Dokumentation

Modbus/TCP – Anbindung S7-1200/1500 ↔ SE-7xx

an die neue S7-Modbusschnittstelle (Port 21303)

Stand der Dokumentation: 11.05.2021
gültig für: Version 1.4

Autor: Lukas Jolbej

In dieser Dokumentation wird die Anbindung einer S7 an die S7-Modbusschnittstelle des Stange SE-7xx erläutert.

Verwendete Geräte:

- Stange SE-702
- Siemens S7-1212C DC/DC/DC (6ES7212-1AE40-0XB0)

verbunden über einen 100 MBit/s-Switch

Verwendete Software:

- Siemens TIA Portal V15
- Windows 7 SP1
- Geräteversion 7.0.2.10 für den SE-702
- Firmwareversion 4.1 für die S7-1212C

Voraussetzungen:

- Die Funktion muss im SE-7xx lizenziert sein
- IP-Adresse und Modbus Unit ID des SE-7xx müssen bekannt sein
- SE-7xx und S7 befinden sich im selben Netzwerk-Subnetz

Zugehörige TIA Portal-Vorlagen:

- se7xx-1200-1500-scl-mb (Vorlageprojekt), Version 1.4
- se7xx-1200-1500-scl-mb-library (Bibliothek), Version 1.4

**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 V15 erstellt,
können aber bei Bedarf auf eine aktuelle Version hochgerüstet werden.**

Inhaltsverzeichnis

Erste Schritte

Funktionsübersicht

Funktionsweise

Die S7-Modbusschnittstelle der Serie SE-7xx am Port 21303 ist eine lizenzierbare Erweiterung der standardmäßig vorhandenen Modbusschnittstelle am Port 502. Beide Schnittstellen ermöglichen lesenden und schreibenden Zugriff auf Daten im SE-7xx per Modbus TCP. Unterschiede gibt es in der Datenmenge sowie der Datenaufbereitung (Zuordnung zu Adressen/Registern sowie S7-optimierte Bytereihenfolge). Die Bausteine sind daher nur für die S7-Modbusschnittstelle am Port 21303 einsetzbar. Nicht alle Daten, die am Port 502 verfügbar sind, sind auch am Port 21303 vorhanden.

Grundsätzlich ändert sich das Verhalten des SE-7xx durch das Einschalten der S7-Modbusschnittstelle so, dass die digitalen Steuersignale der S7 Vorrang haben vor entsprechenden Einträgen in der SPS-Anweisungsliste des SE-7xx. Daher ist die gesamte digitale Logik in der S7 zu realisieren. Es werden immer alle Datenbereiche zum SE-7xx geschrieben bzw. überschrieben. Bei den Istwerten ist zu beachten, dass Werte der CAN-IO (SIOS bzw. CAN-Basis) grundsätzlich Vorrang haben.

Die S7-Modbusschnittstelle ist vom Datenumfang und der Datenstruktur her identisch zur S7-Profinetschnittstelle (Variante mit festem Datenmapping). In beiden Fällen wird das vergrößerte Mengengerüst (ab Geräteversion 7.0.3.x) nicht unterstützt. Die S7-Modbusschnittstelle (Port 21303) und die S7-Profinetschnittstelle dürfen nicht gleichzeitig aktiviert werden, da sie auf denselben Datenbereich zugreifen und sich gegenseitig überschreiben würden. Beide Schnittstellen erfordern Lizenzen.

➔ siehe Dokument: **SE-7xx – Daten der S7-Schnittstelle.pdf**

Was kann man mit den Bausteinen machen?

- Programmgeber steuern und abfragen
- Regelzone steuern und abfragen
- Sollwert und Sollwertstatus 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 (FE 2000-2199)
- Digitalvariable vom SE-7xx lesen (FA 2000-2199)
- 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 ➔ Programmgeberstatus, Regelzonenstatus, Sollwertstatus, Istwertstatus, Toleranzstatus, Grenzwertstatus, Alarmstatus, Alarmhandlerstatus, Verfahrensschrittstatus, Digitalspurstatus,
  Digitale Ausgangsvariablen (FA 2000-2199), 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, Alarmeingänge, Alarmhandlersteuerung, Digitale Eingangsvariablen (FE 2000-2199), Datenloggersteuerung
- 32 Bit-Gleitkommawerte (REAL) ➔ Analogvariablen 41-80, Istwerte

Lizenzierung überprüfen

Die korrekte Lizenzierung der S7-Modbusschnittstelle kann im SE-7xx überprüft werden.
Unter ***Konfiguration*** > ***Hardware-Test*** > ***Lizenz-Info*** > ***Siemens-Modbusanbindung*** wird der aktuelle Lizenzstatus angezeigt.
Im Falle einer fehlenden Lizenz steht dieser Eintrag auf „Nein“ und es wird ein Lizenzalarm ausgegeben.

S7-Modbusschnittstelle im SE-7xx einschalten

Die S7-Modbusschnittstelle muss im SE-7xx eingeschaltet sein. Dies geschieht unter ***Konfiguration*** > ***Grundeinstellungen*** > ***Siemens-Modbusanbindung***. Dort wird der Einstellpunkt ***Modbus Freigabe*** auf ***Freigegeben*** gestellt.

Es ist immer nur eine 1:1-Datenübertragung möglich. Das heißt, ein SE-7xx kann nicht mit mehreren S7 verbunden sein und Daten austauschen. Ebenso kann eine S7 nur mit einem SE-7xx Daten austauschen.[1]

Datenlogger-Konfiguration (SPS-Anweisungsliste)

Damit der Datenlogger ordnungsgemäß mit der S7-Modbusschnittstelle 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.

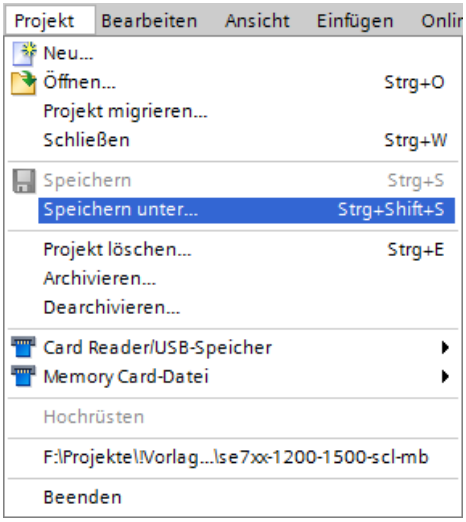
Für den grundlegenden Betrieb der S7-Modbusschnittstelle werden nur diese beiden Zeilen benötigt. Die werksmäßig vorhandenen Zeilen können daher gelöscht werden, sofern keine kunden- oder anlagenspezifische Konfiguration durch den Anlagenhersteller bzw. durch Stange durchgeführt wurde.
Wurde eine solche Konfiguration durchgeführt, dürfen die vorhandenen Anweisungszeilen nicht gelöscht werden. Es ist dann nur darauf zu achten, dass die beiden oben erwähnten Zeilen vorhanden sind.

Weitere Informationen zur Konfiguration des Datenloggers siehe entsprechende Dokumentation.

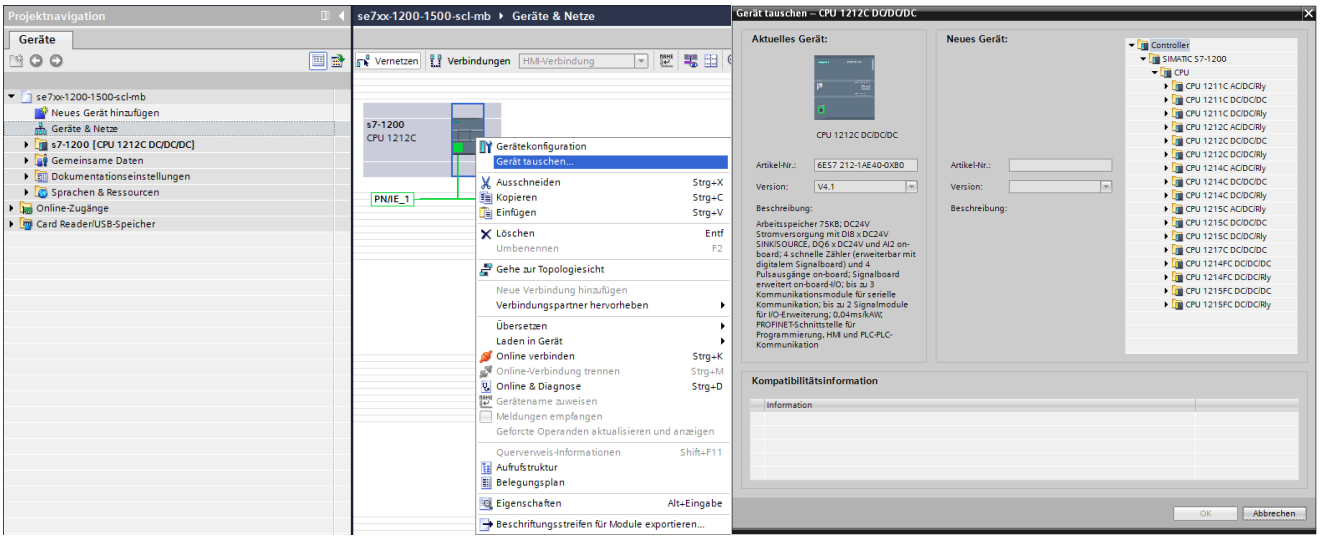
[1] Es wäre technisch möglich, eine Kopie der Bausteine zu erstellen und diese so anzupassen, dass auch ein zweiter SE-7xx angesteuert werden kann. Dies wird aber seitens Stange nicht unterstützt und deswegen wird hier nicht näher darauf eingegangen. Beispielsweise wäre auch zu beachten, dass bei der S7-1200 die Anzahl der möglichen Verbindungen auf 8 begrenzt ist.

Projekt als Vorlage verwenden

Das Projekt ***se7xx-1200-1500-scl-mb*** 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.



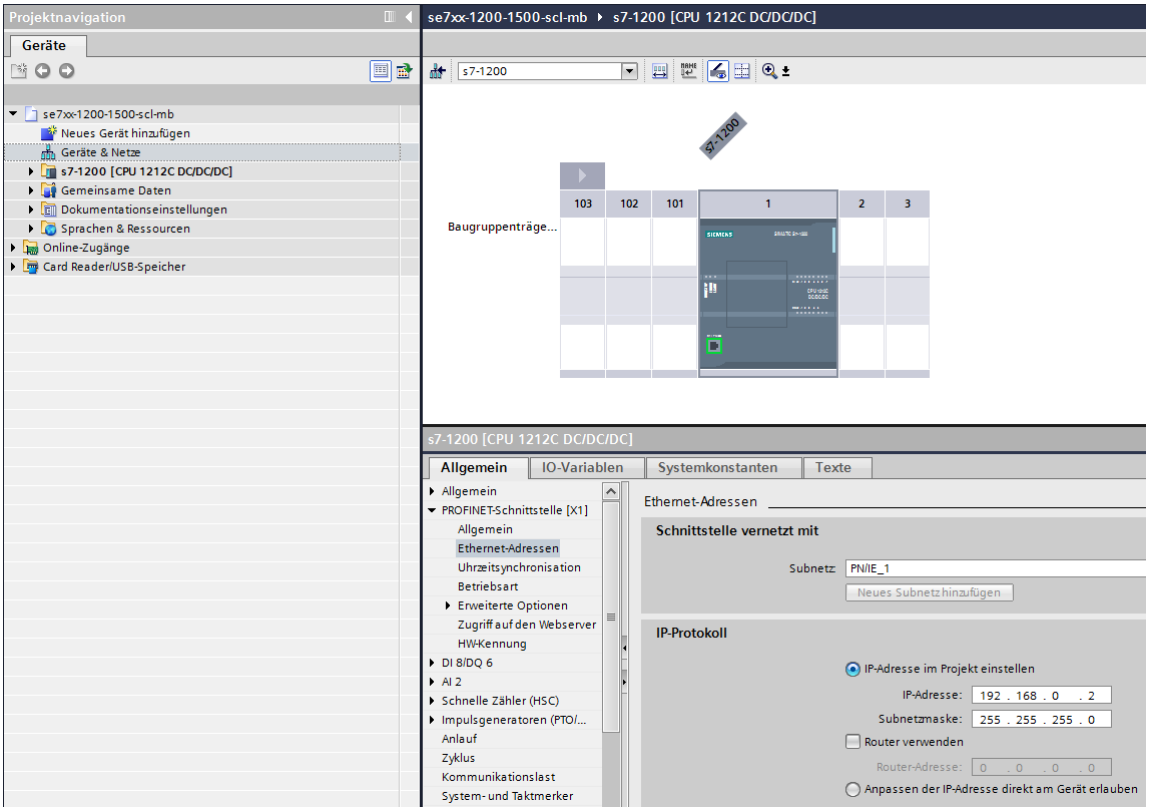
Im Vorlageprojekt sind jeweils eine S7-1212C DC/DC/DC und eine S7-1513-1 PN projektiert. Beide enthalten die gleichen Bausteine. Wenn eine andere SPS als die S7-1212C DC/DC/DC oder S7-1513-1 PN verwendet wird, muss die Projektierung angepasst werden. Dazu wird unter ***Geräte & Netze*** die S7 mit rechts 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. Die Bausteine müssen anschließend übersetzt und auf die S7 geladen werden.

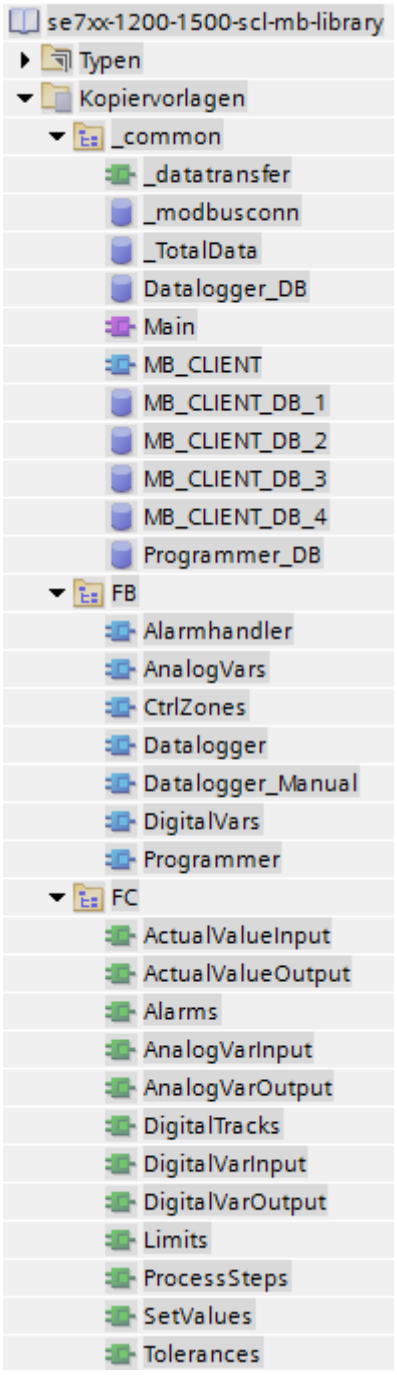
Der SE-7xx und die S7 müssen sich im selben Subnetz (Subnetzmaske) befinden, da es sonst zu Kommunikationsfehlern kommen kann.



Bibliotheksbausteine in bereits bestehendem Projekt verwenden

Wenn bereits ein S7-Projekt in TIA Portal existiert und dieses lediglich um die Modbus-Kommunikation erweitert werden soll, dann kann auf die Bibliothek ***se7xx-1200-1500-scl-mb-library*** zurückgegriffen werden.
Die dort enthaltenen Bausteine sind dieselben, welche sich auch im Vorlageprojekt befinden.

Das folgende Bild zeigt eine Übersicht der enthaltenen Bausteine:



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

- **_datatransfer** und **_modbusconn**
- **_TotalData**
- **Datalogger** [FB114] und ***Datalogger_DB*** (bei Verwendung des Datenloggers)
- **MB_CLIENT**
- **MB_CLIENT_DB**[1-4]

Das Projekt und die Bibliothek nutzen die Systembibliothek „S7 Open user communication“ in der Version 4.0.

Die Bausteine benötigen Systemmerker. Diese können in der Gerätekonfiguration unter „System- und Taktmerker“ durch Setzen des Hakens „Verwendung des Systemmerkerbytes aktivieren“ eingeschaltet werden.

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 eingebunden werden. Ein Beispiel-OB1 ist in der Bibliothek enthalten.

Bei aktiviertem Datenlogger muss der FB ***Datalogger*** über den OB1 eingebunden und mit einem IDB ausgestattet sein. Dieser Schritt ist zwingend notwendig, da der Datenlogger auf externe Steuerung angewiesen ist und 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 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 ***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.

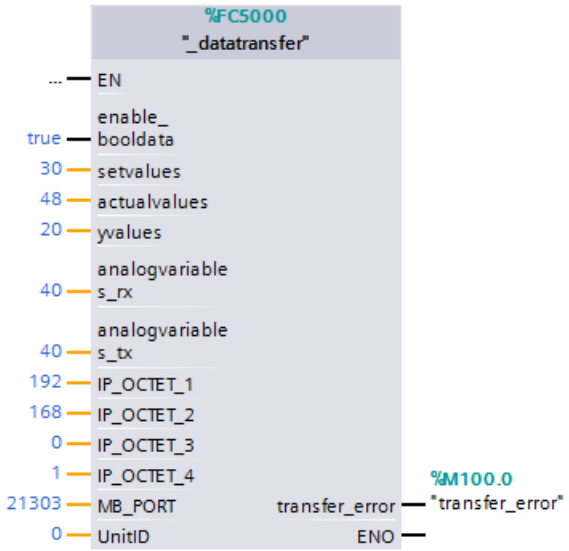
Beschreibung der Funktionalität

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 Modbus-Kommunikation möglich.

_datatransfer [FC5000]	
Parameter	**Beschreibung**
enable_booldata	Steuer-/Statuswerte austauschen [*true/false*]
setvalues	Anzahl Sollwerte [0..30]
actualvalues	Anzahl Istwerte [0..48]
yvalues	Anzahl Regler-Y-Werte [0..20]
analogvariables_rx	Anzahl Analogvariablen vom SE-7xx [0..40]
analogvariables_tx	Anzahl Analogvariablen zum SE-7xx [0..40]
IP_OCTET_[1..4]	IP-Adresse des SE-7xx (Standard: 192.168.0.1)
MB_PORT	Modbus/TCP-Port des SE-7xx (Standard: 21303)
UnitID	Modbus Unit ID des SE-7xx (Standard: 0)

Hier muss die IP-Adresse und ggf. Port des SE-7xx angegeben werden. Am Ausgang ***transfer_error*** wird ggf. ein Fehler bei der Modbus-Übertragung signalisiert. Dieser kann auf einen Merker oder eine Bool-Variable in einem DB gelegt werden.
Die angegebenen Anzahlen beeinflussen nur die übertragene TCP-Datenmenge zwischen SE-7xx und S7. Werden bspw. nur 30 statt 48 Istwerte angegeben (Eingang ***actualvalues***), so werden nur jeweils 30 Istwerte in beide Richtungen aktualisiert. Die restlichen Werte bleiben dann auf dem alten Stand vor der Änderung des Wertes. Die Änderung eignet sich daher nur zur Trafficreduzierung im Netzwerk und ist nur selten nötig. Dies gilt auch für den Eingang ***enable_booldata***.



Die einzelnen Bausteine entsprechen den Funktionen des SE-7xx. 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 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.

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 (FE 2000-2199)
DigitalVarOutput	FC109	Digitalvariable vom SE-7xx lesen (FA 2000-2199)
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 set-zen und lesen
AnalogVars	FB110	Mehrere Analogvariablen gleichzeitig set-zen und lesen
Datalogger	FB114	Datenlogger steuern und abfragen (Auto-matikmodus)
Datalogger_Manual	FB115	Datenlogger steuern und abfragen (manu-ell)

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 Modbus TCP zuständig. Dazu wird auf den Systembaustein ***mb_client*** zurückgegriffen. Es werden vier TCP-Verbindungen genutzt, über die jeweils zwei verschiedene Datenübertragungen gemultiplext werden. Jeder Datenübertragung einer Verbindung wird dabei je einer von zwei ***timeslices*** (Zeitscheiben) zugewiesen. Die Zeitscheiben werden nacheinander und jede Verbindung für sich abgearbeitet.

	Datenübertragung	
Verbindung	**c[1-4]_1**	**c[1-4]_2**
1	Booldata.rx	Booldata.tx
2	Setvalues.rx	Actualvalues.rx
3	Yvalues.rx	Actualvalues.tx
4	Analogvariables.rx	Analogvariables.tx

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 Modbus-Register zu den Funktionen des SE-7xx geschieht vollautomatisch, sodass hier keine Einstellung oder Konfiguration nötig ist.

In seltenen Fällen kann es vorkommen, dass die Modbus Unit ID im SE-7xx geändert werden muss. In diesem Fall ist es nötig, diesen Wert auch am Eingang ***UnitID*** von ***_datatransfer*** anzugeben. Ansonsten ist keine Kommunikation möglich. Standardmäßig ist dieser Wert 0.

Auch wenn am Baustein ***_datatransfer*** geringere Werte angegeben werden als vorhanden sind, so wird stets der gesamte Datenbereich im SE-7xx geschrieben bzw. überschrieben. Aus diesem Grund kann z.B. der von der S7 verwendete Bereich der Analogvariablen (Werte 41-80) nicht reduziert werden. Die Anzahlen geben nur an, welche Werte aktualisiert werden. Die anderen Werte bleiben dann fest auf dem alten Stand.

Es ist immer nur eine 1:1-Datenübertragung möglich. Das heißt, ein SE-7xx kann nicht mit mehreren S7 verbunden sein und Daten austauschen. Ebenso kann eine S7 nur mit einem SE-7xx Daten austauschen.[2]

- Verbindung 1:
  - BoolData empfangen (172 Byte) — status.booldata — c1_1
  - BoolData senden (140 Byte) — control.booldata — c1_2
- Verbindung 2:
  - Sollwerte empfangen (120 Byte[3]) — status.setvalues — c2_1
  - Istwerte empfangen (192 Byte[3]) — status.actvalues — c2_2
- Verbindung 3:
  - Y-Werte empfangen (80 Byte[3]) — status.yvalues — c3_1
  - Istwerte senden (192 Byte[3]) — control.actvalues — c3_2
- Verbindung 4:
  - Analogvariablen empfangen (160 Byte[3]) — status.analogvars — c4_1
  - Analogvariablen senden (160 Byte[3]) — control.analogvars — c4_2

[2] Es wäre technisch möglich, eine Kopie der Bausteine zu erstellen und diese so anzupassen, dass auch ein zweiter SE-7xx angesteuert werden kann. Dies wird aber seitens Stange nicht unterstützt und deswegen wird hier nicht näher darauf eingegangen. Beispielsweise wäre auch zu beachten, dass bei der S7-1200 die Anzahl der möglichen Verbindungen auf 8 begrenzt ist.
[3] bei Übertragung aller Werte (4 Byte pro Wert)

control	Struct	0.0		☐	☑	☑	☐	send data
booldata	Array[0..1119] of B...	0.0		☐	☑	☑	☐	bool data
actvalues	Array[0..47] of Real	140.0		☐	☑	☑	☐	actualvalues
analogvars	Array[0..39] of Real	332.0		☐	☑	☑	☐	analog variables
status	Struct	492.0		☐	☑	☑	☐	receive data
booldata	Array[0..1375] of B...	0.0		☐	☑	☑	☐	bool data
actvalues	Array[0..47] of Real	172.0		☐	☑	☑	☐	actualvalues
setvalues	Array[0..29] of Real	364.0		☐	☑	☑	☐	setvalues
yvalues	Array[0..19] of Real	484.0		☐	☑	☑	☐	controlzone y values
analogvars	Array[0..39] of Real	564.0		☐	☑	☑	☐	analog variables
timeslice	Struct	1216.0		☐	☑	☑	☐	connection 1-4 timeslices 1-2
c1_1	Bool	0.0	false	☐	☑	☑	☐	
c1_2	Bool	0.1	false	☐	☑	☑	☐	
c2_1	Bool	0.2	false	☐	☑	☑	☐	
c2_2	Bool	0.3	false	☐	☑	☑	☐	
c3_1	Bool	0.4	false	☐	☑	☑	☐	
c3_2	Bool	0.5	false	☐	☑	☑	☐	
c4_1	Bool	0.6	false	☐	☑	☑	☐	
c4_2	Bool	0.7	false	☐	☑	☑	☐	
transferblock	Struct	1218.0		☐	☑	☑	☐	transferblocks status data (mb_client)
1	Struct	0.0		☐	☑	☑	☐	1
2	Struct	4.0		☐	☑	☑	☐	2
3	Struct	8.0		☐	☑	☑	☐	3
4	Struct	12.0		☐	☑	☑	☐	4
5	Struct	16.0		☐	☑	☑	☐	5
6	Struct	20.0		☐	☑	☑	☐	6
7	Struct	24.0		☐	☑	☑	☐	7
8	Struct	28.0		☐	☑	☑	☐	8

_TotalData.control.booldata

Array [0..1119] of Bool.

Der SE-7xx wird durch *enable_booldata* am FC5000 *_datatransfer* mit diesen Daten aktualisiert.

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.

Wird durch *enable_booldata* am FC5000 *_datatransfer* mit aktuellen Daten gefüllt.

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.

Werden durch Werte größer Null am FC5000 *_datatransfer* mit aktuellen Daten gefüllt,
bzw. der SE-7xx wird mit diesen Daten aktualisiert.

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.

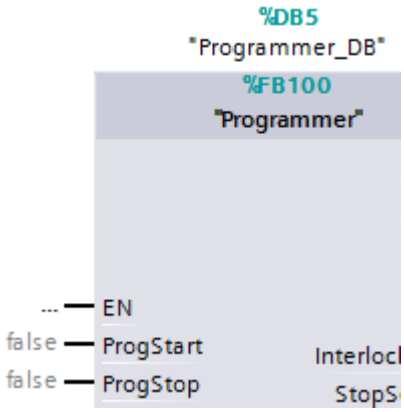
FB100: Programmer und FB114: Datalogger

Der Datenlogger des SE-7xx funktioniert bei eingeschalteter S7-Modbusschnittstelle 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.

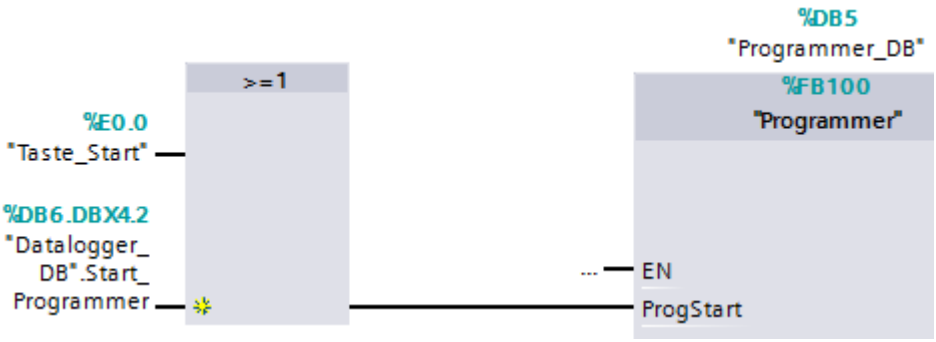
Um den Datenlogger (und den Programmgeber) über die S7 zu starten, genügt es, einen Impuls an den Eingang ***ProcessStart*** anzulegen. Nach fünf Sekunden wird der Programmgeber automatisch gestartet. Sobald das Programm beendet ist oder der Bediener END oder RESET auswählt, stoppt der Logger automatisch. 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 Modbus-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 die Chargendaten im SE-7xx.

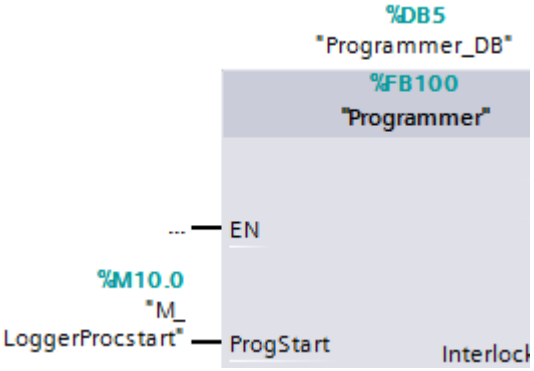
ProcessstartActive kann dann verwendet werden, um mittels ***LogStart*** den Datenlogger zu starten. Außerdem muss dann über den Eingang ***ProgStart*** von ***Programmer*** der Programmgeber gestartet werden. ***ProcessstartActive*** wird automatisch zurückgesetzt, sobald der Programmgeber läuft (das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

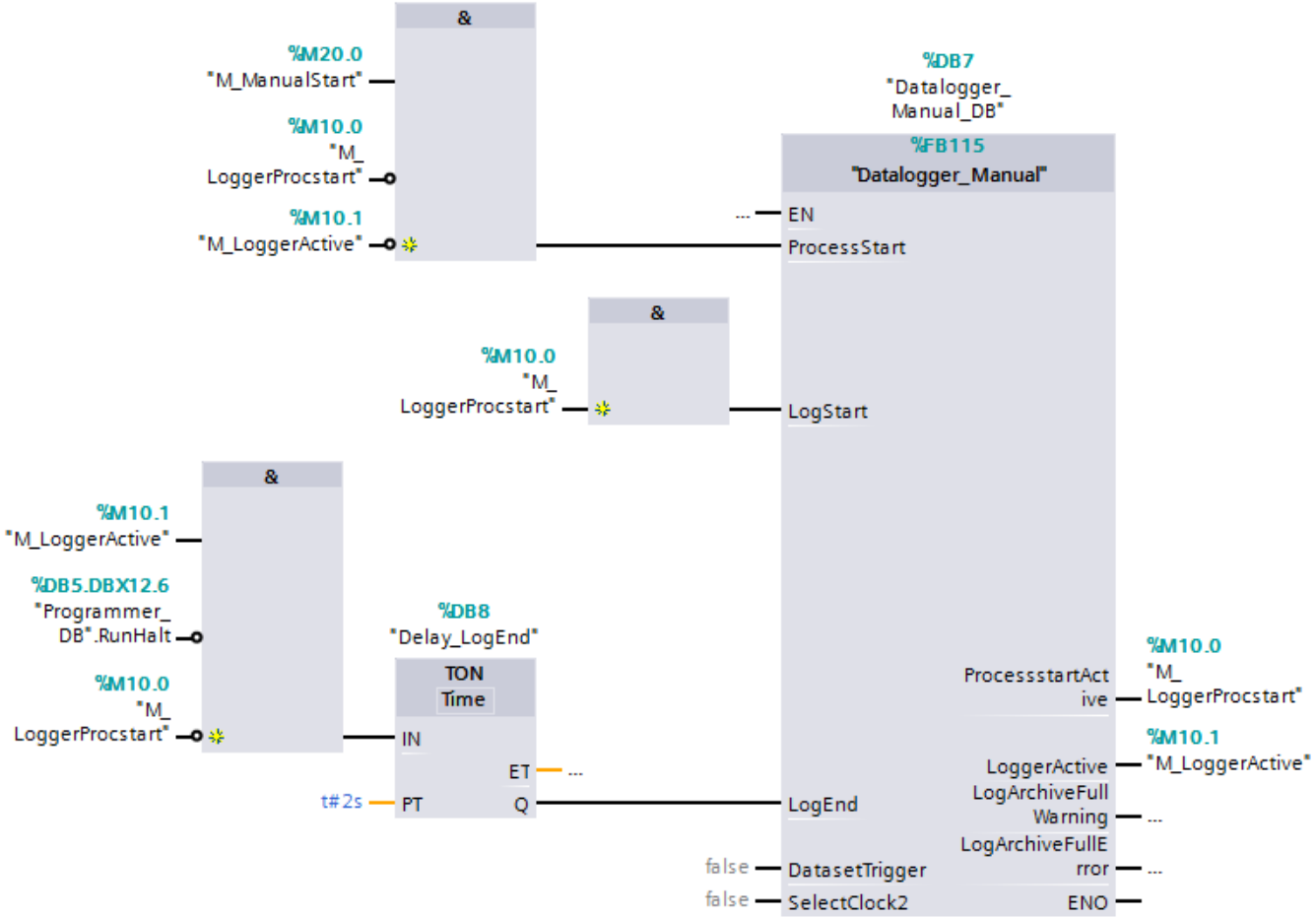
LoggerActive zeigt an, dass der Datenlogger läuft und gerade eine Charge aufzeichnet. Mittels ***LogEnd*** wird die Aufzeichnung der Charge schließlich beendet.

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.

Der Parameter ***Shift10*** an ***DigitalVars*** gibt an, welche zehn Digitalvariablen angesteuert werden sollen. Bei einem Wert von 0 werden die jeweils ersten zehn Variablen angesprochen, bei einem Wert von 1 die jeweils nächsten zehn usw.

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.

Sowohl bei ***AnalogVarInput*** als auch bei ***AnalogVarOutput*** muss als ***InstanceNo*** ein Wert zwischen 1-40 angegeben werden.

Analogvariablen müssen vor ihrer Verwendung als IEEE-Float konfiguriert sein (Konfiguration im SE-7xx).
Andere Variablentypen werden nicht unterstützt.

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 neue S7-Modbusschnittstelle 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 jetzt 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.

Interface der Bausteine (FC/FB)

_datatransfer [FC5000]

Data transfer between datablock _TotalData and SE-7xx.

Input	Format	Function
enable_booldata	Bool	Enable to transfer bool data between SE-7xx and S7
setvalues	UInt	Set the number of Setvalues to receive from SE-7xx
actualvalues	UInt	Set the number of Actualvalues to exchange with SE-7xx
yvalues	UInt	Set the number of Yvalues to receive from SE-7xx
analogvariables_rx	UInt	Set the number of Analogvariables to receive from SE-7xx
analogvariables_tx	UInt	Set the number of Analogvariables to send to SE-7xx
IP_OCTET_[1..4]	UInt	First/second/third/fourth part of IPv4 address of SE-7xx
MB_PORT	UInt	Port number of SE-7xx (default: 21303)
UnitID	UInt	Modbus Unit ID of SE-7xx (default: 0)

Output	Format	Function
transfer_error	Bool	Error flag: one or more transfer failed

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