Hand out Programmierung mit GRAPH S7

Erste Schritte

S7 GRAPH ist eine grafische Programmiersprache, die sich in ihrer Syntax an die Ablaufbeschreibungssprache Grafcet eng anlehnt und auf diese Weise eine einfache Umsetzung von Grafcet Plänen in das TIA Portal ermöglicht. Damit ergänzt sie sich sehr gut zu den bereits etablierten Programmiersprachen FUP und SCL.

Das folgende Hand-Out beschreibt das grundsätzliche Vorgehen bei der Programmierung mit Graph im TIA Portal und bietet einen Überblick über dessen Funktionalitäten.

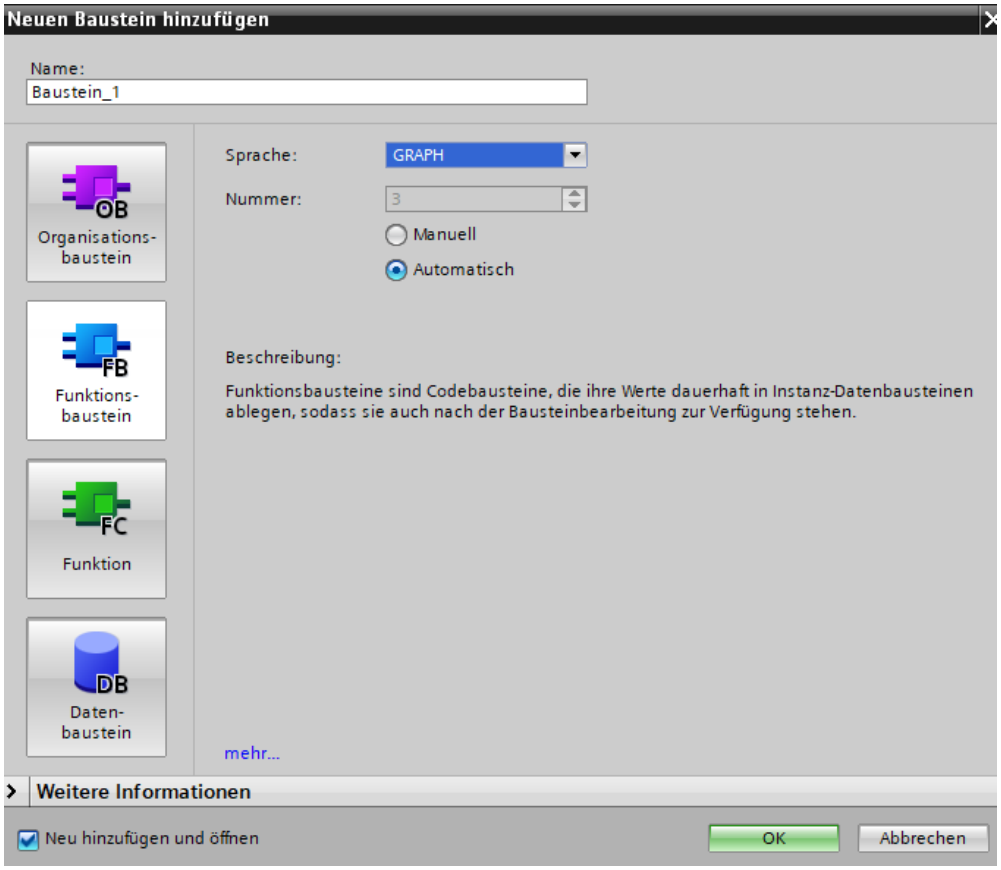
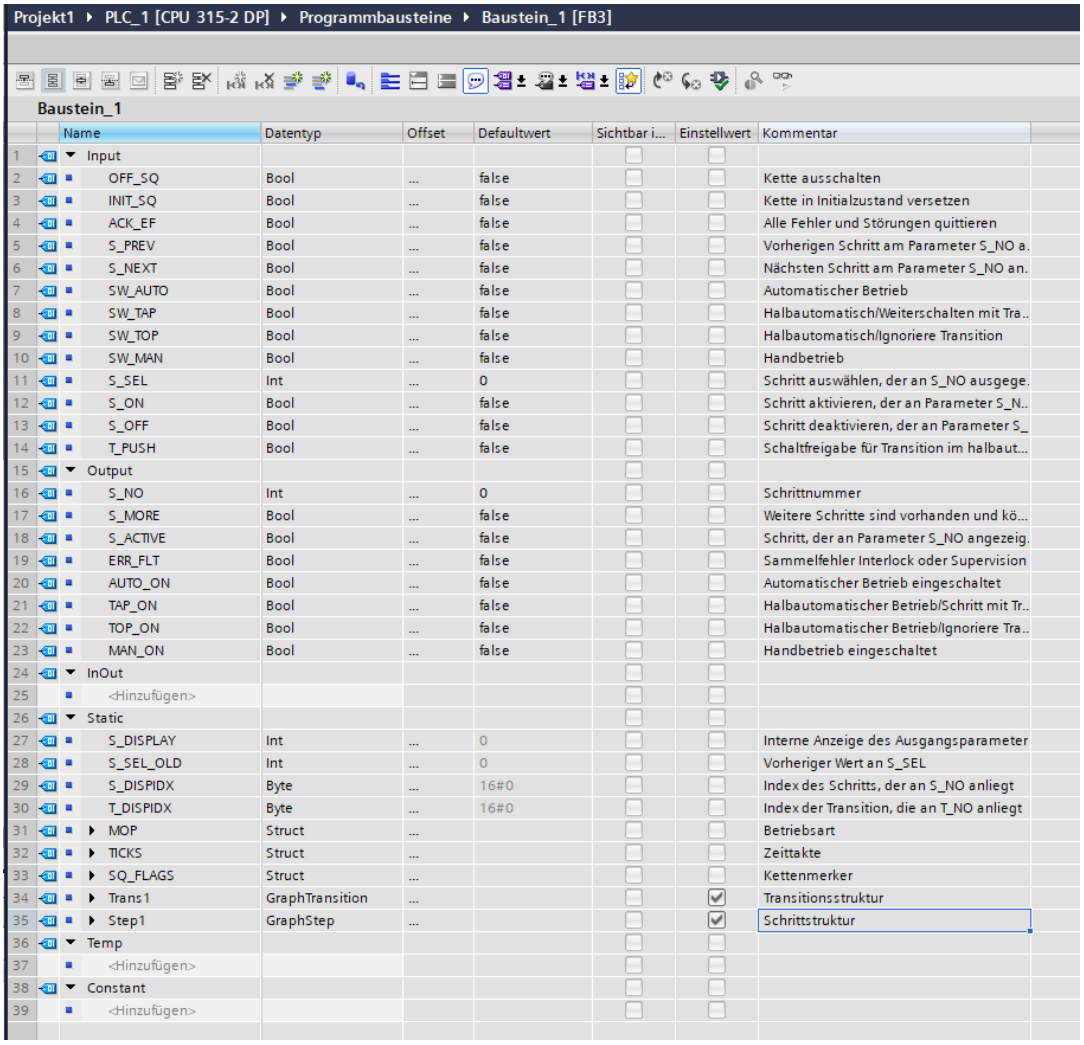


Abbildung 1: Erstellung eines Funktionsbausteins in S7 GRAPH

Für die Erstellung eines Graph- Programms werden im TIA Portal ausschließlich Funktionsbausteine verwendet. Eine Programmierung in einer Funktion ist nicht möglich, weil die Ablaufsteuerung im Verlauf der zyklischen Programmbearbeitung die jeweils aktiven Schritte für den folgenden Programmzyklus zwischenspeichern muss. Da eine Funktion über kein „Gedächtnis“ verfügt, kann sie hier nicht verwendet werden.

Funktionalität des Bausteins

Nach Erstellen des Funktionsbausteins lässt sich dieser bearbeiten; es öffnet sich der gewohnte Arbeitsbereich, inklusive der Variablen des vorgefertigten GRAPH Bausteins, die später als Schnittstelle zwischen der Schrittkette und dem OB 1 dienen. Es können lokale Input, Output, InOut-Variablen (können gelesen und geschrieben werden) sowie statische Variablen (statische Variablen bleiben über den Programmzyklus erhalten) verwendet werden. Abbildung 2 zeigt die Standardmäßige Ausstattung eines leeren FB mit Variablen.



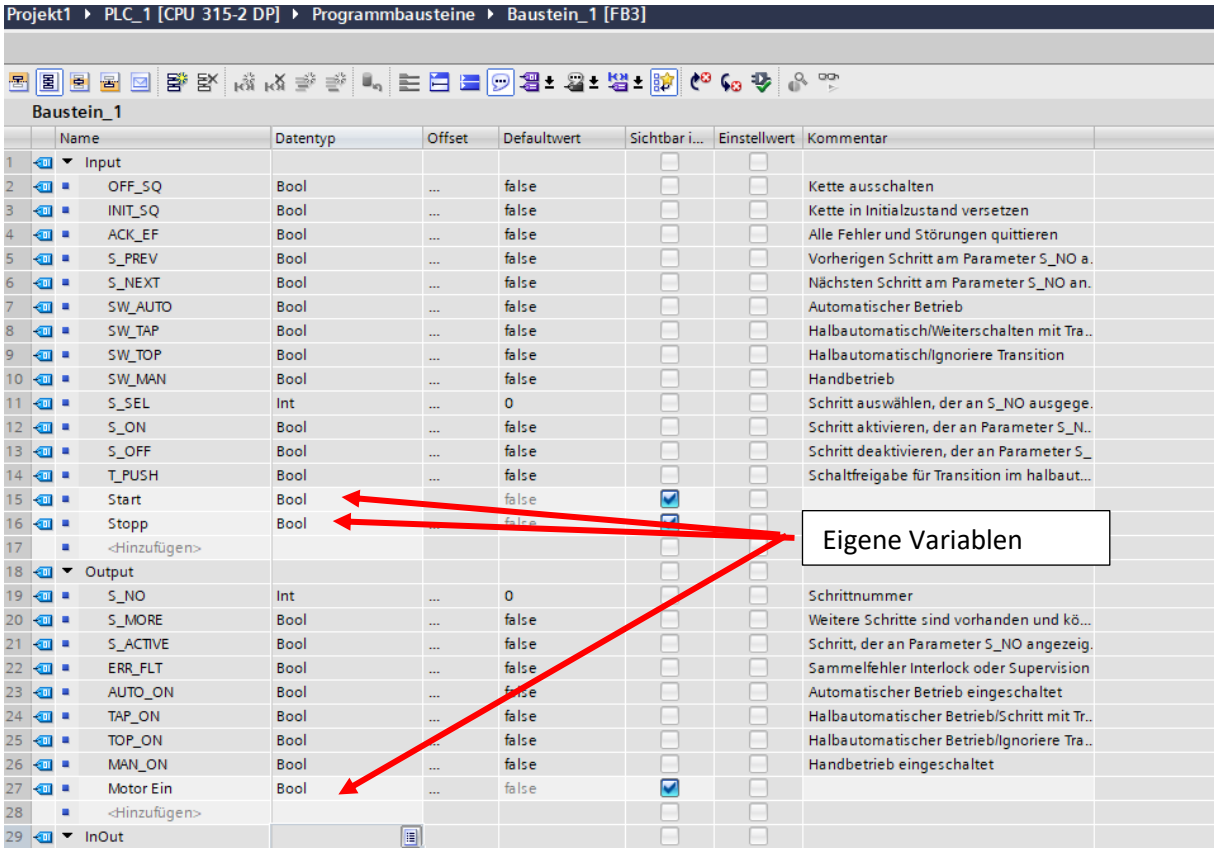
	Name	Datentyp	Offset	Defaultwert	Sichtbar i...	Einstellwert	Kommentar
1	Input				☐	☐	
2	OFF_SQ	Bool	...	false	☐	☐	Kette ausschalten
3	INIT_SQ	Bool	...	false	☐	☐	Kette in Initialzustand versetzen
4	ACK_EF	Bool	...	false	☐	☐	Alle Fehler und Störungen quittieren
5	S_PREV	Bool	...	false	☐	☐	Vorherigen Schritt am Parameter S_NO a.
6	S_NEXT	Bool	...	false	☐	☐	Nächsten Schritt am Parameter S_NO an.
7	SW_AUTO	Bool	...	false	☐	☐	Automatischer Betrieb
8	SW_TAP	Bool	...	false	☐	☐	Halbautomatisch/Weiterschalten mit Tra..
9	SW_TOP	Bool	...	false	☐	☐	Halbautomatisch/Ignoriere Transition
10	SW_MAN	Bool	...	false	☐	☐	Handbetrieb
11	S_SEL	Int	...	0	☐	☐	Schritt auswählen, der an S_NO ausgege.
12	S_ON	Bool	...	false	☐	☐	Schritt aktivieren, der an Parameter S_N..
13	S_OFF	Bool	...	false	☐	☐	Schritt deaktivieren, der an Parameter S_
14	T_PUSH	Bool	...	false	☐	☐	Schaltfreigabe für Transition im halbaut...
15	Output				☐	☐	
16	S_NO	Int	...	0	☐	☐	Schrittnummer
17	S_MORE	Bool	...	false	☐	☐	Weitere Schritte sind vorhanden und kö...
18	S_ACTIVE	Bool	...	false	☐	☐	Schritt, der an Parameter S_NO angezeig.
19	ERR_FLT	Bool	...	false	☐	☐	Sammelfehler Interlock oder Supervision
20	AUTO_ON	Bool	...	false	☐	☐	Automatischer Betrieb eingeschaltet
21	TAP_ON	Bool	...	false	☐	☐	Halbautomatischer Betrieb/Schritt mit Tr..
22	TOP_ON	Bool	...	false	☐	☐	Halbautomatischer Betrieb/Ignoriere Tra..
23	MAN_ON	Bool	...	false	☐	☐	Handbetrieb eingeschaltet
24	InOut				☐	☐	
25	<Hinzufügen>				☐	☐	
26	Static				☐	☐	
27	S_DISPLAY	Int	...	0	☐	☐	Interne Anzeige des Ausgangsparameter
28	S_SEL_OLD	Int	...	0	☐	☐	Vorheriger Wert an S_SEL
29	S_DISPIDX	Byte	...	16#0	☐	☐	Index des Schritts, der an S_NO anliegt
30	T_DISPIDX	Byte	...	16#0	☐	☐	Index der Transition, die an T_NO anliegt
31	MOP	Struct	...		☐	☐	Betriebsart
32	TICKS	Struct	...		☐	☐	Zeittakte
33	SQ_FLAGS	Struct	...		☐	☐	Kettenmerker
34	Trans1	GraphTransition	...		☐	☑	Transitionsstruktur
35	Step1	GraphStep	...		☐	☑	Schrittstruktur
36	Temp				☐	☐	
37	<Hinzufügen>				☐	☐	
38	Constant				☐	☐	
39	<Hinzufügen>				☐	☐	

Abbildung 2: Standard-Variablen eines GRAPH Bausteins

Die Kommentierung auf der rechten Seite gibt dem Anwender Aufschluss über die Funktionalität der Variablen. Wichtige Variablen sind:

- OFF_SQ: Kette ausschalten
- INIT_SQ: Kette in Initialzustand versetzen
- ACK_EF: Fehler und Störungen quittieren

Neben diesen Variablen ist es auch möglich, eigene lokale Variablen im FB zu erstellen. Alle Variablen sind nur im lokal im Funktionsbaustein sichtbar und nicht in anderen Programmbausteinen.



	Name	Datentyp	Offset	Defaultwert	Sichtbar i...	Einstellwert	Kommentar
1	Input						
2	OFF_SQ	Bool	...	false			Kette ausschalten
3	INIT_SQ	Bool	...	false			Kette in Initialzustand versetzen
4	ACK_EF	Bool	...	false			Alle Fehler und Störungen quittieren
5	S_PREV	Bool	...	false			Vorherigen Schritt am Parameter S_NO a.
6	S_NEXT	Bool	...	false			Nächsten Schritt am Parameter S_NO an.
7	SW_AUTO	Bool	...	false			Automatischer Betrieb
8	SW_TAP	Bool	...	false			Halbautomatisch/Weiterschalten mit Tra..
9	SW_TOP	Bool	...	false			Halbautomatisch/Ignoriere Transition
10	SW_MAN	Bool	...	false			Handbetrieb
11	S_SEL	Int	...	0			Schritt auswählen, der an S_NO ausgege.
12	S_ON	Bool	...	false			Schritt aktivieren, der an Parameter S_N..
13	S_OFF	Bool	...	false			Schritt deaktivieren, der an Parameter S_
14	T_PUSH	Bool	...	false			Schaltfreigabe für Transition im halbaut...
15	Start	Bool		false	☑		
16	Stopp	Bool		false	☑		
17	<Hinzufügen>						
18	Output						
19	S_NO	Int	...	0			Schrittnummer
20	S_MORE	Bool	...	false			Weitere Schritte sind vorhanden und kö...
21	S_ACTIVE	Bool	...	false			Schritt, der an Parameter S_NO angezeig.
22	ERR_FLT	Bool	...	false			Sammelfehler Interlock oder Supervision
23	AUTO_ON	Bool	...	false			Automatischer Betrieb eingeschaltet
24	TAP_ON	Bool	...	false			Halbautomatischer Betrieb/Schritt mit Tr..
25	TOP_ON	Bool	...	false			Halbautomatischer Betrieb/Ignoriere Tra..
26	MAN_ON	Bool	...	false			Handbetrieb eingeschaltet
27	Motor Ein	Bool	...	false	☑		
28	<Hinzufügen>						
29	InOut						

Abbildung 3: Eigene Baustein-Variablen

Abbildung 3 zeigt beispielhaft eigens angelegte Variablen. Wird der FB später im OB1 eingefügt, so erscheinen diese ebenfalls als Schnittstellenvariablen im OB.

Diese Art der Programmierung bietet den Vorteil, unabhängig von der realen Verdrahtung der angeschlossenen Geräte eine Schrittkette programmieren zu können. Damit ist der FB an sich in anderen Programmen bzw. in anderen Projekten sehr gut wiederverwendbar. Außerdem bedeutet die symbolische Programmierung (z.B. Motor Ein) eine angenehmere Handhabung als die Benennung nach Ein- und Ausgangsadressen.

Auf der anderen Seite ist die Erstellung von lokalen Variablen in Bezug auf das Gesamtprogramm manchmal nicht sinnvoll, weil mehrere Funktionsbausteine auf die Variablen zugreifen müssen. Dann ist es besser, diese in einem globalen Datenbaustein abzulegen und von dort aus darauf zuzugreifen.

Einbindung des Funktionsbausteins in den Organisationsbaustein

Wie bei allen anderen Bausteinen auch, muss der Funktionsbaustein erst im OB1 bekannt gemacht werden, um aktiviert zu werden. Dies am einfachsten per Drag and Drop.

Nach Einfügen in den OB1 erkennt man dann die Schnittstellen zum lokalen FB, indem Eingänge und Ausgänge als Schnittstellen hier herausgeführt werden (Abbildung 4).

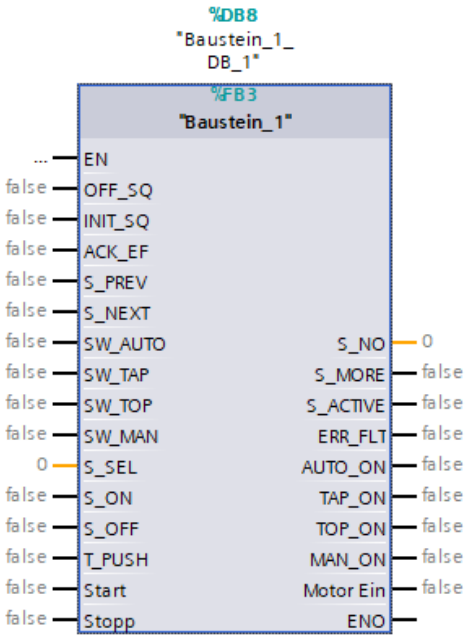


Abbildung 4: Deklarierung des Bausteins im OB1

Grundlagen der Schrittkettenprogrammierung

Nachdem der Rahmen der Programmerstellung nun festgelegt ist, kann die eigentliche Schrittkettenprogrammierung nach Grafcet durchgeführt werden.

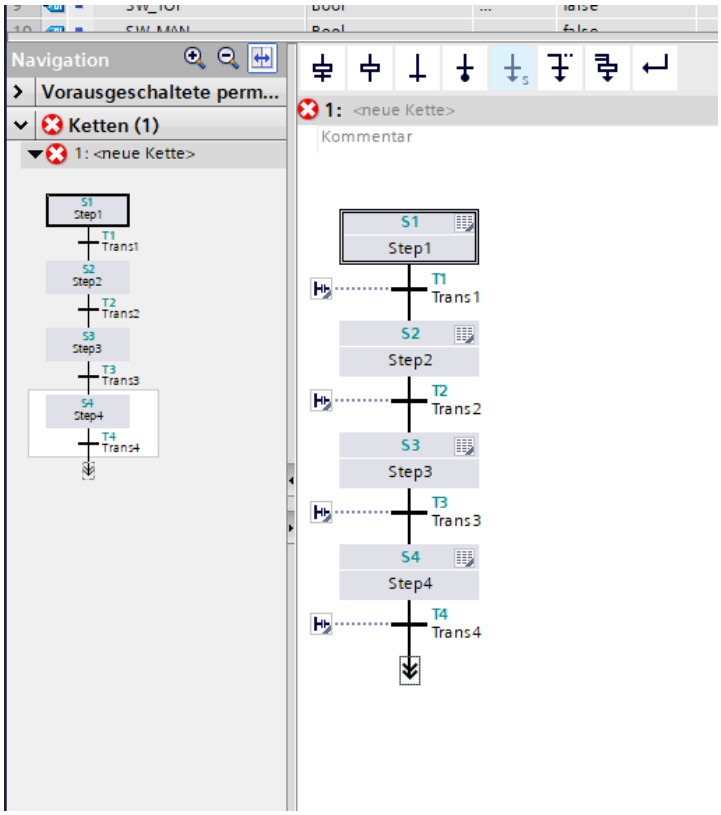


Abbildung 5: Beispiel für eine Schrittkette

Abbildung 5 zeigt ein Beispiel für eine Schrittkette. Wie beim Grafcet auch, ist der Initialschritt hier bereits vorgefertigt und mit einem doppelten Rahmen versehen. In der oberen Leiste können dann Schritte und Transitionen, Sprünge und parallele Programmausführungen eingefügt werden.

Der Grafcet Plan kann somit fast Eins zu Eins in die Software übertragen werden. Es ist darauf zu achten, dass am Ende der Kette entweder ein Sprung zurück zu einem vorigen Programmpunkt erfolgt oder ein Endpunkt eingefügt wird, andererseits wird beim Kompilieren eine Fehlermeldung ausgegeben.

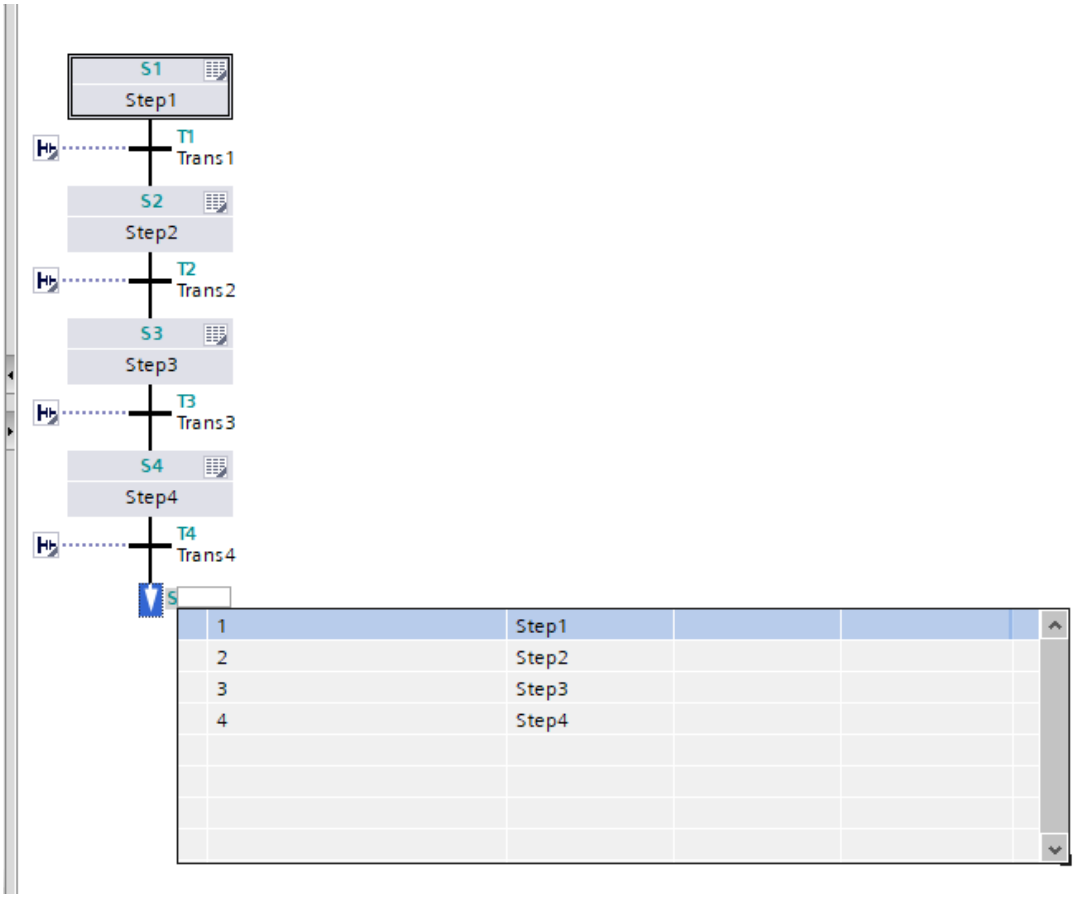


Abbildung 6: Vervollständigen der Schrittkette

Programmieren von Schritten und Aktionen

Bei der Programmierung von Schritten ist wie folgt vorzugehen:

Jeder Schritt umfasst eine eigene kleine Umgebung bzw. Arbeitsbereich, der sich bei Doppelklick öffnet. Er besteht aus jeweils einem separaten Netzwerk für Interlocks, Supervisionen, der folgenden

Transition sowie die Aktion, siehe Abbildung 7.

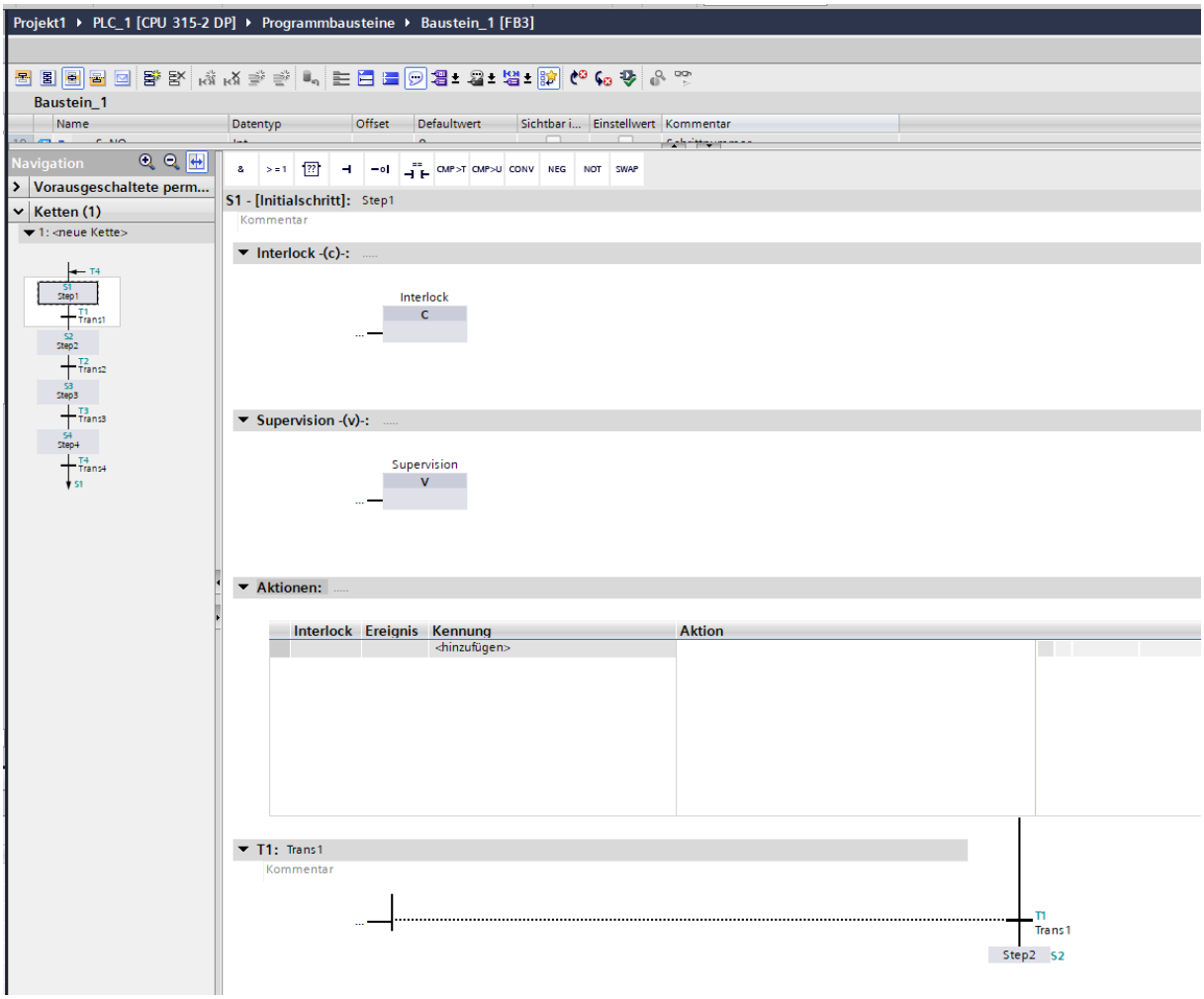


Abbildung 7: Arbeitsbereich des Einzelschritts

Anmerkung: Standardmäßig ist die Programmiersprache der Netzwerke KOP. In den Eigenschaften des Funktionsbaustein lässt sie sich in FUP umstellen:

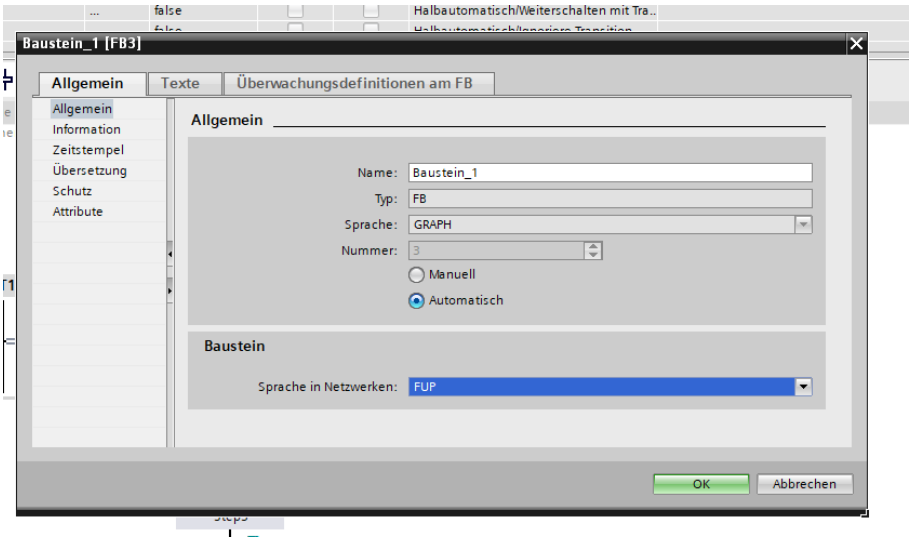


Abbildung 8: Umstellung der Programmiersprache

Zurück zur Schrittkettenprogrammierung: Die Wichtigste Komponente stellt das Netzwerk Aktionen dar. Dort können Aktoren angesteuert oder Zähler erhöht werden. Aktionen können dabei mit Ereignissen oder Kennungen versehen werden. Unter den Ereignissen lässt sich einstellen, wann die Aktion ausgeführt ist. In der Regel wählt man das Ereignis kommender Schritt S1, was bedeutet, dass die Aktion ausgeführt wird, sobald die Kette in diesen Schritt schaltet. Andersherum kann die Aktion auch erst mit dem gehenden Schritt aktiviert werden, usw.

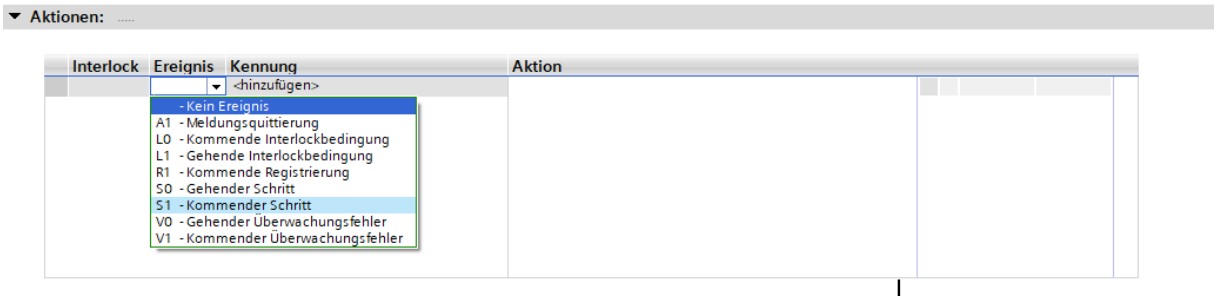


Abbildung 9: Auswahl von Ereignissen

Als nächstes lässt sich die Kennung einstellen. Die Kennung steuert die Zeit, in der der Aktion durchgeführt wird. Wählt man zum Beispiel die Kennung N, so wird die Aktion nur für die Dauer des Schrittes ausgeführt, was der kontinuierlich wirkenden Aktion im Grafcet entspricht. Die Kennung S und R entsprechen der speichernd wirkenden Aktion. Zusätzlich bietet das TIA Portal noch weitere Funktionen, wie hoch- oder herunterzählen.

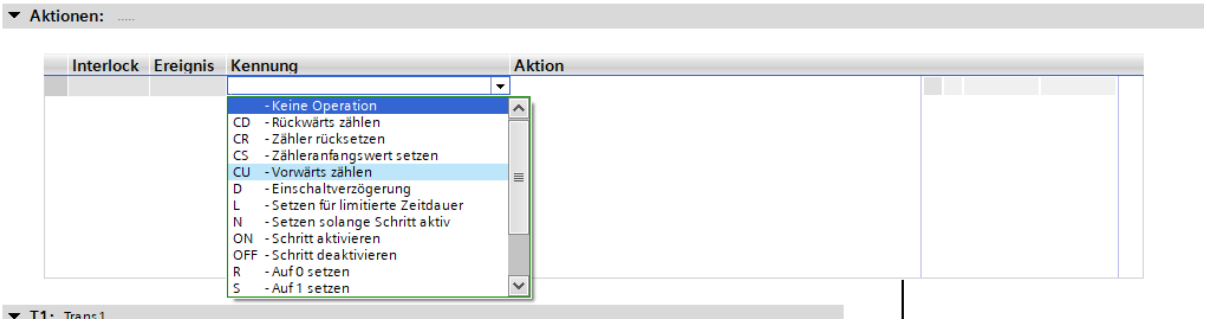


Abbildung 10: Auswahl von Kennungen

Im Feld „Aktion“ wird dann eingetragen, welche Variable oder welcher Ausgang etc. angesteuert werden soll. In der letzten Spalte lassen sich wie gehabt Kommentare eingeben.

Wird eine Aktion an eine Bedingung geknüpft, so wird dies im TIA Portal in einer sogenannten Interlock-Bedingung realisiert.

Sie erscheint im obersten Netzwerk und wird mit einem C (Condition) bezeichnet. Hier können Variablen im bekannten Muster miteinander verknüpft werden und bilden dann die Zuweisungsbedingungen des Grafcet. Zur Aktivierung des Interlocks muss in der Spalte Interlock im Aktions-Netzwerk noch das -c- ausgewählt werden.

Im Gegensatz zum Grafcet ist es hier jedoch nicht möglich, im gleichen Schritt unterschiedliche Interlocks zu programmieren. Pro Schritt ist ein Interlock möglich. Werden in einem Schritt zum Beispiel 2 verschiedene Interlocks benötigt, so muss ein weiterer Schritt in der Kette eingefügt werden, sodass die Anzahl an Schritten in der Software die der Schritte im Grafcet meistens übersteigt.

Eine weitere Gestaltungsmöglichkeit ist die Programmierung von Supervisionen. Der wesentliche Unterschied zwischen einer Supervision und einem Interlock besteht darin, dass bei einer Supervision die Ausführung einer Aktion nicht unterbunden werden kann, allerdings das Weiterschalten in den nächsten Schritt. Bei einem Interlock ist es genau andersrum, dort kann die Ausführung einer Aktion verhindert werden, jedoch nicht das Fortfahren der Kette. Da eine Supervision zur Ablaufüberwachung dient, wird bei dessen Eintreten eine Meldung erstellt, die dann quittiert werden muss (im Gegensatz zum Interlock).

Vorausgeschaltete und Nachgeschaltete Permanente Anweisungen

Neben der klassischen Schrittkettenprogrammierung lassen sich auch sogenannte vorausgeschaltete und nachgeschaltete permanente Anweisungen programmieren. Dabei handelt es sich um Programmcode, der entweder vor oder nach jedem Zyklus ausgeführt wird, unabhängig davon, in welchem Schritt sich die Ablaufsteuerung befindet.

Eine mögliche Anwendung für vorausgeschaltete permanente Anweisungen kann etwa eine Einschaltverzögerung sein. Abbildung 7 zeigt eine Möglichkeit der Programmierung. Hier gilt, wie beim Programmieren allgemein: Viele Wege führen nach Rom.

Dieses Netzwerk wird durch den Baustein TON realisiert, die vorgefertigte Verzögerung aus der Bibliothek. Der Parameter IN startet bei steigender Flanke die Einschaltverzögerung und schaltet den Ausgang Q durch, sobald die Zeit PT in Sekunden abgelaufen ist. Das Verknüpfungsergebnis wird der Variablen Time zugewiesen, die zuvor als InOut im lokalen Datenbaustein erstellt wurde.

Dieses Netzwerk eignet sich zum Beispiel für einen Übergang von einem Schritt zum Anderen.

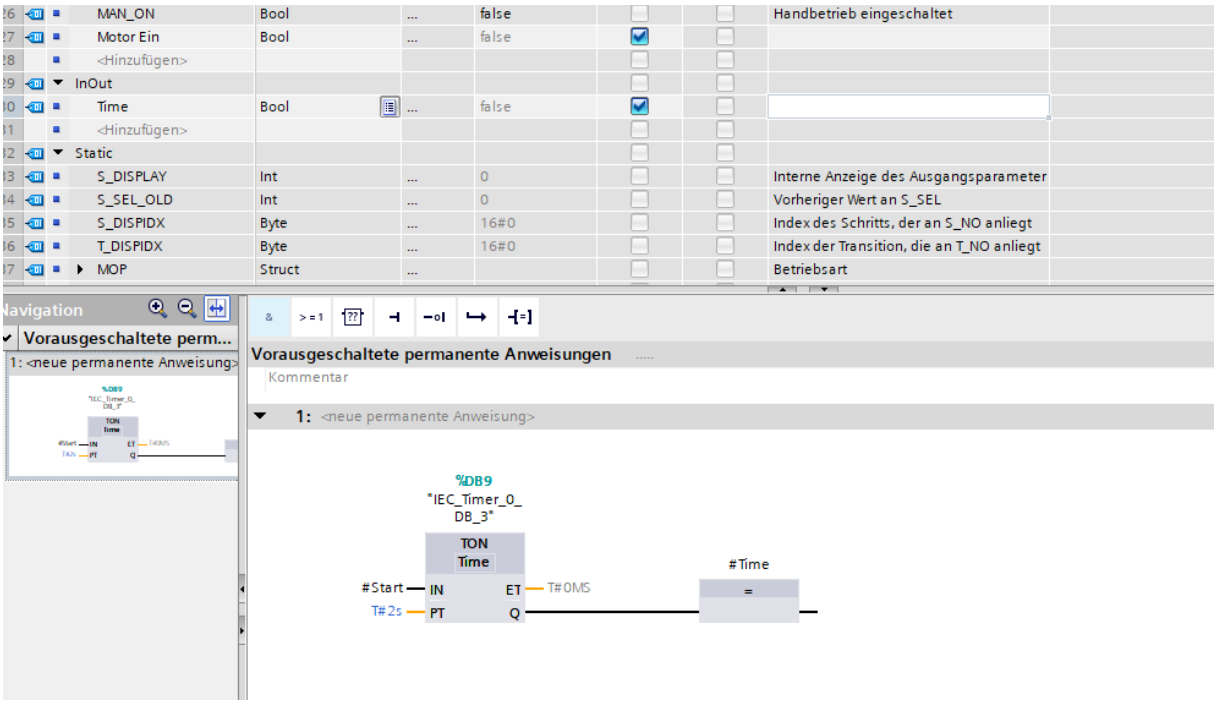


Abbildung 11: Programmieren einer Einschaltverzögerung