Dokumentation

Profinet-Anbindung
S7-1200/1500 ↔ SE-7xx
über Hilscher NetJack 100

(festes Mengengerüst)

Stand der Dokumentation: 14.09.2018
gültig für: Version 1.1

Autor: Lukas Jolbej

In dieser Dokumentation wird die S7-Profinetanbindung an den STANGE SE-7xx erläutert.

Verwendete Geräte:

- Stange SE-702
- Siemens S7-1212C DC/DC/DC als IO-Controller
- Hilscher NetJack 100 (NJ 100DN-RE/PNS) als IO-Device

verbunden über einen 100 MBit/s-Switch

Verwendete Software:

- Siemens TIA Portal V13 SP1 Update 9
- Windows 7 SP1
- Geräteversion 7.0.1.10 für den SE-702
- Firmwareversion 4.1 für die S7-1212C
- Firmwareversion 3.5.26.0 für den Hilscher NetJack 100

Voraussetzungen:

- Die Funktion muss im SE-7xx lizenziert sein

zugehörige TIA Portal-Vorlagen:

- se7xx-1200-1500-scl-pn (Vorlageprojekt), Version 1.1
- se7xx-1200-1500-scl-pn-library (Bibliothek), Version 1.1

**Beachten Sie bitte, dass abhängig vom Stand der S7-Firmware ggf. ein aktuelles TIA Portal benötigt wird.
Das Vorlageprojekt und die Bausteine wurden mit TIA Portal V13 SP1 erstellt,
können aber bei Bedarf auf eine aktuelle Version hochgerüstet werden.**

Inhaltsverzeichnis

Erste Schritte

Funktionsübersicht

Was kann man mit den Bausteinen machen?

- Programmgeber steuern und abfragen
- Regelzone steuern und abfragen
- Sollwert und -Status abfragen
- Alarmhandler steuern und abfragen
- Alarm generieren und Alarmstatus abfragen
- Datenlogger steuern und abfragen
- Verfahrensschritt-Status abfragen
- Digitalspur-Status abfragen
- Toleranz-Status abfragen, Toleranz aktivieren (falls auf Extern konfiguriert)
- Grenzwert-Status abfragen
- Digitalvariable im SE-7xx setzen (ab FE 2000)
- Digitalvariable vom SE-7xx lesen (ab FA 2000)
- Analogvariable im SE-7xx setzen (Werte 41-80)
- Analogvariable vom SE-7xx lesen (Werte 1-40)
- Istwert im SE-7xx setzen, Overflow/Underflow/Break erzeugen
- Istwert vom SE-7xx lesen, Istwert-Fehlerstatus abfragen

Welche Daten werden vom SE-7xx zur S7 übertragen (Statusdaten)?

- Boolsche Daten
  - ➔ Regelzonenstatus, Sollwertstatus, Istwertstatus, Toleranzstatus, Grenzwertstatus, Programmgeberstatus, Verfahrensschrittstatus, Digitalspurstatus, Digitale Ausgangsvariablen, Alarmhandlerstatus, Alarmstatus, Datenloggerstatus
- 32 Bit-Gleitkommawerte (REAL)
  - ➔ Regelzonenausgänge (Y-Werte), Sollwerte, Analogvariablen 1-40, Istwerte

Welche Daten werden von der S7 zum SE-7xx übertragen (Controldaten)?

- Boolsche Daten
  - ➔ Programmgebersteuerung, Regelzonensteuerung, Toleranzaktivierung, Digitale Eingangsvariablen, Alarmhandlersteuerung, Alarmeingänge, Datenloggersteuerung
- 32 Bit-Gleitkommawerte (REAL)
  - ➔ Analogvariablen 41-80, Istwerte (Istwert-Korrektur, Mittelwertberechnung u.a. konfigurierbar)

Lizenzierung überprüfen

Die korrekte Lizenzierung der Hardware-Option kann im SE-7xx überprüft werden.
Unter ***Konfiguration***, ***Hardware-Test***, ***Lizenz-Info***, ***Profinet IO-Device*** wird der aktuelle Lizenzstatus angezeigt.
Im Falle einer fehlenden Lizenz steht dieser Eintrag auf „Nein“ und es wird ein Lizenzalarm ausgegeben.

Hardware-Option im SE-7xx konfigurieren

Als erstes wird die Hardware-Option im SE-7xx konfiguriert. Dies geschieht unter ***Konfiguration***, ***Hardware***, ***Hardware-Optionen***. Dort wird der Einstellpunkt ***Feldbus-Modul*** auf ***Profinet IO-Device*** gestellt.
Im jetzt wählbaren Unterpunkt ***Profinet IO-Device*** kann der ***Profinet-Name*** eingestellt werden, standardmäßig ***nj100repns***.

Einschränkungen bei der Wahl des Profinet-Gerätenamens

Beachten Sie bitte die Regeln der Profinet-Namenskonvention. Folgende Regeln bei der Wahl des Profinet-Gerätenamens müssen eingehalten werden, damit die Kommunikation stattfinden kann:

- Beschränkung auf 127 Zeichen insgesamt (Buchstaben "a" bis "z", Ziffern "0" bis "9", Bindestrich oder Punkt)
- Ein Namensbestandteil innerhalb des Gerätenamens, d. h. eine Zeichenkette zwischen zwei Punkten, darf max. 63 Zeichen lang sein
- Keine Sonderzeichen wie Umlaute, Klammern, Unterstrich, Schrägstrich, Leerzeichen etc. Der Bindestrich ist das einzige erlaubte Sonderzeichen
- Im Gerätenamen dürfen keine Großbuchstaben verwendet werden
- Der Gerätename darf nicht mit den Zeichen "-" oder "." beginnen und auch nicht mit diesem Zeichen enden
- Der Gerätename darf nicht mit Ziffern beginnen
- Der Gerätename darf nicht die Form n.n.n.n haben (n = 0...999)
- Der Gerätename darf nicht mit der Zeichenfolge "port-xyz-" beginnen (x,y,z = 0...9)

(Quelle: https://support.industry.siemens.com)

Datenlogger-Konfiguration (SPS-Anweisungsliste)

Damit der Datenlogger ordnungsgemäß mit der S7-Schnittstelle funktioniert, müssen in der SPS-Anweisungsliste des SE-7xx folgende beiden Zeilen vorhanden sein:

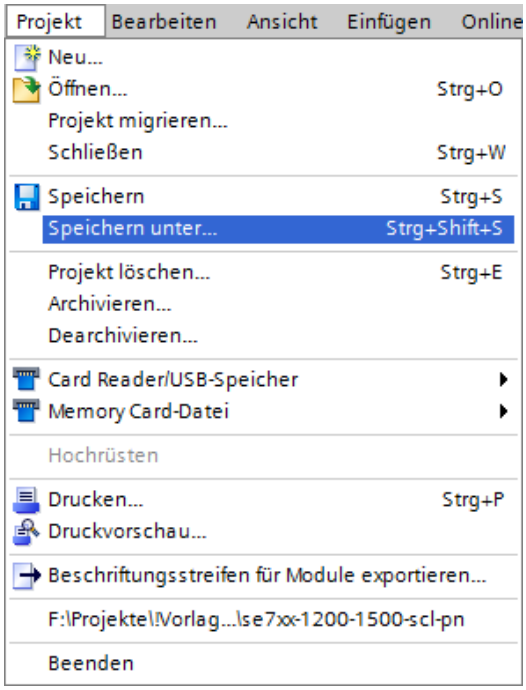
```
L FA 768
R FA 1311
```

Die SPS-Anweisungsliste findet sich unter ***Konfiguration***, ***Funktionen***, ***SPS-Anweisungsliste***. Nach dem Hinzufügen beider Zeilen wird die Änderung durch Tippen von „Übernahme“ übernommen und durch Tippen von „Zurück“ gespeichert.

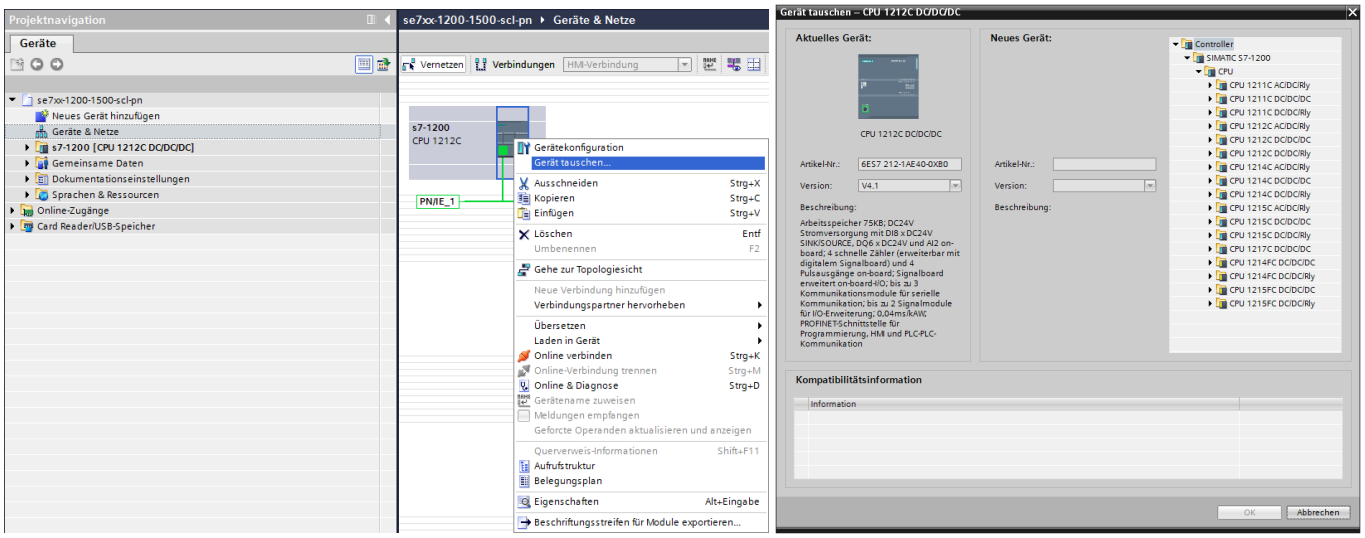
Weitere Informationen zur Konfiguration des Datenloggers siehe entsprechende Dokumentation.

Projekt als Vorlage verwenden

Das Projekt *se7xx-1200-1500-scl-pn* kann als Vorlage verwendet werden. Es wird in das TIA Portal geladen und kann dann direkt über *Projekt > Speichern unter* als Kopie unter einem neuen Namen gespeichert werden. Dies ermöglicht es, die Vorlage wieder zu verwenden.

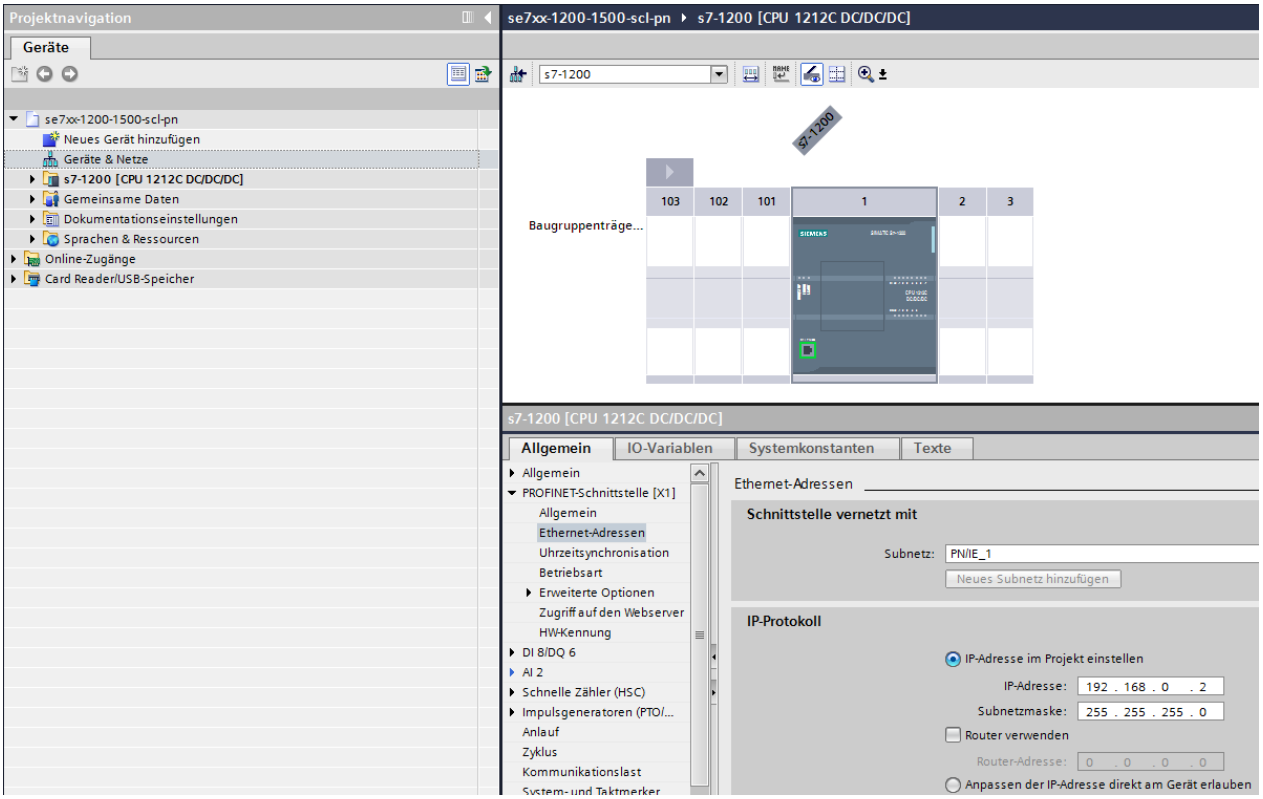


Im Vorlageprojekt sind jeweils eine S7-1212C DC/DC/DC und eine S7-1513-1 PN projektiert. Beide enthalten die gleichen Bausteine. Wenn eine andere SPS als die S7-1212C DC/DC/DC oder S7-1513-1 PN verwendet wird, muss die Projektierung angepasst werden. Dazu wird unter Geräte & Netze die S7 mit Rechts angeklickt und *Gerät tauschen* ausgewählt. Nun kann das verbaute Gerät gewählt werden. Die jeweils nicht benötigte S7 kann dann mittels Rechtsklick gelöscht werden.



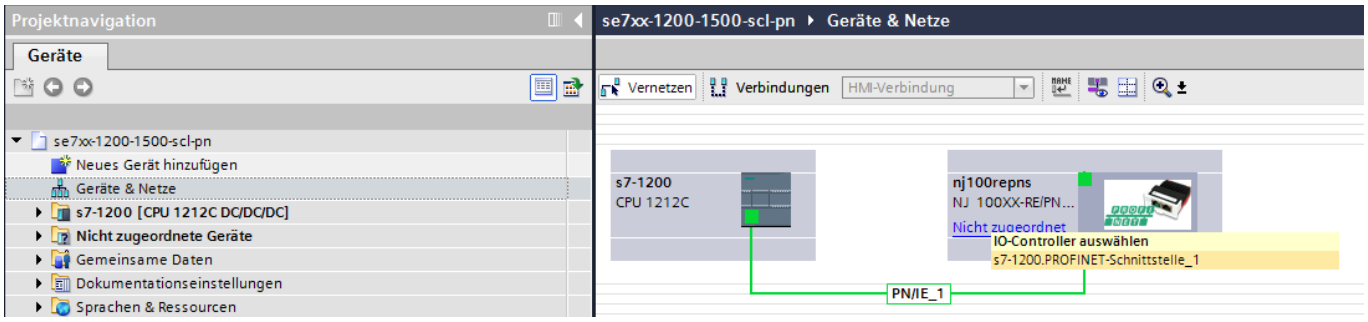
Um die IP-Adresse der S7 zu ändern, wird unter ***Geräte & Netze*** die S7 doppelt angeklickt und in der Baugruppenansicht nochmal doppelt angeklickt. Unter dem Eintrag ***PROFINET-Schnittstelle*** können die IP-Adresse, Subnetzmaske und ggf. Router-Adresse gesetzt werden.

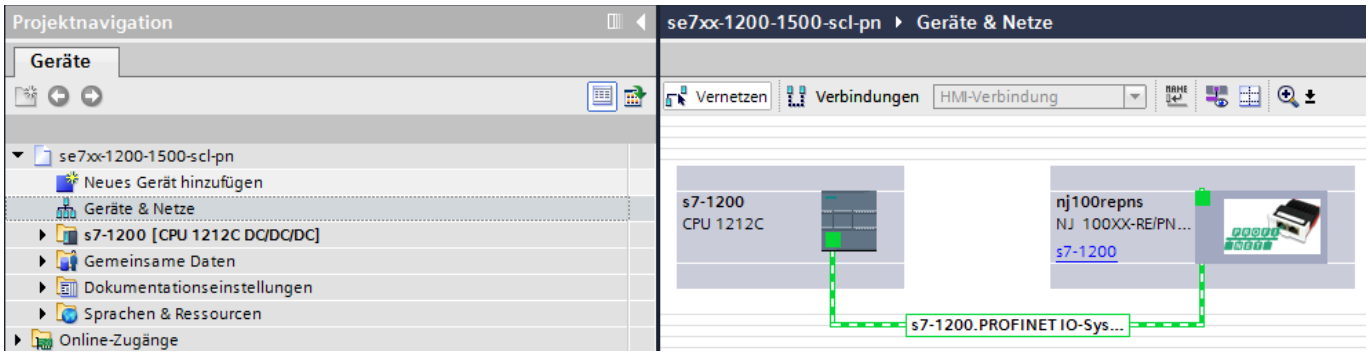
Die S7 kann dann über „Subnetz“ einem bestehenden Subnetz zugeordnet oder mit einem Klick auf „Neues Subnetz hinzufügen“ in ein neues Subnetz eingebunden werden. Dieser Schritt ist notwendig, damit sich das TIA Portal mit der S7 verbinden kann.



Profinet IO-Controller (Master) auswählen

Zuletzt muss dem NetJack 100 ein Profinet IO-Controller (Master) zugewiesen werden, damit die Kommunikation stattfinden kann. Dazu wird unter ***Geräte & Netze*** die ***Netzsicht*** ausgewählt. Nun kann durch Klicken auf *„Nicht zugeordnet“* am NetJack 100 der Master ausgewählt werden, also die zu verwendende S7. In diesem Beispiel ist das die S7-1212C.

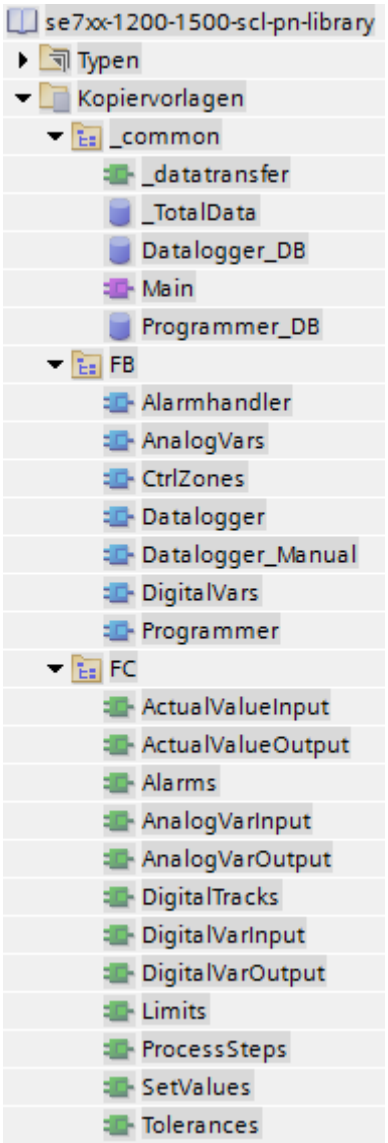




Die projektierte S7 muss anschließend übersetzt und die Hardwarekonfiguration und die Bausteine auf die S7 geladen werden. Im Fehlerfall blinkt die rote Error-LED der S7 und die rote BF-LED am NetJack 100. Die häufigste Fehlerursache ist ein im SE-7xx konfigurierter Profinet-Name, der nicht mit der Projektierung in TIA Portal übereinstimmt. Im Vorlageprojekt ist der Standardname ***nj100repns*** eingestellt. Dieser muss dann gegebenenfalls in den Eigenschaften geändert werden, damit die Profinet-Namen im SE-7xx und im Projekt gleich lauten. Anschließend muss erneut übersetzt und geladen werden.

Bibliotheksbausteine in bereits bestehendem Projekt verwenden

Wenn bereits ein S7-Projekt in TIA Portal existiert und dieses lediglich um die Profinet-Kommunikation erweitert werden soll, dann kann auf die Bibliothek ***se7xx-1200-1500-scl-pn-library*** zurückgegriffen werden.
Das folgende Bild zeigt eine Übersicht der enthaltenen Bausteine:



Damit die Profinet-Kommunikation funktioniert, werden mindestens folgende Bausteine benötigt. Diese müssen in das Projekt unter ***Programmbausteine*** kopiert werden:

- **_datatransfer**
- **_TotalData**
- **Datalogger** [FB114] und ***Datalogger_DB*** (bei Verwendung des Datenloggers)

Die restlichen FC/FB können je nach Bedarf in das Projekt eingebunden werden. Sie funktionieren aber nur, wenn sich auch die oben beschriebenen Bausteine im Projekt befinden. ***_datatransfer*** ist der Baustein, der die eigentliche Kommunikation der beiden Geräte durchführt. Er muss über OB1 (Main) eingebunden werden. Ein Beispiel-OB1 ist in der Bibliothek enthalten.

Bei aktiviertem Datenlogger muss der FB ***Datalogger*** muss über den OB1 eingebunden und mit einem IDB ausgestattet sein. Dieser Schritt ist zwingend notwendig, da der Datenlogger sonst nicht arbeitet. Für volle Flexibilität kann stattdessen auch ***Datalogger_Manual*** benutzt werden (siehe entsprechendes Kapitel).

Wenn im Projekt der Datenlogger-Baustein und der Programmgeber-Baustein verwendet werden, muss der Eingang ***ProgStart*** des Programmgebers (***Programmer***) mit dem Bit ***Start_Programmer*** des Datenlogger-IDBs über eine ODER-Verknüpfung beschaltet sein, ansonsten läuft der Programmgeber nicht an. Dies gilt nur, wenn ***ProgStart*** bereits beschaltet ist. Ansonsten kann ***ProgStart*** unbeschaltet bleiben.

Um Probleme bei Verwendung des Programmgeber-Bausteins und des Datenlogger-Bausteins zu vermeiden, wird empfohlen, den Programmgeber-Baustein vor dem Datenlogger-Baustein aufzurufen. Ansonsten könnte der Programmgeber nicht anlaufen. Außerdem sollten der Programmgeber-Baustein und der Datenlogger-Baustein nur einmal eingefügt werden. Um Störungen zu vermeiden, sollte nur einer der beiden Datenlogger-FB im Programm verwendet werden.

Wird der Datenlogger nicht verwendet, muss ***Datalogger_DB*** und der FB ***Datalogger*** (oder ***Datalogger_Manual***) nicht eingebunden werden.

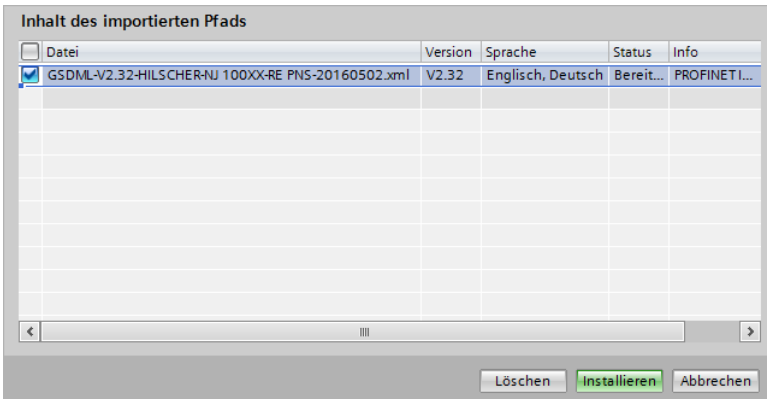
Die Bausteine müssen anschließend übersetzt und auf die S7 geladen werden.

Sollte beim Übersetzen der Fehler „Netzwerk 1: Der Operand ***se7xx_error*** ist nicht definiert“ auftreten, so muss im Baustein ***Main*** [OB1] im Netzwerk 1 (***_datatransfer***) mit Rechtsklick auf ***se7xx_error*** eine Variable definiert werden. Danach müssen die Bausteine erneut übersetzt und auf die S7 geladen werden.

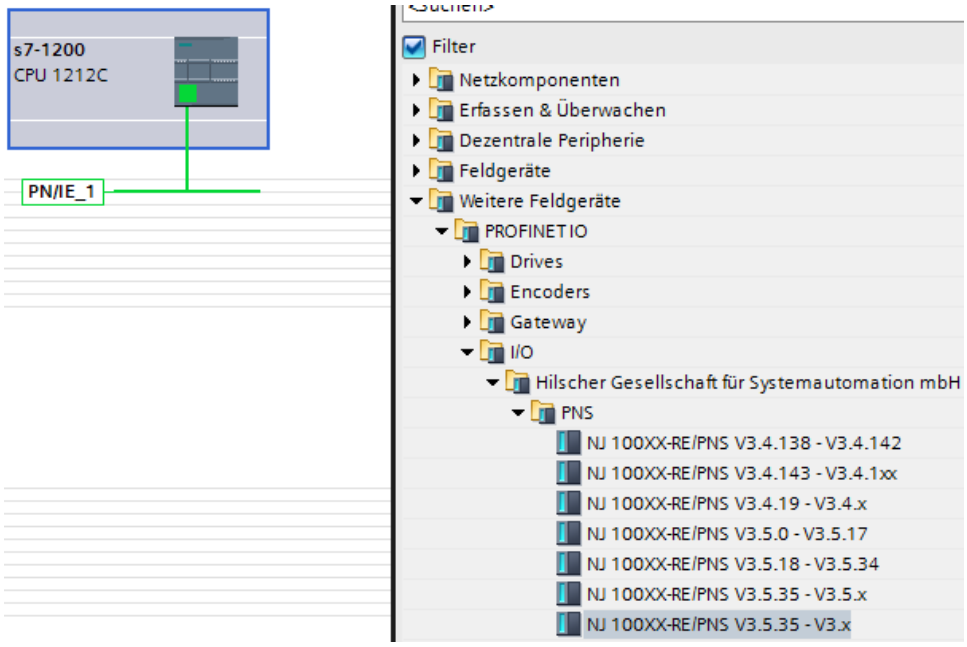
Einbinden der GSDML-Datei

Damit der Hilscher NetJack 100 in das Projekt eingebunden werden kann, muss die zugehörige GSDML-Datei importiert werden. Dies geschieht über ***Extras*** und ***Gerätebeschreibungsdateien (GSD) verwalten***:

Im folgenden Fenster wird der Speicherort der GSDML-Datei angegeben und die GSDML-Datei installiert:



Nach der Installation kann das Fenster geschlossen werden. Der NetJack 100 ist jetzt im Hardware-Katalog verfügbar.

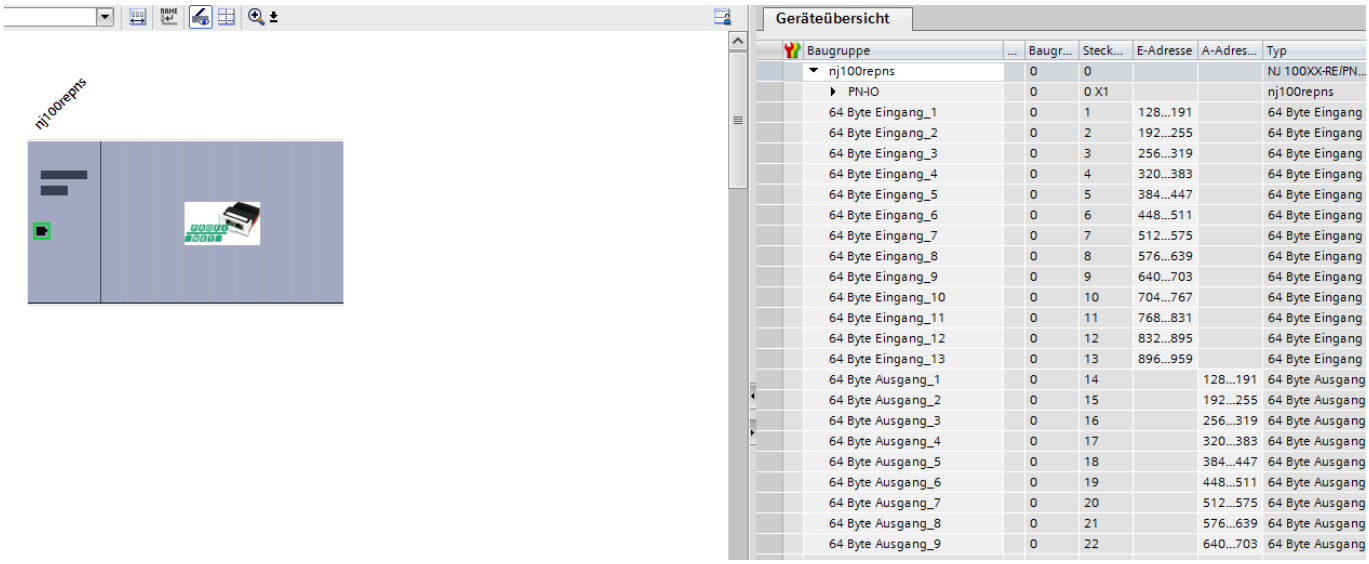


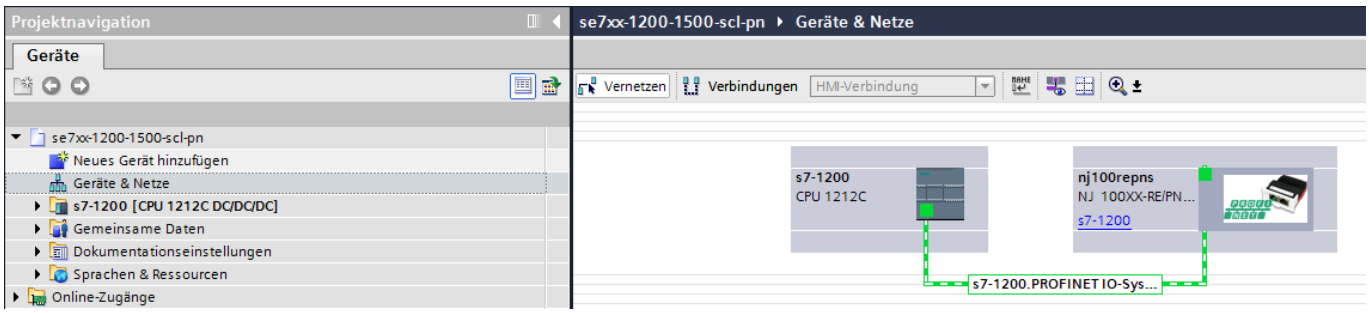
Per Drag & Drop werden die grünen Kästchen der S7 und des NetJack 100 (in der Netzsicht) verbunden, alternativ kann in den Eigenschaften des NetJack 100 das Subnetz ausgewählt werden. Dort kann auch die IP-Adresse eingestellt werden. Anschließend muss dem NetJack 100 ein IO-Controller (Master) zugewiesen werden (siehe entsprechender Abschnitt).

Modulkonfiguration

Für den Datenaustausch zwischen dem SE-7xx (NetJack 100) und der S7 müssen Profinet-Module projektiert werden. Dazu werden 13 Eingangsmodule mit je 64 Byte und 9 Ausgangsmodule mit je 64 Byte benötigt. Diese werden über die mitgelieferte GSDML-Datei in die Konfiguration unter ***Geräte & Netze*** gezogen. Im Vorlageprojekt ist dies bereits geschehen.

Über diese Module findet der gesamte Datenaustausch statt. Es ist nicht vorgesehen und auch nicht notwendig, den Umfang der ausgetauschten Daten und damit die Anzahl der benötigten Module von Hand einzustellen. Die Zuordnung der Module zu den Daten im SE-7xx erfolgt automatisch nach einer festen Struktur.





Im obigen Beispiel wurden nur die Profinet-Module projektiert, die für den Datenaustausch zwischen SE-7xx (NetJack 100) und der S7 benötigt werden. Die Eingangsmodule und die Ausgangsmodule müssen jeweils unbedingt zusammenhängend projektiert werden, um eine korrekte Datenübertragung zu gewährleisten.

Es ist aber möglich, die Gruppe der Eingangsmodule und die Gruppe der Ausgangsmodule mit einem jeweils unterschiedlichen Byteoffset zu versehen. Zum Beispiel könnten die 13 Eingangsmodule bei Adresse 125.0 starten (***pn_inputoffset*** = 125) und die 9 Ausgangsmodule bei Adresse 210.0 (***pn_outputoffset*** = 210). Diese Offsets werden im nächsten Schritt benötigt.

Die S7-Hardwarekonfiguration und die Bausteine müssen anschließend übersetzt und auf die S7 geladen werden.
Am einfachsten ist es, wenn man die S7 in der Projektnavigation (links) markiert und nacheinander die Buttons „Übersetzen“ und „Laden“ in der Symbolleiste von TIA Portal wählt. So wird sowohl die Hardwarekonfiguration als auch die Software (Programmbausteine) übersetzt und übertragen.

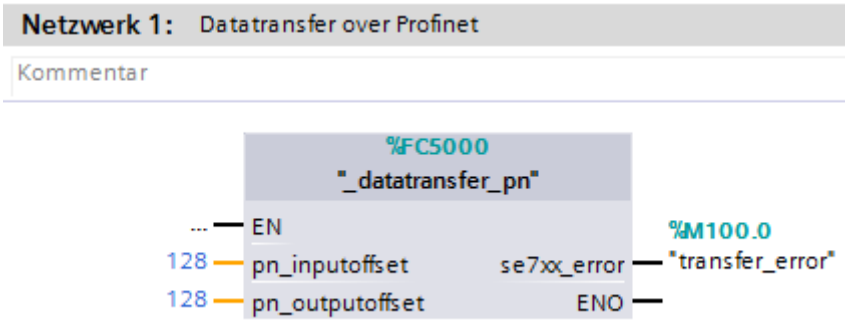
Sollte beim Übersetzen der Fehler „Netzwerk 1: Der Operand ***transfer_error*** ist nicht definiert“ auftreten, so muss im Baustein ***Main*** [OB1] im Netzwerk 1 (***_datatransfer***) mit Rechtsklick auf ***transfer_error*** eine Variable definiert werden. Danach müssen die Bausteine erneut übersetzt und auf die S7 geladen werden.

Beschreibung der Funktionalität

Allgemeines

Der wichtigste Baustein ist ***Main*** [OB1]. Sein Inhalt wird zyklisch aufgerufen und enthält mindestens ***_datatransfer*** [FC5000] und bei Verwendung des Datenloggers ***Datalogger*** [FB114] oder ***Datalogger_Manual*** [FB115]. ***_datatransfer*** steuert den allgemeinen Datenaustausch zwischen der S7 und dem SE-7xx. Ohne diesen FC ist keine Profinet-Kommunikation möglich.

_datatransfer [FC5000]	
Parameter	**Beschreibung**
pn_inputoffset	Byteoffset der Profinet-Eingangsmodule (status)
pn_outputoffset	Byteoffset der Profinet-Ausgangsmodule (control)
se7xx_error	Fehlerausgabe des Watchdogs



Über die beiden Eingangsparameter ***pn_inputoffset*** und ***pn_outputoffset*** werden die beiden Byteoffsets der Profinet-Module angegeben (siehe Abschnitt ***Modulkonfiguration***). Im obigen Beispiel beginnen die Eingangsmodule für den NetJack 100 bei Eingang 128.0 und die Ausgangsmodule bei Ausgang 128.0. Diese Angaben sind wichtig, damit die Bausteine im richtigen Bereich Daten lesen und schreiben können. Die Eingangsmodule übertragen die Daten vom SE-7xx zur S7, die Ausgangsmodule in Gegenrichtung.

Am Ausgang ***se7xx_error*** wird gegebenenfalls ein Fehler ausgegeben, wenn der Verbindungs-Watchdog auslöst. Dieser Ausgang kann auf einen Merker oder eine DB-Variable gelegt werden.

Die einzelnen Bausteine entsprechen den Funktionen des STANGE-Gerätes. Sie können einfach in den OB1 oder einen selbst erstellten FC/FB gezogen werden. Diese werden dann über ihre Eingänge/Ausgänge in den Programmablauf eingebunden.

Als InstanceNo wird die Nummer der Instanz beschrieben, beispielsweise Digitalspur 4 oder Grenzwert 2. Es erfolgt dabei ein Grenzen-Check, d.h. bei einer InstanceNo außerhalb des gültigen Bereichs (etwa Sollwert 23 bei maximal 20 möglichen) wird der Wert auf die technisch maximal mögliche Instanz gesetzt; bei Werten kleiner/gleich 0 wird Instanz 1 gewählt.

Die Anzahl der einfügbaren Bausteine ist nicht begrenzt. Für jeden eingefügten FB wird ein separater IDB (Instanz-DB) erstellt. Nicht genutzte Eingänge/Ausgänge bei FCs können auf einen nicht verwendeten Merker oder Variable in einem DB gesetzt werden. Die Reihenfolge von Instanzen eines FC/FB ist egal; jeder neue Aufruf eines Bausteins mit bereits benutzter Instanzennummer überschreibt aber jeden vorherigen Aufruf dieses Bausteins mit dieser Instanz. Nicht benutzte Bausteine können über den Eingang EN deaktiviert werden.

Übersicht FCs/FBs

Name	Baustein	Funktion
SetValues	FC101	Sollwert und -Status abfragen
Alarms	FC103	Alarm generieren, Alarmstatus abfragen
ProcessSteps	FC104	Verfahrensschritt-Status abfragen
DigitalTracks	FC105	Digitalspur-Status abfragen
Tolerances	FC106	Toleranz-Status abfragen, Toleranz extern aktivieren
Limits	FC107	Grenzwert-Status abfragen
DigitalVarInput	FC108	Digitalvariable im SE-7xx setzen (ab FE 2000)
DigitalVarOutput	FC109	Digitalvariable vom SE-7xx lesen (ab FA 2000)
AnalogVarInput	FC110	Analogvariable im SE-7xx setzen (Werte 41-80)
AnalogVarOutput	FC111	Analogvariable vom SE-7xx lesen (Werte 1-40)
ActualValueInput	FC112	Istwert im SE-7xx setzen, Overflow/Underflow/Break erzeugen
ActualValueOutput	FC113	Istwert vom SE-7xx lesen, Istwert-Fehlerstatus abfragen
Programmer	FB100	Programmgeber steuern und abfragen
CtrlZones	FB102	Regelzone steuern und abfragen
Alarmhandler	FB103	Alarmhandler steuern und abfragen
DigitalVars	FB108	Mehrere Digitalvariablen gleichzeitig setzen und lesen
AnalogVars	FB110	Mehrere Analogvariablen gleichzeitig setzen und lesen
Datalogger	FB114	Datenlogger steuern und abfragen (Automatikmodus)
Datalogger_Manual	FB115	Datenlogger steuern und abfragen (manuell)

Allgemeiner Aufbau FC/FB

Inputs: InstanceNo [Instanznummer] und jeweilige Funktionseingänge

Outputs: Funktionsausgänge

Temp: instno_tmp: Kopie von InstanceNo; verwendet für Grenzencheck

Constant: entries: enthält maximale Anzahl an Instanzen; verwendet für Grenzencheck

FC5000: _datatransfer und DB5000: _TotalData

Der FC5000 *_datatransfer* ist für das Senden und Empfangen der Daten über Profinet zuständig. Dazu werden für jeden Datenbereich mittels POKE_BLK die Daten zwischen den E/A-Modulen und *_TotalData* übertragen. Außerdem geschieht hier die Überwachung der Verbindung mittels eines Watchdogs. Nach 10 Zyklen ohne Antwort wird der Fehlerausgang aktiviert.

```
// 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;

```

Lokal werden die Daten im DB *_TotalData* zwischengespeichert. Auf diesen DB greifen die FC/FB letztendlich zu. Daten aus *control* werden zum SE-7xx geschickt, Daten vom SE-7xx in *status* gespeichert. Es wird geraten, keine direkten Verdrahtungen/Verschaltungen mit DB-Einträgen durchzuführen, sondern stattdessen die Schnittstellen der FC/FB zu verwenden.

Die interne Zuordnung der einzelnen Profinet-Adressen zu den Funktionen des SE-7xx geschieht vollautomatisch, sodass hier keine Einstellung oder Konfiguration nötig ist.

_TotalData

	Name	Datentyp	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.

Enthält alle Bools, die zum SE-7xx gesendet werden sollen. Diese Bits werden durch die FC/FB gesetzt.

_TotalData.status.booldata

Array [0..1375] of Bool.

Enthält alle Bools, die vom SE-7xx empfangen wurden. Diese Bits werden durch die FC/FB ausgelesen.

_TotalData.control.actvalues/analogvars

und

_TotalData.status.actvalues/setvalues/yvalues/analogvars

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

Enthält 32-Bit-Istwerte(/Sollwerte/Y-Werte)/Analogvariablen.

Die zu sendenden Istwerte dürfen im SE-7xx nicht als unbelegt konfiguriert sein.
Die gesendeten Analogvariablen 1-40 werden im SE-7xx als Analogvariablen 41-80 abgebildet.

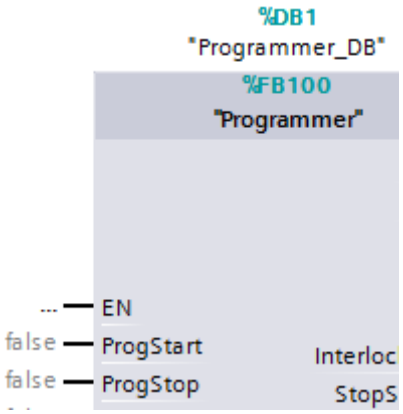
FB100: Programmer und FB114: Datalogger

Der Datenlogger des SE-7xx funktioniert bei eingeschalteter S7-Schnittstelle nur, wenn der ***Datalogger***-Baustein über OB1 eingebunden wird. Er enthält die Logik zur Ablaufsteuerung des Loggers. Die Eingänge und Ausgänge des Bausteins können gegebenenfalls in den S7-Programmablauf eingebunden werden, dies ist aber nicht notwendig.

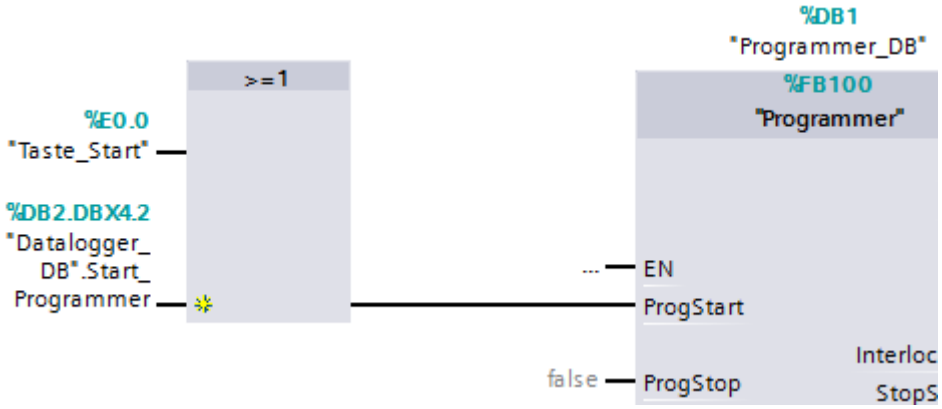
Um den Datenlogger (und den Programmgeber) über die S7 zu starten, genügt es, einen Impuls an den Eingang ***ProcessStart*** anzulegen. Nach fünf Sekunden wird der Programmgeber automatisch gestartet. Sobald das Programm beendet ist oder der Bediener END oder RESET auswählt, stoppt der Logger automatisch. Die aufgezeichneten Daten sind dann in der Logliste des SE-7xx aufrufbar. Bei den Chargendetails wird als Benutzer „plc“ angegeben, wenn die Chargenaufzeichnung über die S7 gestartet wurde, ansonsten der Name des aktuell eingeloggten Benutzers.

Der Eingang ***ProgStart*** am FB ***Programmer*** startet nur den Programmgeber selbst – ohne Datenlogger.

Da ein eventuell vorhandener Programmgeber-Baustein im S7-Projekt bei einer Beschaltung an ***ProgStart*** das Signal zum Starten des Programmgebers überschreiben könnte, sollte an diesem Eingang mittels einer ODER-Verknüpfung die Variable ***Start_Programmer*** des Instanz-Datenbausteins vom FB ***Datalogger*** verdrahtet werden. Wird der Eingang ***ProgStart*** nicht verwendet, so muss die Verdrahtung nicht vorgenommen werden.



Programmgeber ohne Verdrahtung an ProgStart



Programmgeber mit Verdrahtung an ProgStart

FB114: Datalogger und FB115: Datalogger_Manual

Der FB ***Datalogger*** arbeitet im „Automatikmodus“. Das bedeutet, er enthält die nötige Logik, um das Starten des Programmgebers über die Oberfläche des SE-7xx zu erkennen und den Datenlogger und den Programmgeber schließlich automatisch zu starten. Dies ist nötig, da durch die Profinet-Anbindung die SPS-Zeilen im SE-7xx entfallen.
Dazu muss der FB nur über OB1 aufgerufen werden; Verschaltungen der Eingänge/Ausgänge sind nicht nötig. Bei Bedarf kann der Datenlogger zusammen mit dem Programmgeber auch von der S7 gestartet werden. Normalerweise ist diese Funktionalität völlig ausreichend für den Anwendungsfall.

Möchte man jedoch volle Flexibilität bei der Datenloggeransteuerung haben, kann ***Datalogger_Manual*** verwendet werden. Er enthält keine Logik, sondern bietet stattdessen völlige Freiheit bei der Verarbeitung der Steuer- und Statussignale.
Um Störungen zu vermeiden, sollte nur einer der beiden FB im Programm verwendet werden.

Wenn der Datenlogger in der Konfiguration des SE-7xx eingeschaltet ist, erzeugt der Startbutton auf der Programmgeberseite nur eine Prozessstartanforderung (und startet noch nicht den Programmgeber). Diese Anforderung wird durch den Ausgang ***ProcessstartActive*** am FB angezeigt. Ebenso erzeugt der Eingang ***ProcessStart*** am Baustein eine Prozessstartanforderung. Die Anforderung setzt unter anderem den korrekten Benutzernamen in der Chargenliste im SE-7xx („plc“ falls von S7).

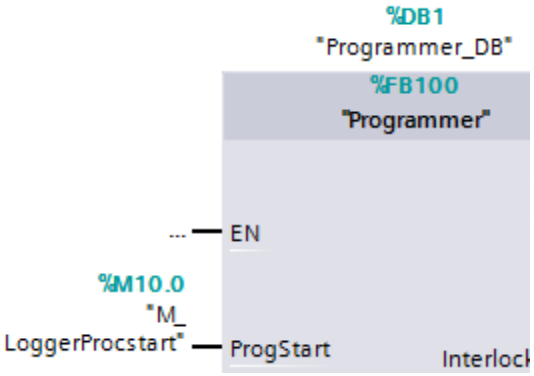
ProcessstartActive kann dann verwendet werden, um mittels ***LogStart*** den Datenlogger zu starten. Außerdem muss dann über den Eingang ***ProgStart*** von ***Programmer*** der Programmgeber gestartet werden. ***ProcessstartActive*** wird automatisch zurückgesetzt, sobald der Programmgeber läuft (das sind die zwei Zeilen in der SPS-Anweisungsliste des SE-7xx).

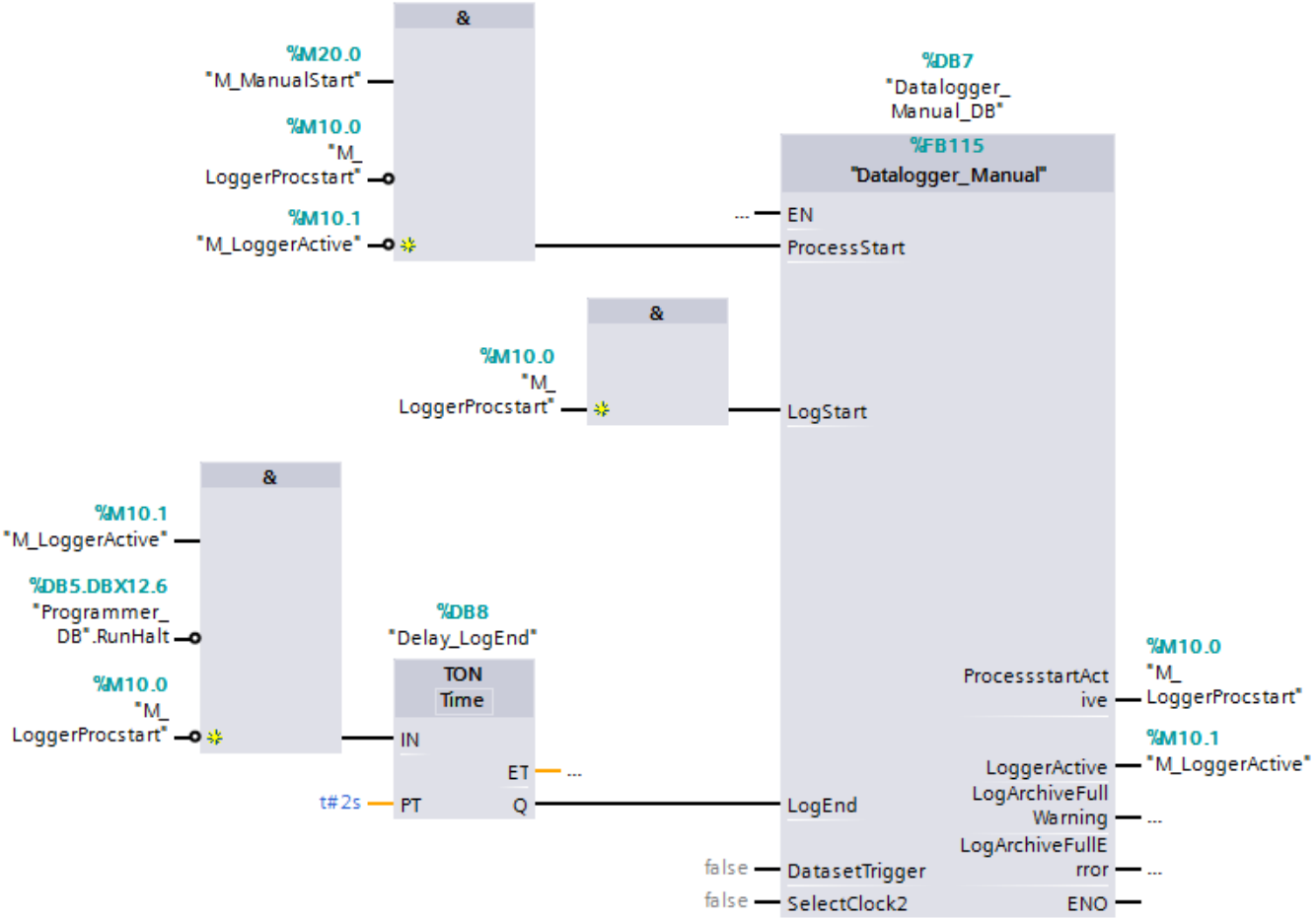
LoggerActive zeigt an, dass der Datenlogger läuft und gerade eine Charge aufzeichnet. Mittels ***LogEnd*** wird die Aufzeichnung der Charge schließlich beendet.

Auf der folgenden Seite ist eine Beispielansteuerung von ***Datalogger_Manual*** abgebildet.

Die Prozessstartanforderung kann lokal erzeugt werden (***M_ManualStart***); dazu darf keine Anforderung bestehen und der Datenlogger darf nicht aktiv sein. Die Anforderung kann auch vom SE-7xx erzeugt werden (Programmgeber).
Durch die Anforderung gibt ***ProcessstartActive*** ein Signal aus. Dieses Signal (gespeichert in ***M_LoggerProcstart***) kann dann dazu verwendet werden, um mittels ***LogStart*** die Aufzeichnung durch den Datenlogger zu starten. Der Merker ***M_LoggerActive*** zeigt dann den aktiven Datenlogger an.
Dieser Statusmerker wird dann dafür verwendet, um den Datenlogger abzuschalten, sobald der Programmgeber das Programmende erreicht hat oder der Programmgeber geresettet wird.

Das Starten des Datenloggers und des Programmgebers wird in diesem Beispiel einfach durch einen Impuls an ***M_ManualStart*** ausgelöst. Die Statusausgänge können nach Belieben im Programm ausgewertet werden.
Alternativ kann der FB ***Datalogger*** verwendet werden, der die gesamte Ansteuerungslogik bereits enthält.





FC108/FC109, FB108: DigitalVarInput, DigitalVarOutput, DigitalVars (Digitalvariablen)

Digitale Eingangsvariablen des SE-7xx können mit den Bausteinen ***DigitalVarInput*** und ***DigitalVars*** von der S7 gesetzt werden. Diese werden auf die Funktionseingänge (FE) 2000 bis 2199 des SE-7xx abgebildet und können zum Beispiel zur Statusanzeige in der Visualisierung genutzt werden.
Diese Variablen können von der S7 nur beschrieben und nicht gelesen werden.

Digitale Ausgangsvariablen des SE-7xx können mit den Bausteinen ***DigitalVarOutput*** und ***DigitalVars*** zur S7 übertragen werden. Diese befinden sich in den Funktionsausgängen (FA) 2000 bis 2199 des SE-7xx und können zum Beispiel von Tastern in der Visualisierung gesetzt werden.
Diese Variablen können von der S7 nur gelesen und nicht beschrieben werden.

FC110/FC111, FB110: AnalogVarInput, AnalogVarOutput, AnalogVars (Analogvariablen)

Analoge Eingangsvariablen des SE-7xx können mit den Bausteinen ***AnalogVarInput*** und ***AnalogVars*** von der S7 gesetzt werden. Diese werden auf die Analogvariablen 41-80 des SE-7xx abgebildet und können zum Beispiel zur Statusanzeige in der Visualisierung oder als Ersatz-Sollwert der Regelzone genutzt werden.
Diese Variablen können von der S7 nur beschrieben und nicht gelesen werden.

Analoge Ausgangsvariablen des SE-7xx können mit den Bausteinen ***AnalogVarOutput*** und ***AnalogVars*** zur S7 übertragen werden. Diese befinden sich in den Analogvariablen 1-40 des SE-7xx und können zum Beispiel durch Eingabefelder in der Visualisierung gesetzt werden.
Diese Variablen können von der S7 nur gelesen und nicht beschrieben werden.

Analogvariablen müssen vor ihrer Verwendung als IEEE-Float konfiguriert sein (Konfiguration im SE-7xx).

FC112/FC113: ActualValueInput, ActualValueOutput (Istwerte)

Istwerte des SE-7xx können mit dem Baustein ***ActualValueInput*** von der S7 gesetzt und zum Beispiel als Regelistwert verwendet werden. Diese können wie alle anderen Istwerte auch mit Istwert-Korrektur, Mittelwertberechnung u.a. konfiguriert werden.

Der jeweilige Istwert darf in der Konfiguration des SE-7xx nicht „unbelegt“ sein; es reicht, ihn als „linear“ zu definieren.

Spezialwerte gemäß IEEE 754 können im SE-7xx einen Istwertalarm auslösen:

Istwert	IEEE 754-Bezeichnung	Anzeige am 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

Mittels der Eingänge ***ForceOverflow***/***ForceUnderflow***/***ForceBreak*** kann der jeweilige Status im SE-7xx erzeugt werden.

Istwerte des SE-7xx können mit dem Baustein ***ActualValueOutput*** zur S7 übertragen werden. Zusätzlich wird am Ausgang ***ActValueError*** angegeben, ob dieser Istwert einen Fehler hat (zum Beispiel Bruch).

Gewusst wie

Sollwertvorgabe durch die S7

Obwohl im SE-7xx Programmabläufe definiert werden können, die einen bestimmten Sollwertverlauf ergeben, können diese Sollwerte auch extern durch eine S7 vorgegeben werden. Dadurch kommt die programmierte Sollwertkurve nicht mehr zur Anwendung. Der SE-7xx regelt dann mit diesen externen Sollwerten und ignoriert die internen vom Programmgeber.

Eine direkte Änderung der Programmgeber-Sollwerte ist aus technischen Gründen so nicht vorgesehen; dennoch ist es möglich, an den Sollwerten der dahintergelagerten Regelzonen anzusetzen. Die Idee hier ist, der entsprechenden Regelzone einen Ersatz-Sollwert bereitzustellen. Dieser Ersatz-Sollwert ergibt sich aus einer Analogvariable, die durch die S7 zum SE-7xx gesendet werden kann. Auf der S7-Seite geschieht dies einfach durch einen Baustein, an dem der Sollwert angegeben wird. Zusätzlich wird der Ersatz-Sollwert dauerhaft eingeschaltet.

Es ist hierfür eine einmalige Konfiguration im SE-7xx erforderlich. Die 80 Analogvariablen des SE-7xx wurden für die S7-Profinetschnittstelle aufgeteilt in 40 Lesewerte (1-40) und 40 Schreibwerte (41-80). Somit müssen in der Konfiguration des SE-7xx mindestens 41 Analogvariablen konfiguriert sein, damit die S7 mindestens eine Analogvariable schreiben kann.

➔ Konfiguration > Funktionen > Analogvariablen > Param. (Button auf der linken Seite)
Hier müssen mindestens **41** Werte eingestellt werden. Nach Tippen auf “Zurück” erscheint wieder die Variablenliste. Beispielsweise wird hier Analogvariable 41 konfiguriert.
Nach Wählen der Variable 41 und “Ändern” erscheint die Konfigurationsseite. Die Bezeichnung der Variable ermöglicht später ein einfaches Wiederfinden. Der Variablentyp muss auf **IEEE-Float** eingestellt werden. Das Anzeigeformat ermöglicht die Einstellung der Nachkommastellen (je mehr Nachkommastellen, desto weniger Stellen links vom Komma). Empfohlen werden maximal zwei Nachkommastellen. Weiter unten lassen sich Untergrenze und Obergrenze der Analogvariable – also des externen Sollwerts – einstellen. Hier kann der maximale Bereich von bspw. **-99999.9** bis **99999.9** ausgenutzt werden.
Die Leitsystem-Adresse ist für diesen Zweck nicht relevant. Der Initialisierungsmodus besagt, welchen Wert die Variable nach einem Reset haben soll und kann bei Bedarf konfiguriert werden. Der Standardwert hier ist **Keiner**.

Die Konfiguration der Analogvariable ist hiermit fertig.

➔ Konfiguration > Funktionen > Regelzonen
Hier werden alle Regelzonen aufgelistet. Die zu konfigurierende Regelzone wird markiert und rechts “Ändern” angetippt. Auf der Konfigurationsseite kann jetzt als Ersatz-Sollwerttyp **Variable** gewählt werden.
Die Ersatz-Sollwertnummer entspricht der Nummer der Analogvariable, die den Sollwert enthalten wird; also in diesem Beispiel **41**. Der Name der Analogvariable wird in Klammern angezeigt.

Die Konfiguration kann jetzt durch Verlassen gespeichert werden. Damit ist die Konfiguration im SE-7xx beendet.

In der S7 kann zum Senden der Analogvariable ***AnalogVarInput*** verwendet werden. Am Eingang ***Value*** wird der zu sendende Sollwert im Real-Format (Float) angegeben. ***InstanceNo*** = **1** entspricht dann der Analogvariable 41 im SE-7xx, die oben als Ersatz-Sollwert konfiguriert wurde. Analog dazu lassen sich auch weitere Ersatz-Sollwerte definieren, die dann als Analogvariablen 42 usw. zum SE-7xx übertragen werden.

Mit dem Baustein ***CtrlZones*** wird die Verwendung des Ersatz-Sollwerts an der gewählten Regelzone konfiguriert. ***InstanceNo*** gibt die Nummer der Regelzone an (z.B. **1**). Durch Setzen von **true** am Eingang ***EnableSubstSV*** wird schließlich der im SE-7xx konfigurierte Ersatz-Sollwert aktiviert. Der aktuelle Sollwert kann dann auf der Reglerseite angezeigt werden.

Der SE-7xx regelt unabhängig vom Status des Programmgebers. Am Eingang ***Disable*** von ***CtrlZones*** kann ggf. die Regelzone deaktiviert werden, also der Y-Regelausgang auf 0.0 gesetzt werden.

Um ein Start/Stop-Signal vom SE-7xx zu erhalten, kann über die Visu ein Button definiert werden, dessen Status über ***DigitalVarOutput*** ausgelesen werden kann. Die andere Möglichkeit wäre, ein Pseudo-Programmrezept zu erstellen, um so wie gewohnt den Programmgeber starten und stoppen zu können. Dies hätte dann auch den Vorteil, dass der integrierte Datenlogger dann Chargenaufzeichnungen erstellen könnte.

Interface der Bausteine (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 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 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 to use the datalogger in automatic mode (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 to use the datalogger in manual mode (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