, otherwise the SE-7xx datalogger will not work. For full flexibility, ***Datalogger_Manual*** can be used instead (cf. corresponding chapter).

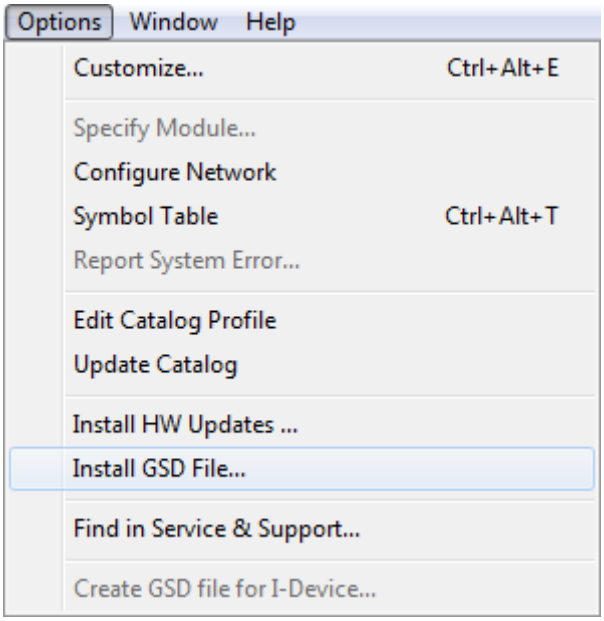
If both the ***Programmer*** block and the ***Datalogger*** block are used in the project, the input ***ProgStart*** of the ***Programmer*** block must be connected with the bit ***Start_Programmer*** of the ***Datalogger*** IDB by an OR block, otherwise the programmer will not start. If ***ProgStart*** 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.
The ***Programmer*** block and the ***Datalogger*** block shall only be inserted once each in the project.

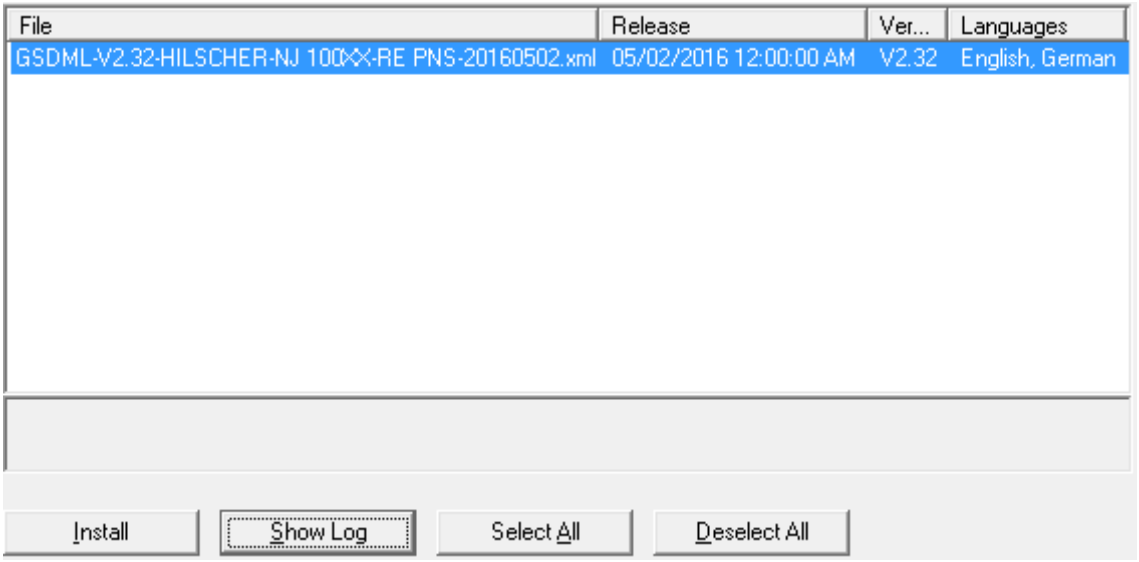
The OB86 (RACK_FLT) keeps the S7 in RUN after a connection failure (rack fault). If the block is not transferred, the CPU goes to STOP after the loss of connection, e.g. due to an unplugged cable. There are no further instructions in the block.

Installing the GSDML file

To use the Hilscher NetJack 100 in the project, the corresponding GSDML file must be imported. This happens in ***HW Config***, ***Options*** and then ***Install GSD file***.



In the following window select the storage path and install the GSDML file:

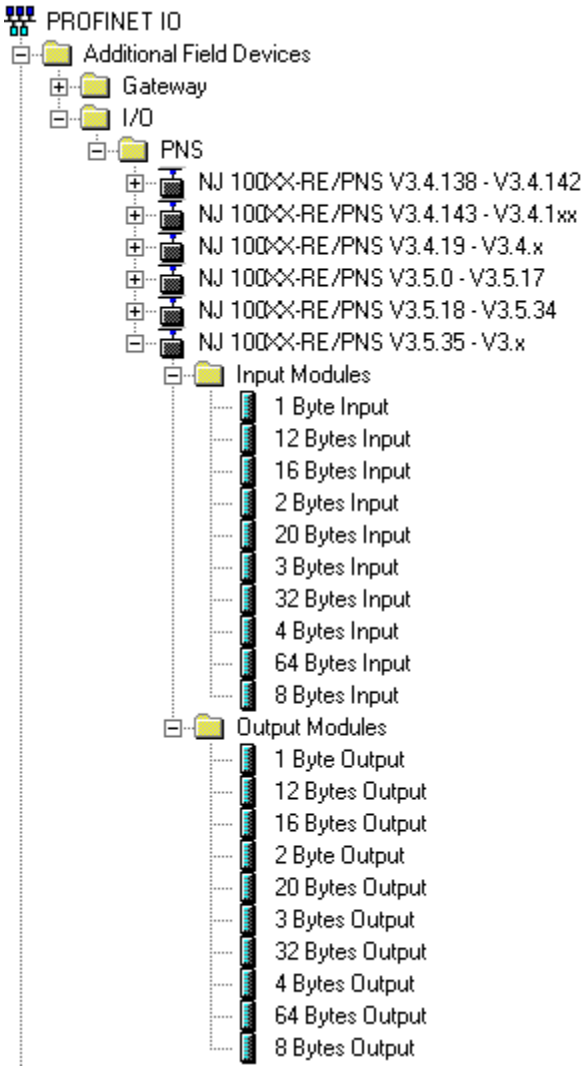


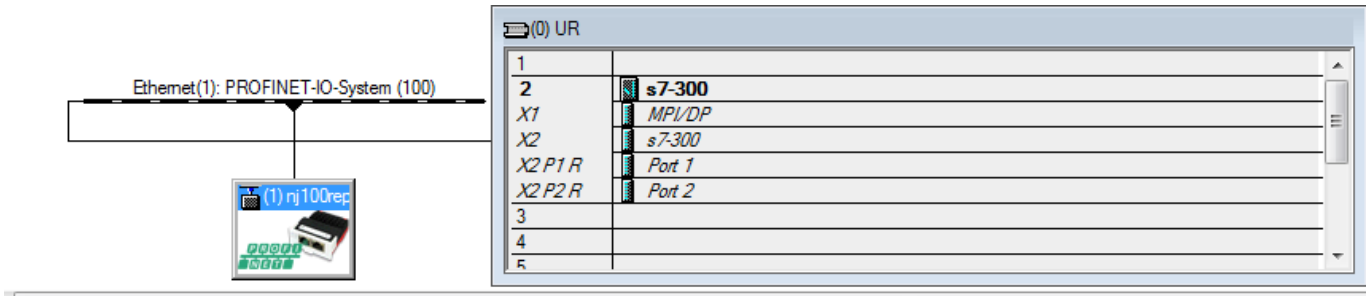
After installation, the window can be closed. Now the NetJack 100 is available in the hardware catalogue.

Module configuration

For the data exchange between the SE-7xx (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.

These modules are used in the data exchange process. It is not intended to change the amount of exchange data and the number of modules. The mapping of the modules to the SE-7xx data happens automatically with a fixed structure.





(1) nj100repns

Slot	Module	Order number	I Address	Q address	Diagnostic Address	Comment	Access
0	*nj100repns*	*162x 100*			*2042**		*Full*
X1	*PN-IO*				*2041**		*Full*
X1 P1 R	*Port 1*				*2040**		*Full*
X1 P2 R	*Port 2*				*2039**		*Full*
1	64 Bytes Input		256...319				Full
2	64 Bytes Input		320...383				Full
3	64 Bytes Input		384...447				Full
4	64 Bytes Input		448...511				Full
5	64 Bytes Input		512...575				Full
6	64 Bytes Input		576...639				Full
7	64 Bytes Input		640...703				Full
8	64 Bytes Input		704...767				Full
9	64 Bytes Input		768...831				Full
10	64 Bytes Input		832...895				Full
11	64 Bytes Input		896...959				Full
12	64 Bytes Input		960...1023				Full
13	64 Bytes Input		1024...1087				Full
14	64 Bytes Output			256...319			Full
15	64 Bytes Output			320...383			Full
16	64 Bytes Output			384...447			Full
17	64 Bytes Output			448...511			Full
18	64 Bytes Output			512...575			Full
19	64 Bytes Output			576...639			Full
20	64 Bytes Output			640...703			Full
21	64 Bytes Output			704...767			Full
22	64 Bytes Output			768...831			Full

In the above example only the Profinet modules have been projected which are needed for the data exchange between the SE-7xx (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). The offsets are needed in the next step.

Now download the project to the S7.

The size of the process images of the input area and the output area must cover all the 22 modules. You can change the values in the CPU object properties window and then ***Cycle/Clock memory***.

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*** [FB1014] or ***Datalogger_Manual*** [FB1015]. ***_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

```
Netzwerk 1 : Datatransfer over Profinet

    CALL  "_datatransfer"              FC5000
     pn_inputoffset :=256
     pn_outputoffset:=256
     se7xx_error    :=M100.0
```

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 NetJack 100 start at I256.0 and the output modules start at Q256.0. These 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.

The respective blocks correspond to the components of the STANGE device. 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. Not used blocks can be deactivated via input EN (set to false).

Overview FCs/FBs

Name	Block	Function
SetValues	FC1001	Read setvalue and setvalue status
Alarms	FC1003	Generate alarm and read alarm status
ProcessSteps	FC1004	Read process step status
DigitalTracks	FC1005	Read digital track status
Tolerances	FC1006	Read tolerance status, external tolerance activation
Limits	FC1007	Read limit status
DigitalVarInput	FC1008	Set digital input variable in SE-7xx (starting at FI 2000)
DigitalVarOutput	FC1009	Read digital output variable from SE-7xx (starting at FO 2000)
AnalogVarInput	FC1010	Set analog input variable in SE-7xx (values 41-80)
AnalogVarOutput	FC1011	Read analog output variable from SE-7xx (values 1-40)
ActualValueInput	FC1012	Set actual value in SE-7xx
ActualValueOutput	FC1013	Read actual value from SE-7xx, read actual value error status
Programmer	FB1000	Control Programmer and get status
CtrlZones	FB1002	Control Control zone and get status
Alarmhandler	FB1003	Control Alarmhandler and get status
DigitalVars	FB1008	Write and read multiple digital variables
AnalogVars	FB1010	Write and read multiple analog variables
Datalogger	FB1014	Control Datalogger and get status (automatic mode)
Datalogger_Manual	FB1015	Control Datalogger and get status (manual mode)

General structure 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 any pointers are created which are used in BLKMOV 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 flag is set.

```
⊞ Network 1: Prepare datatransfer

⊞ Network 2 : System.tx: build anypointers

⊞ Network 3 : System.tx: BLKMOV

⊞ Network 4 : System.rx: build anypointers

⊞ Network 5 : System.rx: BLKMOV

⊞ Network 6 : Booldata.rx: build anypointers

⊞ Network 7 : Booldata.rx: BLKMOV

⊞ Network 8 : Yvalues.rx: build anypointers

⊞ Network 9 : Yvalues.rx: BLKMOV

⊞ Network 10 : Setvalues.rx: build anypointers

⊞ Network 11 : Setvalues.rx: BLKMOV

⊞ Network 12 : Analogvariables.rx: build anypointers

⊞ Network 13 : Analogvariables.rx: BLKMOV

⊞ Network 14 : ActualValues.rx: build anypointers

⊞ Network 15 : ActualValues.rx: BLKMOV

⊞ Network 16 : ---

⊞ Network 17 : Watchdog handling

⊞ Network 18 : ---

⊞ Network 19 : Booldata.tx: build anypointers

⊞ Network 20 : Booldata.tx: BLKMOV

⊞ Network 21 : Analogvariables.tx: build anypointers

⊞ Network 22 : Analogvariables.tx: BLKMOV

⊞ Network 23 : ActualValues.tx: build anypointers

⊞ Network 24 : ActualValues.tx: BLKMOV
```

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.

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	control	STRUCT		
+0.0	booldata	ARRAY[0..1119]		
*0.1		BOOL		
+140.0	actvalues	ARRAY[0..47]		
*4.0		REAL		
+332.0	analogvars	ARRAY[0..39]		
*4.0		REAL		
=492.0		END_STRUCT		
+492.0	status	STRUCT		
+0.0	booldata	ARRAY[0..1375]		
*0.1		BOOL		
+172.0	actvalues	ARRAY[0..47]		
*4.0		REAL		
+364.0	setvalues	ARRAY[0..29]		
*4.0		REAL		
+484.0	yvalues	ARRAY[0..19]		
*4.0		REAL		
+564.0	analogvars	ARRAY[0..39]		
*4.0		REAL		
=724.0		END_STRUCT		
+1216.0	System	STRUCT		
+0.0	control	STRUCT		
+0.0	BlockStart	ARRAY[0..3]		
*1.0		BYTE		
+4.0	FrameName	ARRAY[0..7]		
*1.0		CHAR		
+12.0	InterfaceVersion	INT	0	
+14.0	echobyte_s7	BYTE	B#16#0	
+15.0	echobyte_se7xx	BYTE	B#16#0	
+16.0	reserved	ARRAY[0..43]		
*1.0		BYTE		
+60.0	BlockEnd	ARRAY[0..3]		
*1.0		BYTE		
=64.0		END_STRUCT		
+64.0	status	STRUCT		
+0.0	BlockStart	ARRAY[0..3]		
*1.0		BYTE		
+4.0	FrameName	ARRAY[0..7]		
*1.0		CHAR		
+12.0	InterfaceVersion	INT	0	
+14.0	echobyte_s7	BYTE	B#16#0	
+15.0	echobyte_se7xx	BYTE	B#16#0	
+16.0	reserved	ARRAY[0..43]		
*1.0		BYTE		
+60.0	BlockEnd	ARRAY[0..3]		
*1.0		BYTE		
=64.0		END_STRUCT		
+128.0	cyclecounter	INT	10	
=130.0		END_STRUCT		
=1346.0		END_STRUCT		

_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 and 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.

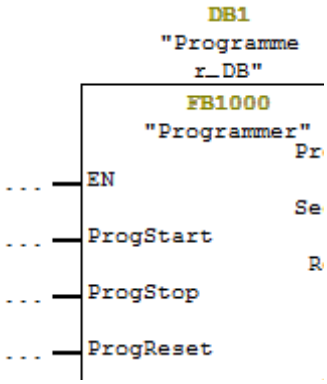
FB1000: Programmer and FB1014: Datalogger

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.

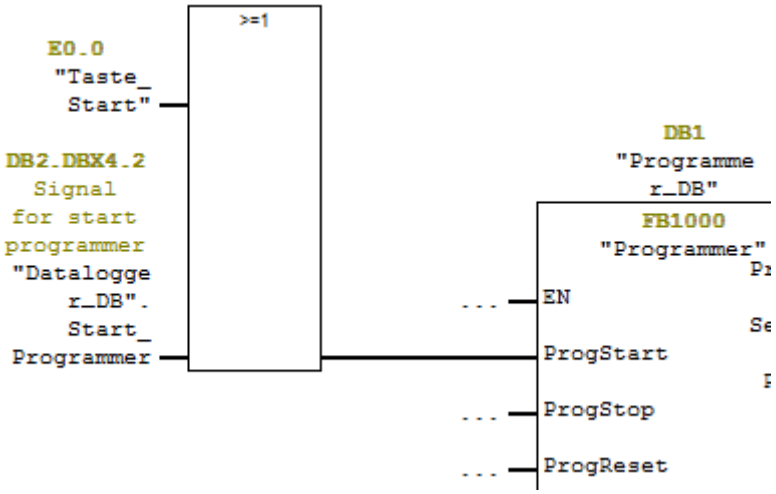
To start the datalogger (and the programmer) via the S7, set an impulse to the input *ProcessStart*. After five seconds, the programmer will be started automatically. As soon as the program has reached the end or the operator selected END or RESET, the logger stops automatically. The recorded log data 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_IDB* 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

FB1014: Datalogger and FB1015: 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 Modbus 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 right user name in the batch list of the SE-7xx (“plc” if started from the S7).

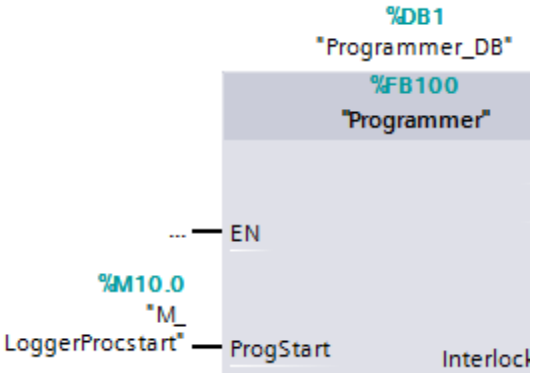
ProcessstartActive can be used to trigger ***LogStart*** to start the datalogger. Also, the input ***ProgStart*** of ***Programmer*** gets the signal to start the programmer. ***ProcessstartActive*** will be reset automatically when the programmer is running (these are the two lines which must be inserted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

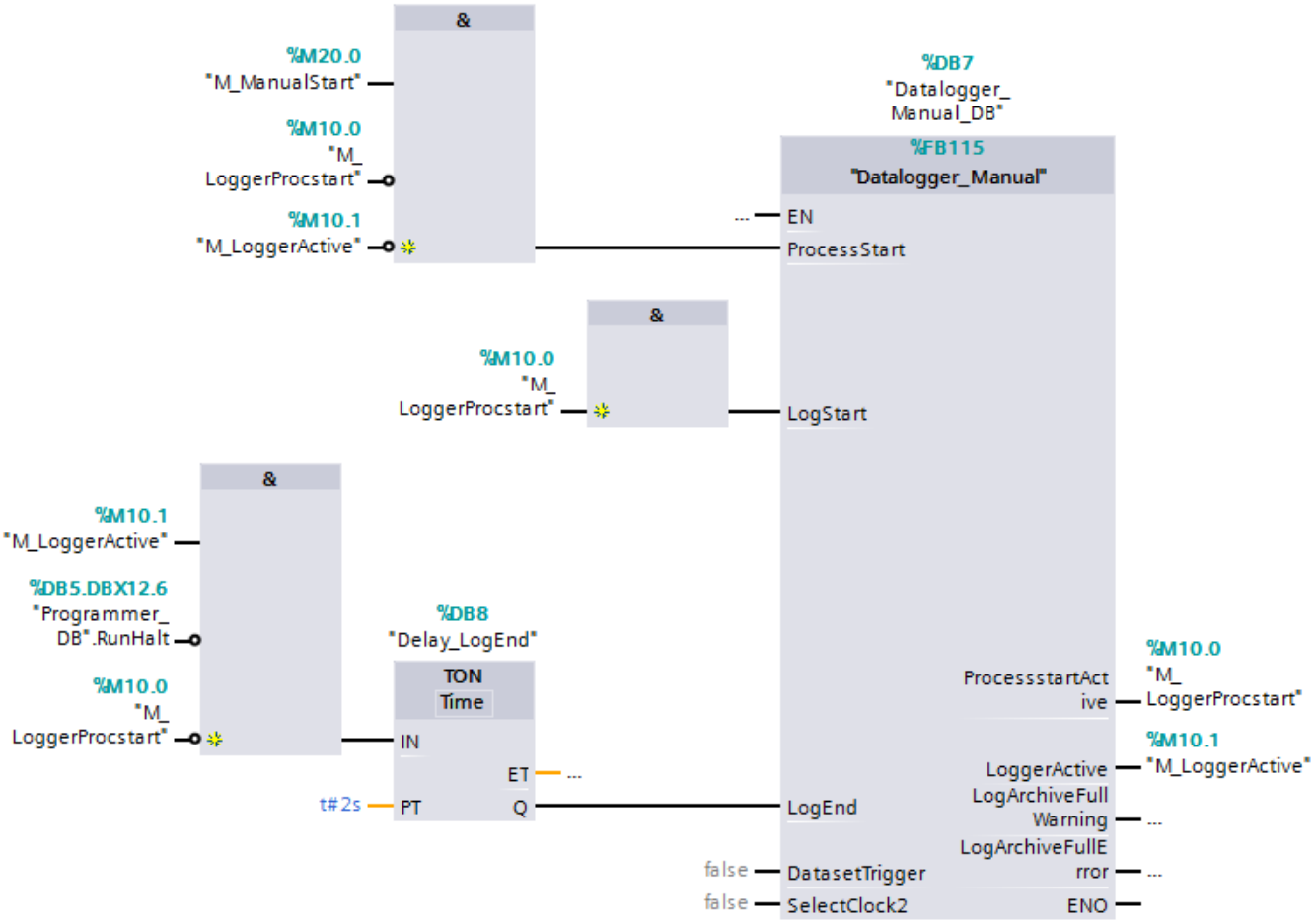
LoggerActive shows that the datalogger is recording. Using ***LogEnd*** the recording will be finished.

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 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.





FC1008/FC1009, FB1008: 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 Visualization, 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 Visualization, for example.
These variables can only be read and not be written by the S7.

FC1010/FC1011, FB1010: 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.

Analog variables must be configured as IEEE-Float before using them (configuration in SE-7xx).

FC1012/FC1013: ActualValueInput, ActualValueOutput (Actual values)

Actual values of the SE-7xx can be set from 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	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

Using the inputs ***ForceOverflow***/***ForceUnderflow***/***ForceBreak*** the corresponding status of the actual value can be forced.

Actual values of the SE-7xx can be read by 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 new S7 Modbus 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. This would also have the advantage that the integrated datalogger could then create batch records.

Description of the interface (FC/FB)

_datatransfer [FC5000]

Data transfer between datablock _TotalData and SE-7xx (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 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 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 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 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 value and status of 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 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