

# Grollmus SPS-Simulator Handbuch

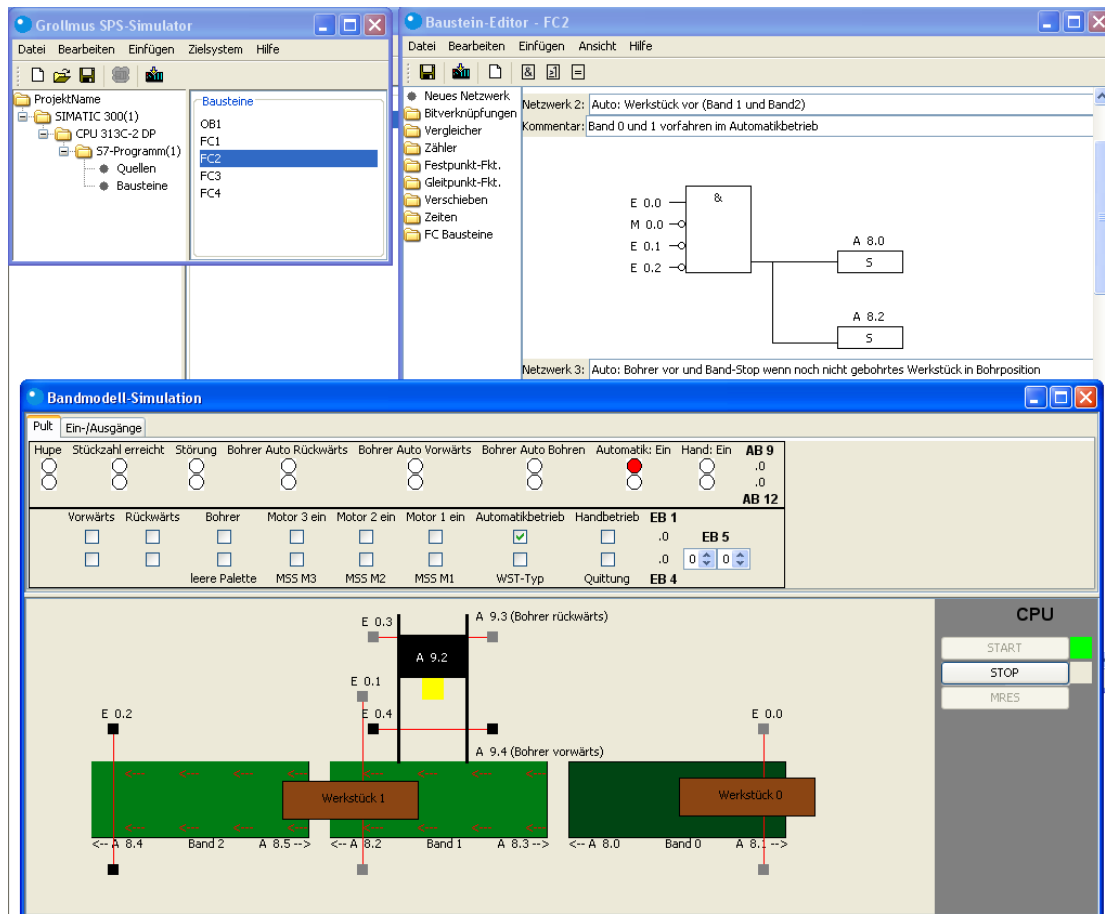


Abbildung 1: SPS-Simulator mit Bandmodell und FUP-Editor

Grollmus SPS-Simulator, Version 0.2  
[www.grollmus.de/sps-simulator.html](http://www.grollmus.de/sps-simulator.html)  
[simulator@grollmus.de](mailto:simulator@grollmus.de)  
 Stand: 30.06.2010

Copyright Grollmus GmbH, 2010

## Inhalt

1. Tutorial.....	3
1.1 Anlegen eines Bausteins .....	4
1.2 Editieren eines Bausteins .....	6
1.3 Starten der Bandmodell Simulation .....	14
1.4 Übertragen des Programms in die simulierte CPU .....	15
1.5 Starten der simulierten CPU.....	16
1.6 Speichern und Laden des Projektes.....	20
2. Zulässige AWL-Befehle.....	21
3. Belegung der Ein- und Ausgänge des Bandmodells.....	24

# 1. Tutorial

Das Tutorial ist eine kurze Einführung in den Grollmus SPS-Simulator. Es wird gezeigt, wie Sie ein Projekt mit Bausteinen erzeugen, eine Simulation starten und auswerten können.

Das erste was Sie sehen, wenn Sie den Grollmus SPS-Simulator starten ist das folgende Fenster:

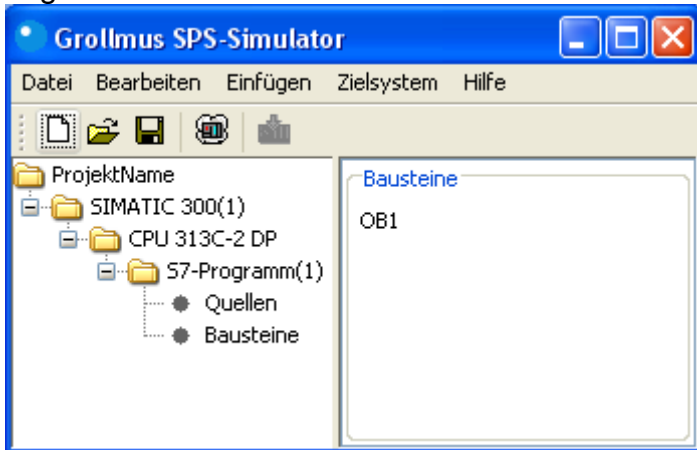


Abbildung 2: Startbildschirm

Der Startscreen zeigt in einer Liste die im Projekt vorhandenen Bausteine. Nach dem Start ist nur der Organisationsbaustein OB1 vorhanden. OB1 ist anfangs noch leer, enthält also keinen Code.

Dieses Tutorial zeigt Ihnen, wie Sie folgende Schritte ausführen:

- Bausteine (Funktionen) anlegen und löschen
- Bausteine editieren
- Bandmodell-Simulation starten
- SPS-Programm in die simulierte CPU übertragen können
- CPU starten und ihr Programm in der Bandmodell-Simulation testen
- SPS-Programm erweitern und sofort testen
- Projekt speichern/laden

## 1.1 Anlegen eines Bausteins

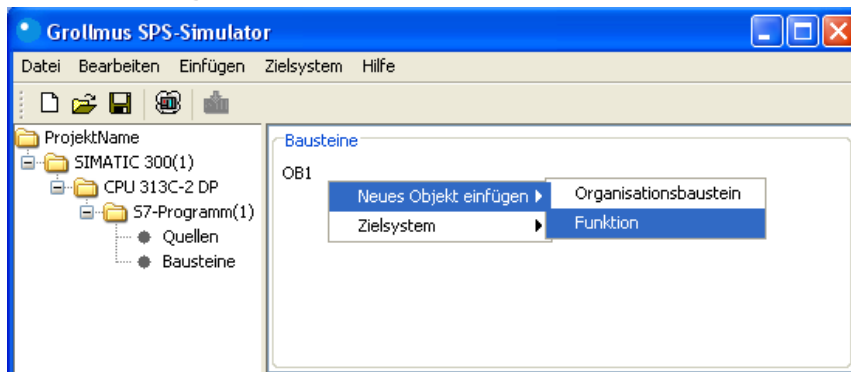


Abbildung 3: Anlegen einer Funktion über Kontextmenü

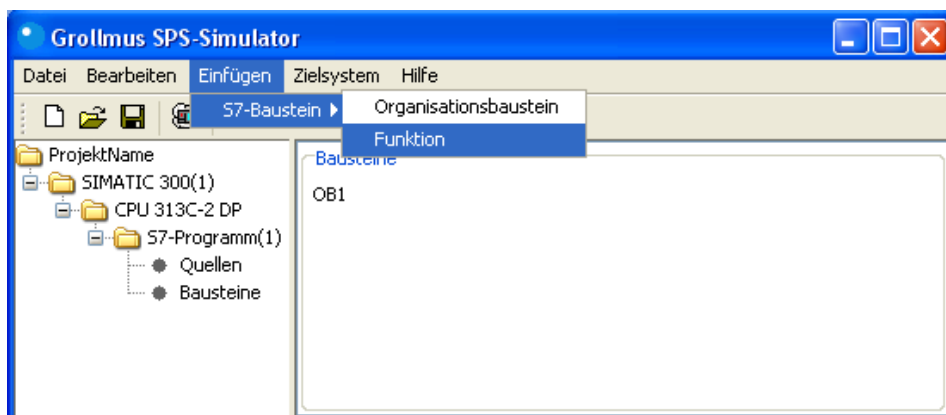


Abbildung 4: Anlegen einer Funktion über Menü

Sie können eine Funktion auf zwei Wegen anlegen:

- Über das Kontextmenü der Liste welche die Bausteine anzeigt. (siehe Abbildung 3)
- Über das Menü (siehe Abbildung 4)

In beiden Fällen erscheint ein Dialog, in dem Sie den Namen der Funktion festlegen. Dieser Name muss eindeutig sein und für Funktionen ist das Muster FC<nr> einzuhalten. Sie können hier in der Regel einfach den Vorgabewert übernehmen.

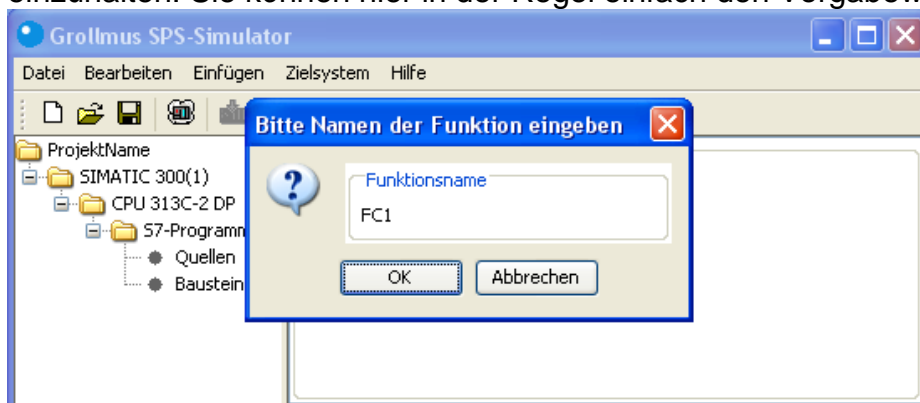


Abbildung 5: neue Funktion zum Projekt hinzufügen – Eingabe Funktionsname

Nach Bestätigung des Dialogs mit [OK] erscheint die neue Funktion in der Liste. Wiederholen Sie dies, so erhalten Sie als neuen Namens-Vorschlag für die Funktion „FC2“:

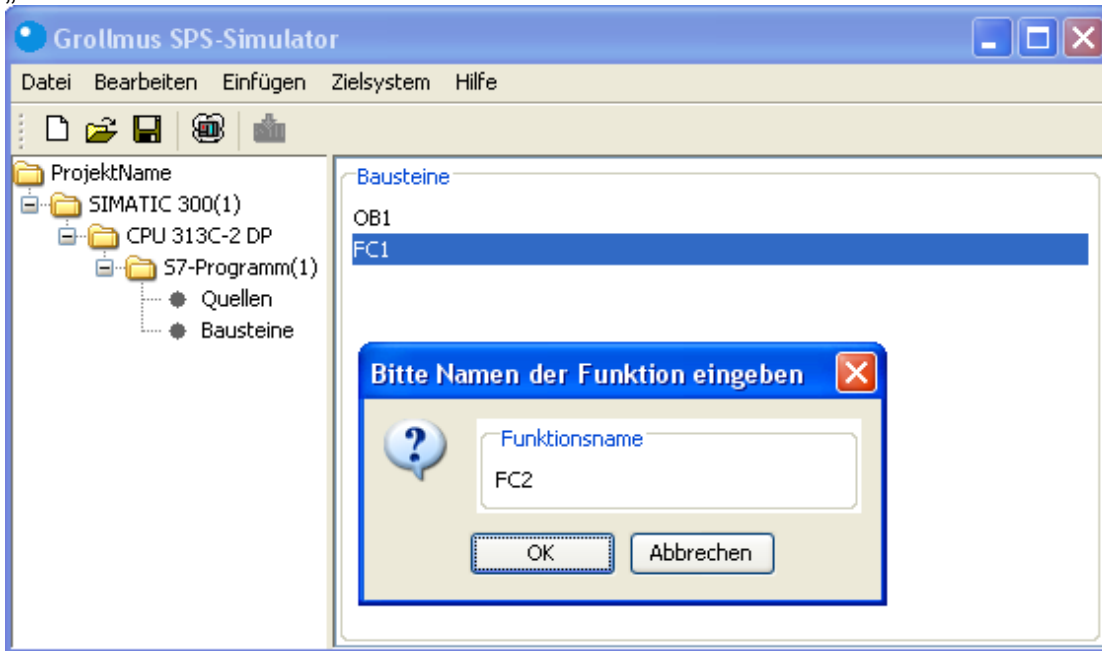


Abbildung 6: Neue Funktion hinzufügen – Namen der Funktion festlegen

Nun soll die Funktion FC2 wieder gelöscht werden. Dazu wählen Sie die Funktion in der Liste aus und rufen im Kontext-Menü „Löschen“ auf.

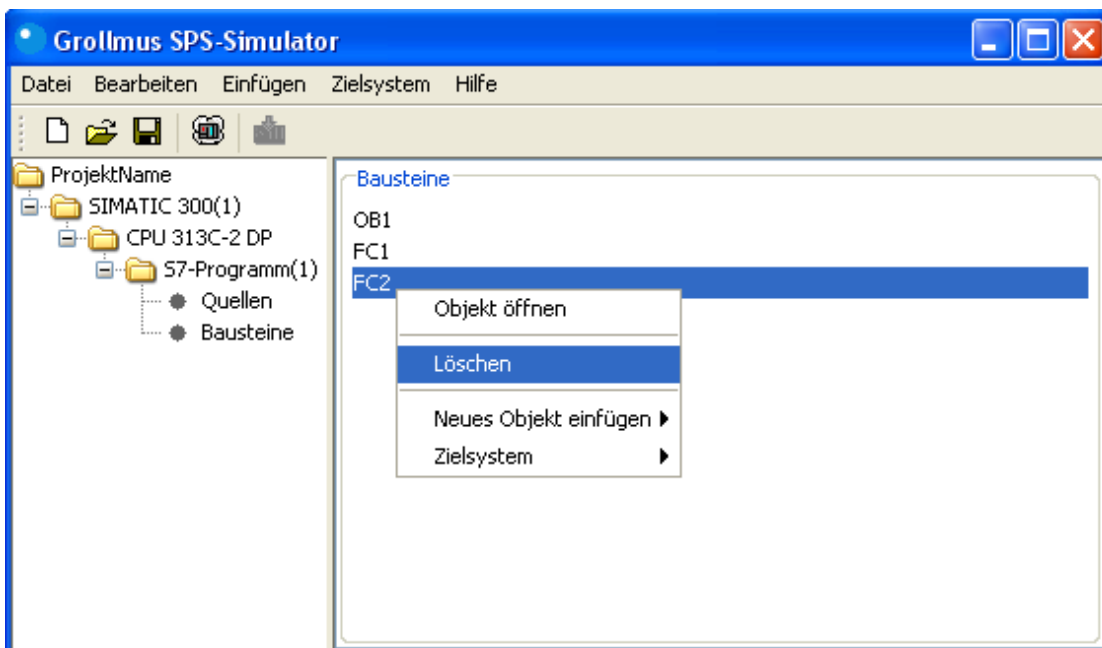


Abbildung 7: Löschen eines Bausteins über Kontextmenü

Hinweis: In dieser Version können zwar Organisationsbausteine zum Projekt hinzugefügt werden, sie werden jedoch nicht von der CPU aufgerufen, haben also keinerlei Wirkung.

## 1.2 Editieren eines Bausteins

Entweder über das Kontextmenü (erst den Eintrag ‚FC1‘ selektieren dann Rechtsklick) oder einfacher über einen Doppelklick auf ‚FC1‘ wird in einem zweiten Fenster der Baustein Editor für diese Funktion gestartet.

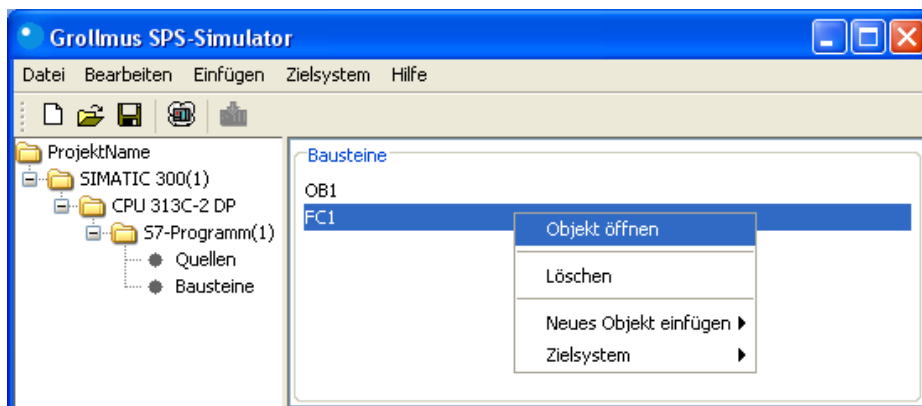


Abbildung 8: Starten des Baustein-Editors für FC1 über Kontext-Menü

Es erscheint als zweites Fenster der Baustein Editor.

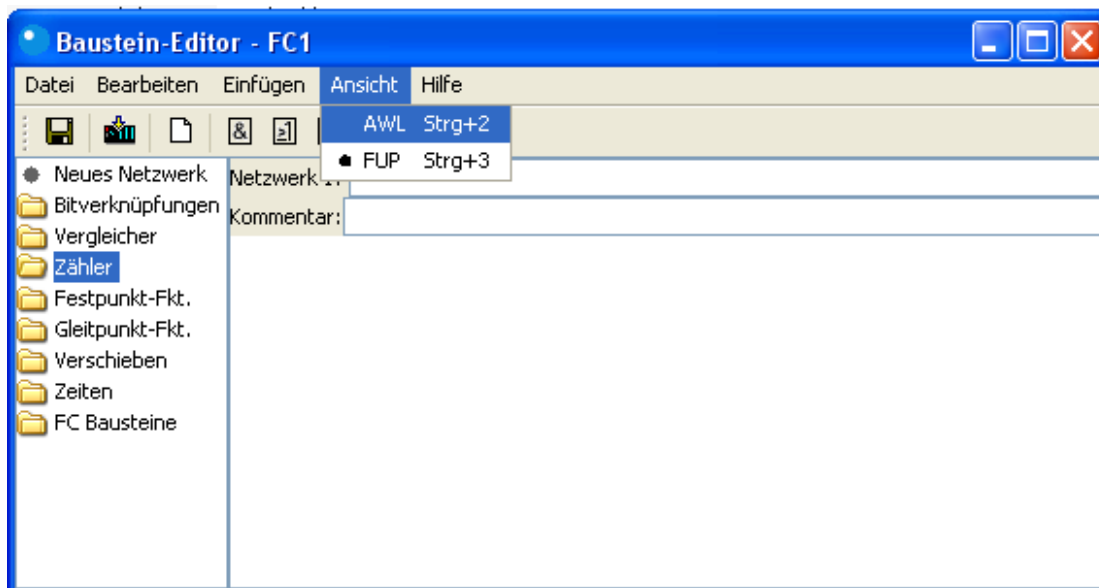


Abbildung 9: Baustein-Editor - Ansicht FUP

Die neu angelegte Funktion FC1 enthält noch keinen Programmcode. Es soll nun ein kleines Programm erstellt werden.

Programme können in diesem Simulator in AWL<sup>1</sup> und FUP<sup>2</sup> - jedoch nicht in KOP<sup>3</sup> - erstellt werden.

<sup>1</sup> AWL steht für Anweisungsliste – das Programm schreibt man als Text

<sup>2</sup> FUP steht für Funktionsplan – das Programm wird mit Symbolen in grafischer Form erstellt.

<sup>3</sup> KOP steht für Kontaktplan - Die graphische Programmiersprache Kontaktplan KOP nach der Norm IEC DIN EN 61131-3 ist der Darstellung von Stromlaufplänen nachempfunden.

Nach dem Start ist die Ansicht FUP eingestellt. Dies lässt sich auf zwei Wegen ändern:

- Über Menü->Ansicht ->AWL
- oder über den Kurzbefehl Strg+2

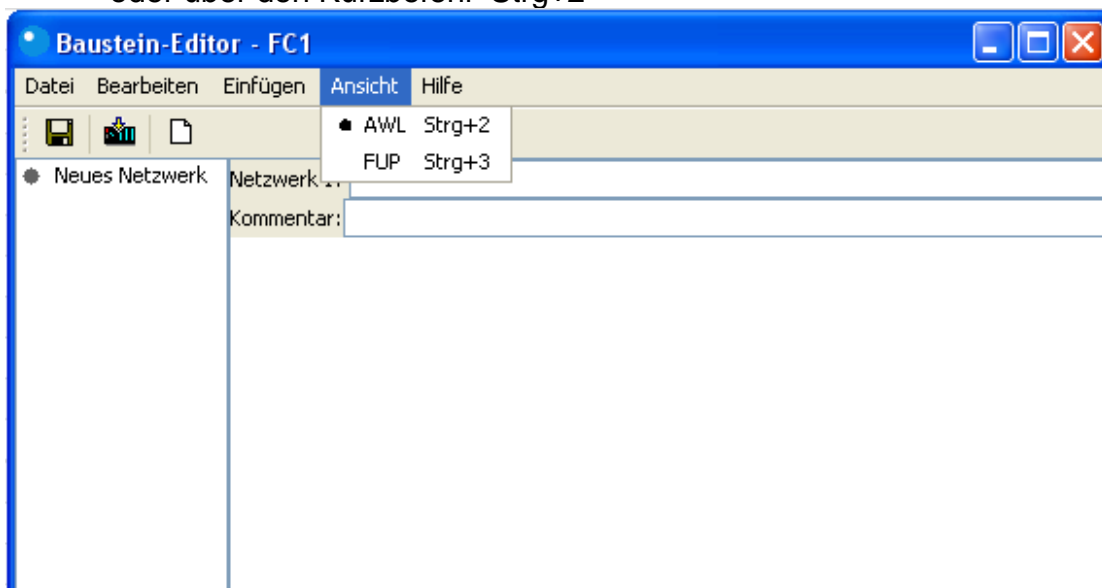


Abbildung 10; Baustein Editor - Ansicht AWL

Jeder Baustein besteht aus einem oder mehreren Netzwerken. Aktuell besteht der Baustein aus einem leeren Netzwerk.

Es soll nun das erste Netzwerk in AWL editiert werden. Dazu klicken Sie in das Netzwerk und tippen:

```
U E 0.0 // Wenn Eingang 0.0 true
S A 8.0 // Dann setze Ausgang 8.0 auf true
S A 8.2 // und setze Ausgang 8.2 auf true
```

Außerdem soll dem Netzwerk 1 noch einen Titel gegeben werden. Schreiben Sie hier: „Bänder 1, 2 starten wenn Werkstück Lichtschranke Pos 1 unterbricht“.

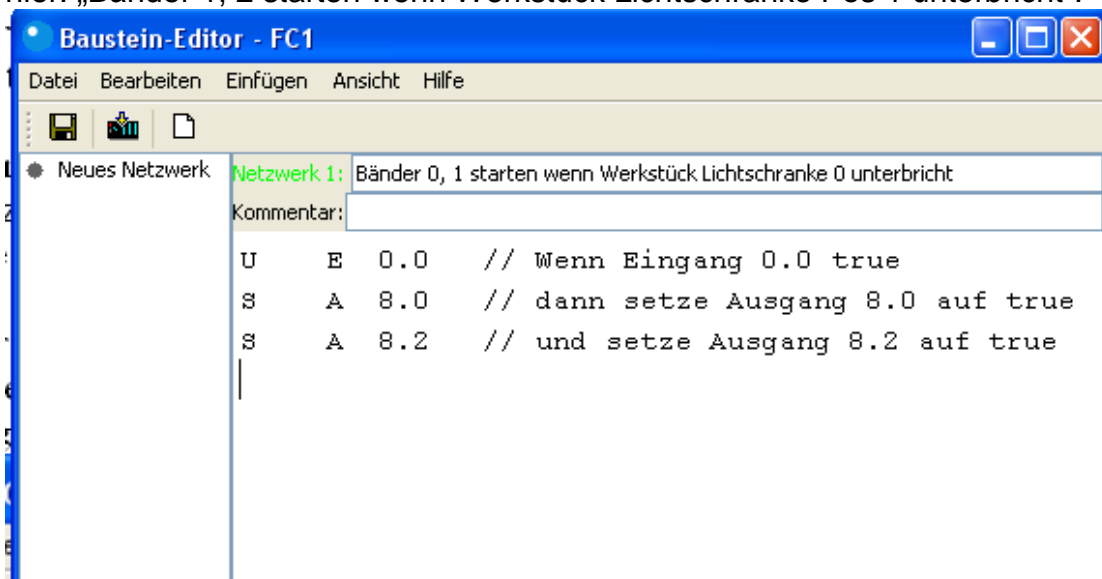


Abbildung 11: Netzwerk in AWL-Ansicht

Wenn die Ansicht nun auf FUP umgeschaltet wird sieht man:

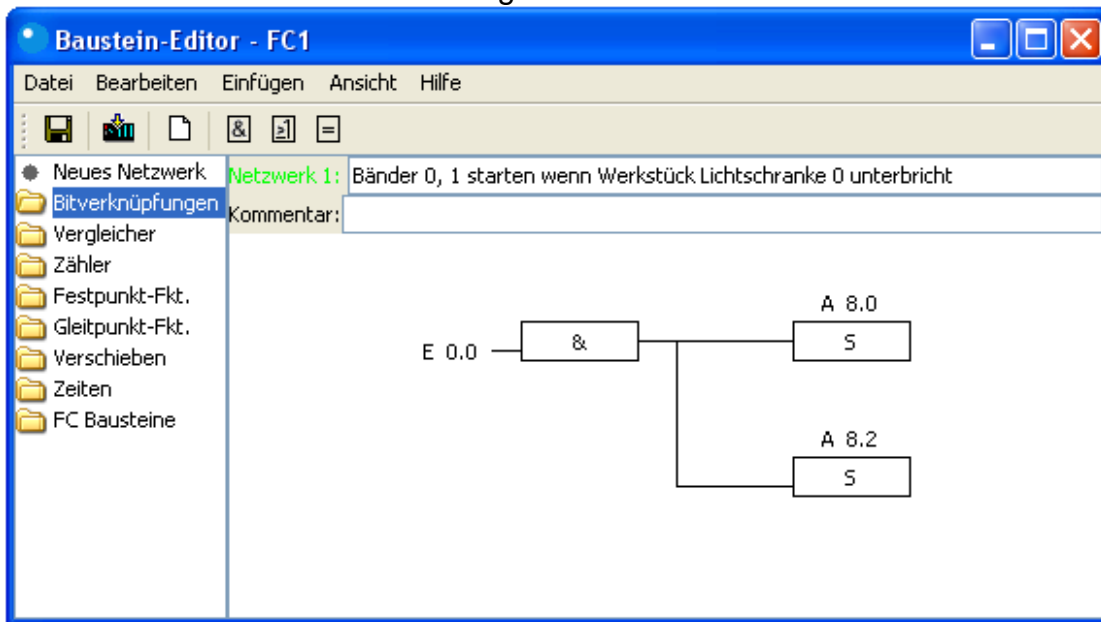


Abbildung 12: FUP Ansicht

Nun soll ein weiteres Netzwerk hinzugefügt werden. Dazu klicken Sie auf der linken Seite auf „Neues Netzwerk“. Daraufhin wird am Ende ein neues noch leeres Netzwerk hinzugefügt.

Das neue Netzwerk soll in FUP editiert werden. Ziel ist es, die beiden zuvor gesetzten Ausgänge (A 8.0 und A 8.1) zurückzusetzen, sobald der Eingang 0.1 (ist Licht-Taster Pos. 2) gesetzt - also Licht-Taster Pos.2 unterbrochen ist. Dies schreiben wir als Titel des neuen Netzwerks:

„Bänder stoppen wenn Werkstück LS 2 unterbricht“

Klicken Sie in das neue Netzwerk – daraufhin wird „Netzwerk 2“ grün markiert. Anschließend klicken Sie in der Toolbar auf das &-Symbol.

Sie erhalten:

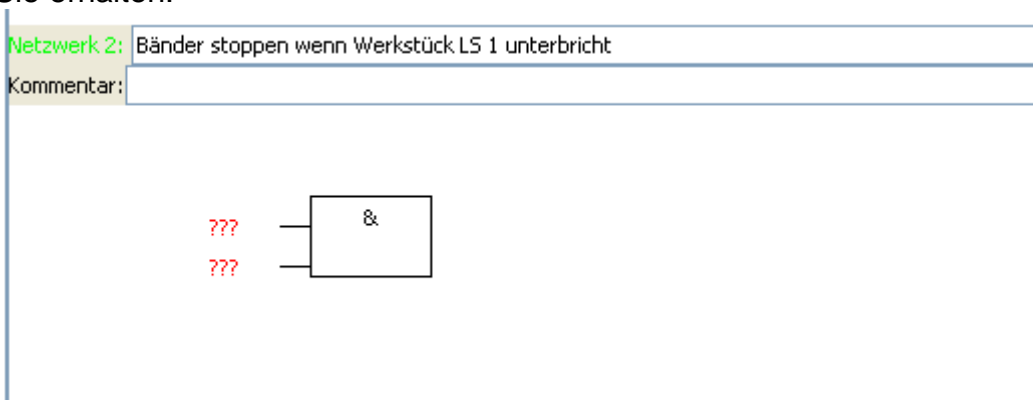


Abbildung 13: FUP-Netzwerk – UND-Gatter mit zwei noch „unbeschalteten“ Eingängen

Sie benötigen lediglich einen Eingang für das &-Gatter. Daher klicken Sie auf eines der rot markierten Eingänge und rufen dann über das Kontextmenü ‚Löschen‘ auf.

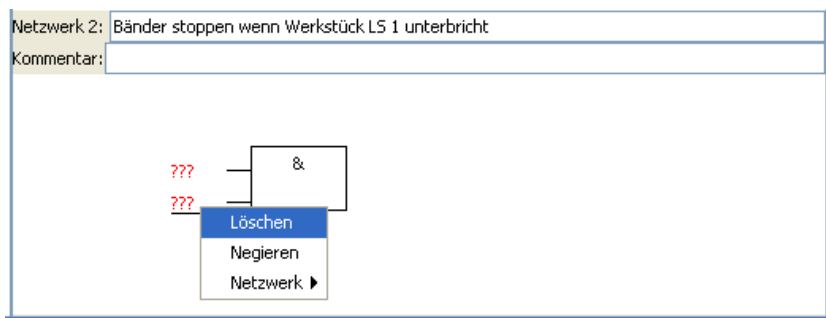


Abbildung 14: Löschen von Eingängen eines UND-Gatters über Kontextmenü

Anschließend klicken Sie auf den verbleibenden Eingang und schreiben darin ‚E 0.1‘. Da der Gatter-Eingang nun syntaktisch korrekt ist, wechselt die Schriftfarbe auf schwarz.

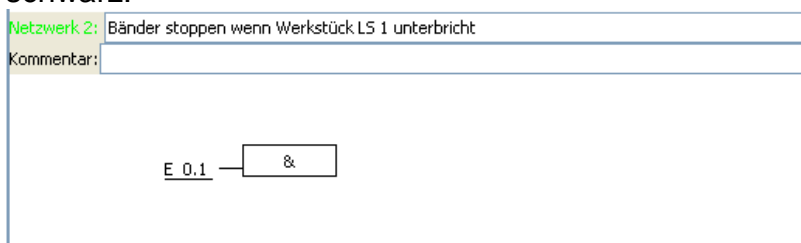


Abbildung 15: UND-Gatter - jedoch noch ohne Zuweisung des Verknüpfungsergebnisses

Das Ergebnis dieser „Verknüpfung“ soll nun ein „Reset“ der Ausgänge A 8.0 und A 8.2 bewirken. Um dies im FUP-Editor zu erreichen, klicken Sie zunächst auf das Gatter an welches wir ein Folgegatter anhängen wollen. Anschließend öffnen Sie im Baum die Kategorie „Bitverknüpfungen“ und klicken dann auf den Eintrag: „ - - [R]“

Sie erhalten:

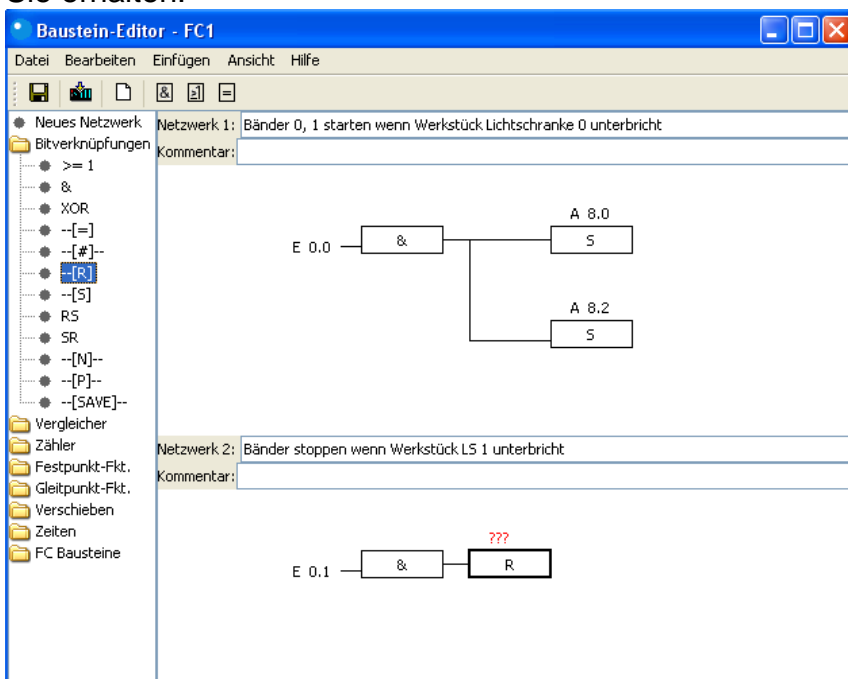


Abbildung 16 FUP-Ansicht UND-Gatter dessen Ausgang auf Reset-Gatter geschaltet

Die drei roten Fragezeichen weisen uns darauf hin, dass das Gatter noch nicht korrekt mit Eingaben versehen ist. Sie klicken auf die roten Fragezeichen und tippen in das Eingabefeld: „A 8.0“

Sie erhalten:

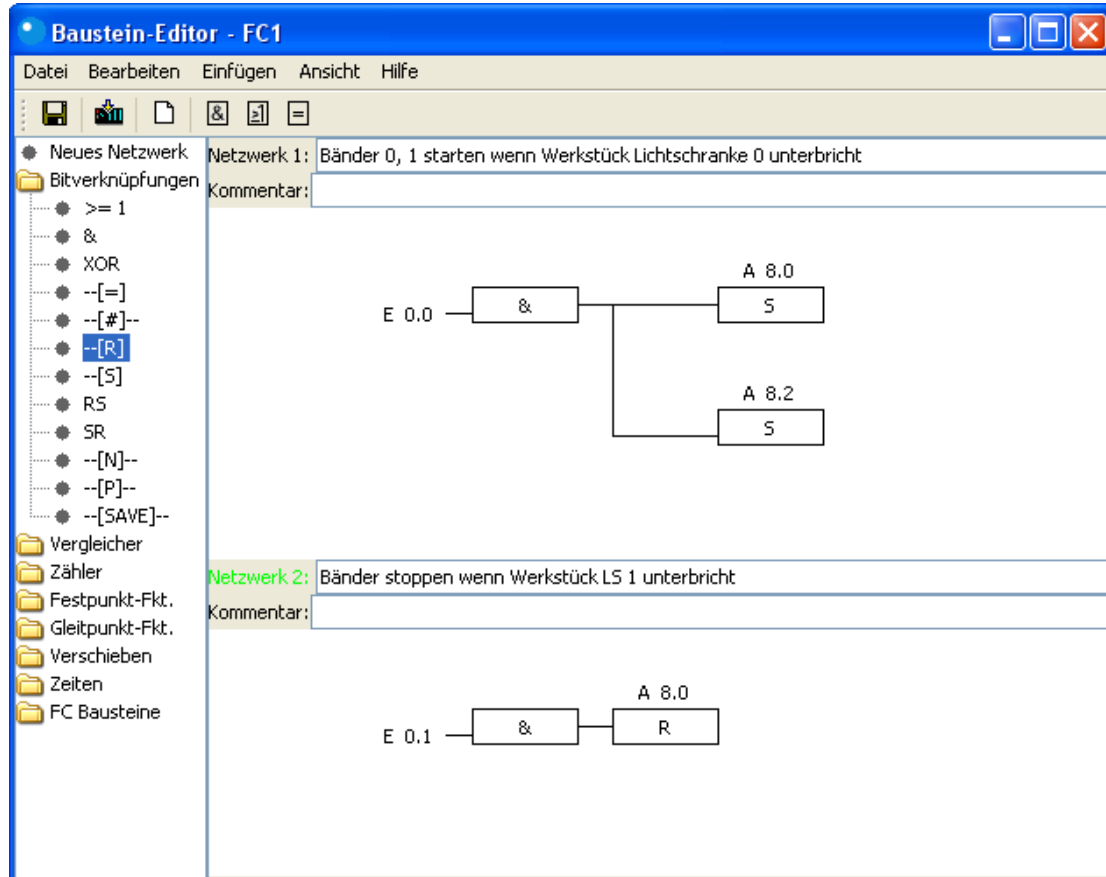


Abbildung 17

Nun fehlt noch der Reset von Ausgang 8.2.

Dazu klicken Sie wieder auf das Und-Gatter und anschließend fügen wie zuvor ein Reset-Gatter hinzu.

Sie erhalten:

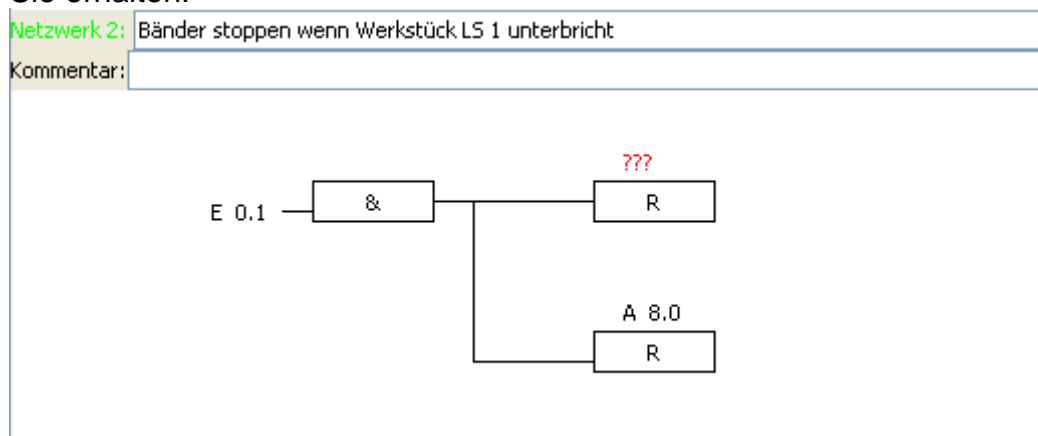


Abbildung 18

Auch hier ist noch die Bit-Adresse anzugeben, die zurückgesetzt werden soll. Sie klicken auf die drei Fragezeichen und tippen dort „A 8.2“ ein. Nun haben wir für die Funktion FC1 zwei Netzwerke.

Das erste Netzwerk dient zum Einschalten der Bänder-Motoren (siehe dazu Anhang), sobald ein Werkstück (WS) den ersten Licht-Taster (E 0.0) unterbricht. Das zweite Netzwerk dient zum Ausschalten der Förderband-Motoren sobald ein Werkstück (WS) den zweiten Licht-Taster (E 0.1) unterbricht.

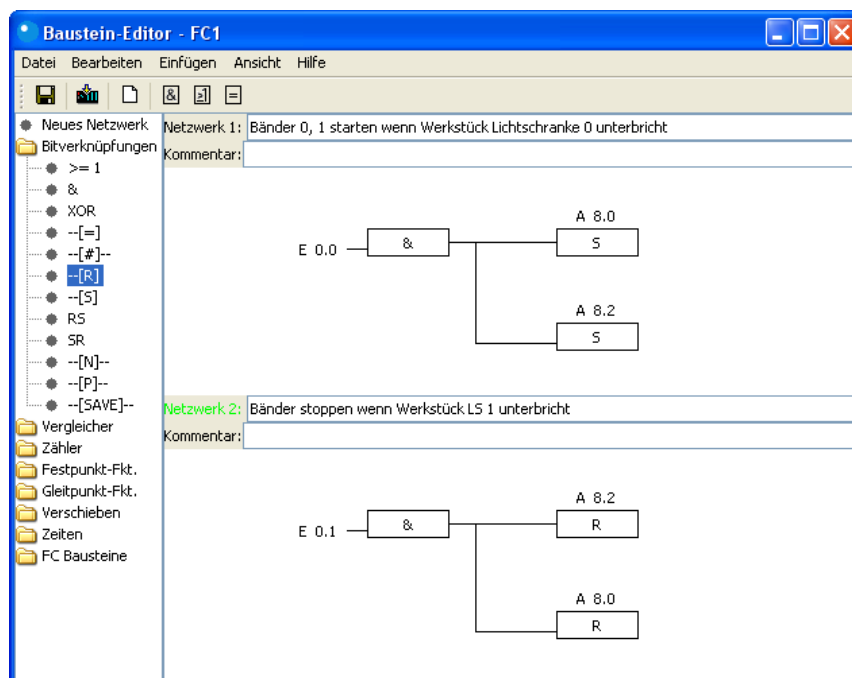


Abbildung 19: Ansicht Baustein Editor - editiert wird FC1 mit zwei Netzwerken in FUP-Ansicht.

Sie können zur AWL-Ansicht umschalten und erhalten:

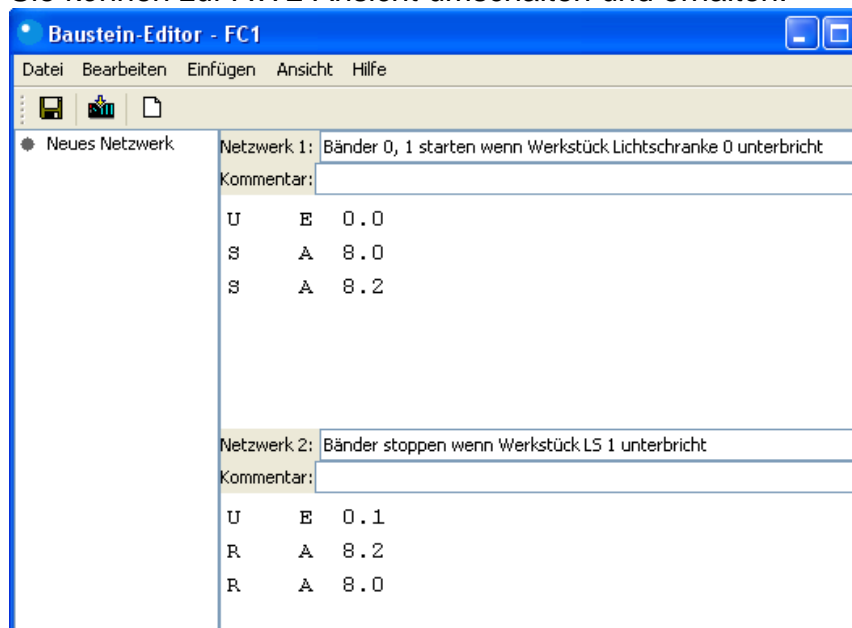


Abbildung 20: Ansicht - Baustein Editor editiert wird FC1 mit zwei Netzwerken in AWL-Ansicht

Hinweise:

1. Die AWL-Kommentare, die zuvor für Netzwerk 1 geschrieben wurden, gehen beim Umschalten nach FUP verloren.
2. Das Umschalten von FUP nach AWL ist immer möglich. Der umgekehrte Weg (AWL-> FUP) ist nur unter bestimmten syntaktischen Voraussetzungen möglich.

Schließen Sie nun den Baustein-Editor, bestätigen Sie die Abfrage mit [Ja], so dass die Änderungen im Projekt gespeichert werden.

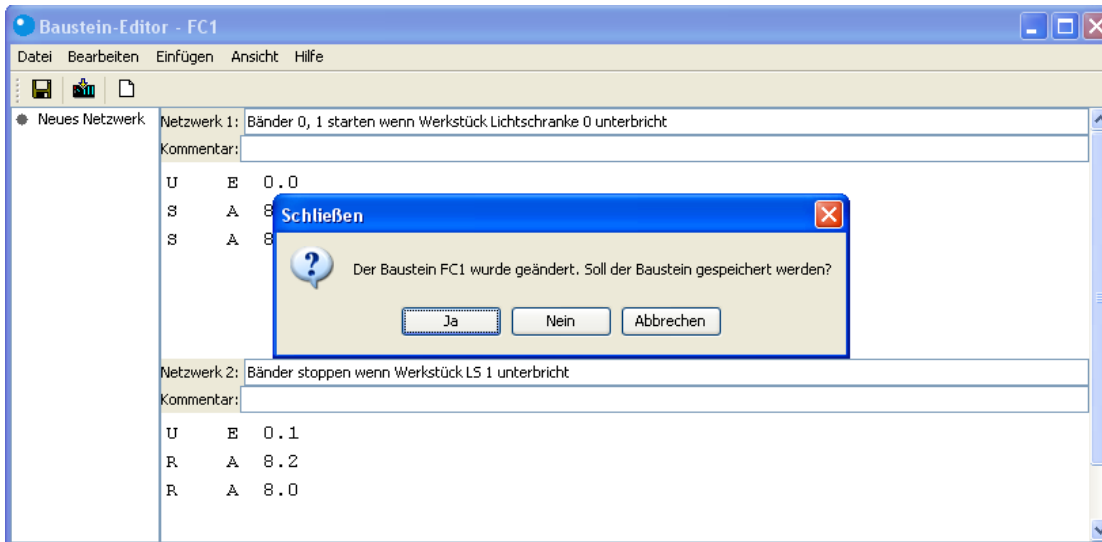


Abbildung 21: Dialog beim Schließen des Baustein-Editors

### 1.3 Aufrufen von Bausteinen

Bisher ist die Funktion FC1 zwar im Projekt vorhanden – sie wird jedoch noch nicht aufgerufen und ist daher nicht wirksam. Dies soll nun geändert werden.

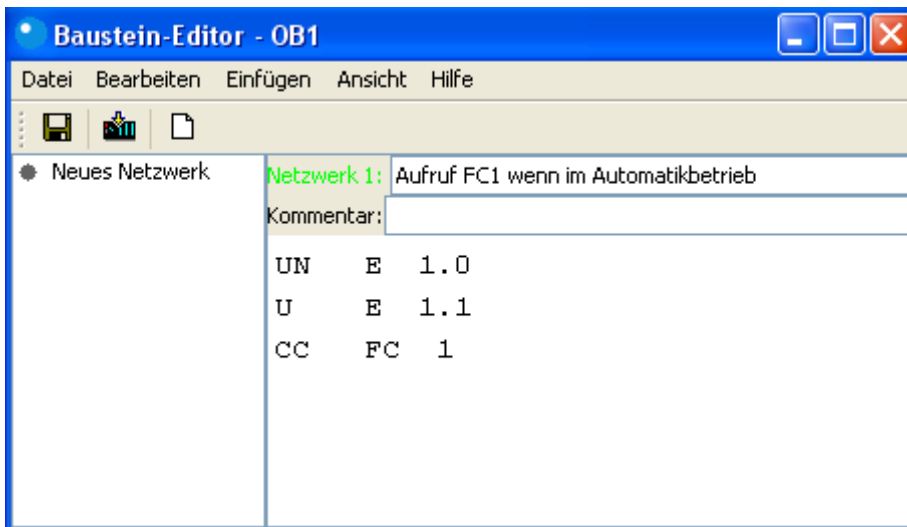
Daher ändern wir OB1 und rufen FC1 nur dann auf, wenn beim Bandmodel-Kontrollpult:

- der Schalter auf „automatisch“ (E 1.1) steht
- und der Schalter nicht auf „manuell“ (E 1.0) steht.

Zur Belegung der Ein- und Ausgänge (siehe Kapitel 3)

Öffnen Sie dafür den OB1, schalten um auf AWL und schreiben:

UN	E	1.0
U	E	1.1
CC	FC	1



Schließlich beenden Sie den Editor für OB1.

## 1.4 Starten der Bandmodell Simulation

Nun soll das Programm getestet werden. Dazu klicken wir auf das Icon („Bandmodell-Simulation starten“).

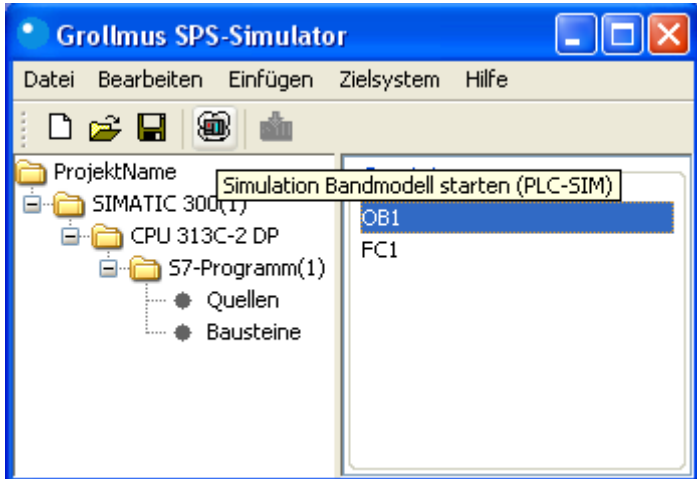


Abbildung 22: SPS-Simulator - mit Schaltfläche zum Starten der Bandmodell-Simulation

Daraufhin erscheint ein neues Fenster:

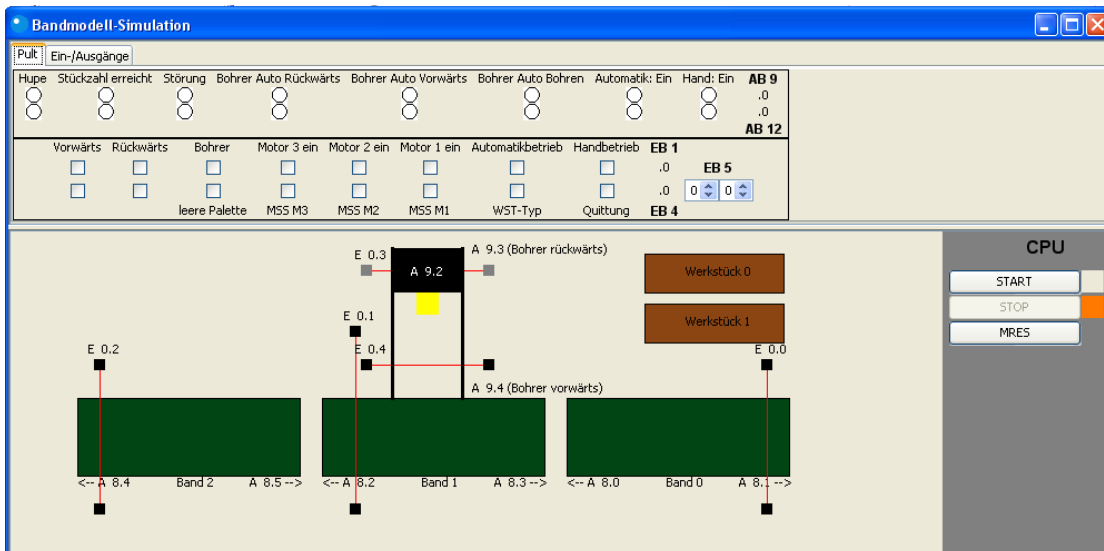


Abbildung 23: Bandmodell-Simulation

Nun ist es möglich das Programm in die CPU zu übertragen. Dieser Schritt wird im nächsten Abschnitt erläutert.

## 1.5 Übertragen des Programms in die CPU

Nun nachdem die Simulation gestartet wurde, kann das Programm in die simulierte CPU übertragen werden. Dazu klicken Sie entweder auf das Icon „Laden“ in der Toolbar oder rufen es über das Menü Zielsystem->Laden auf.

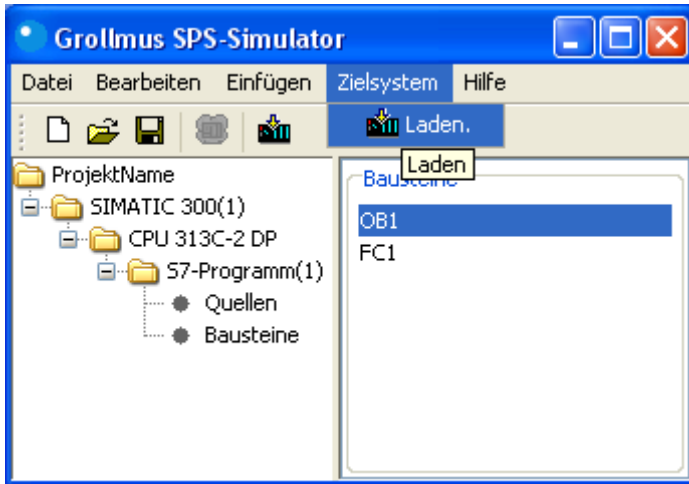


Abbildung 24: Übertragung des Programms in die simulierte CPU

Auch im Baustein-Editor existiert dieses Icon. In diesem Falle wird nur das Programm des jeweiligen Bausteins in die CPU übertragen. Somit haben Sie die Möglichkeit während, des Schreibens des Programms dieses im Simulator zu testen.

Da die CPU bisher nicht gestartet wurde, sehen sie bisher keine Reaktionen. Dieser Schritt wird im nächsten Abschnitt beschreiben.

## 1.6 Starten der CPU

Damit das Programm ausgeführt wird, muss die CPU im „Run-Modus“ sein. Nach dem Starten der Bandmodell-Simulation ist die CPU noch im Stop-Modus.

Um die CPU zu veranlassen das Programm zyklisch abzuarbeiten drücken Sie im Fenster der Bandmodell-Simulation auf die Schaltfläche [Start].

Wenn Sie nun den Schalter auf dem Kontrollpult auf Automatikbetrieb stellen und eines der beiden Werkstücke (via Drag & Drop) auf das Band 0 legen, so dass der Licht-Taster, der auf Band 1 montiert ist, unterbrochen wird, dann laufen Band 1 und Band 2 an und transportieren das Werkstück.

Dies ist das gewünschte Ergebnis von Netzwerk 1 im FC1.

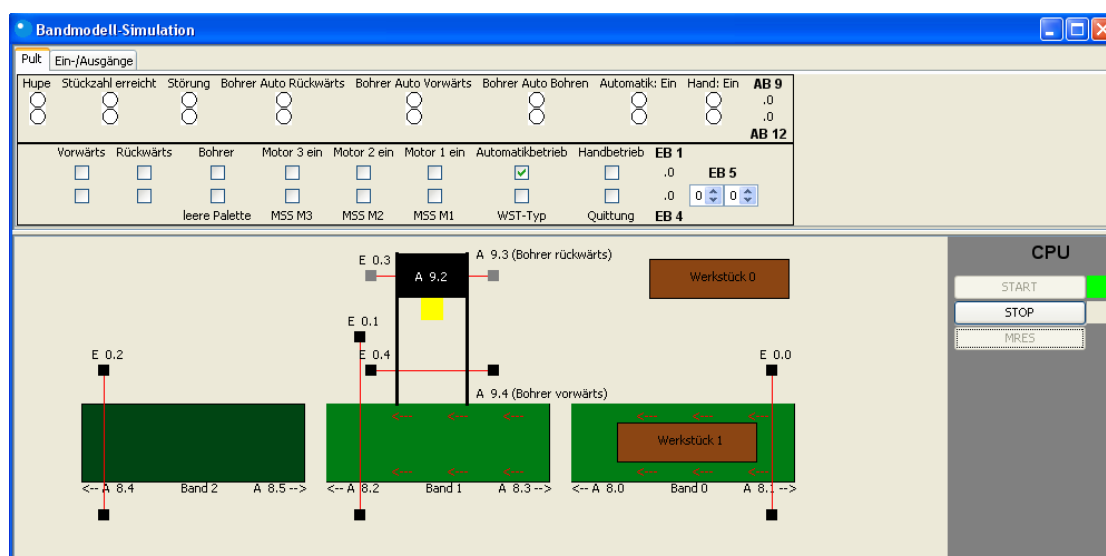


Abbildung 25 gestartete simulierte CPU im Bandmodell – Werkstück 1 wird weiter transportiert

Wenn dann das Werkstück die Lichttaster an Band 2 montiert unterbricht, stoppen beide Bänder. Dies ist das gewünschte Ergebnis von Netzwerk 2 von FC1.

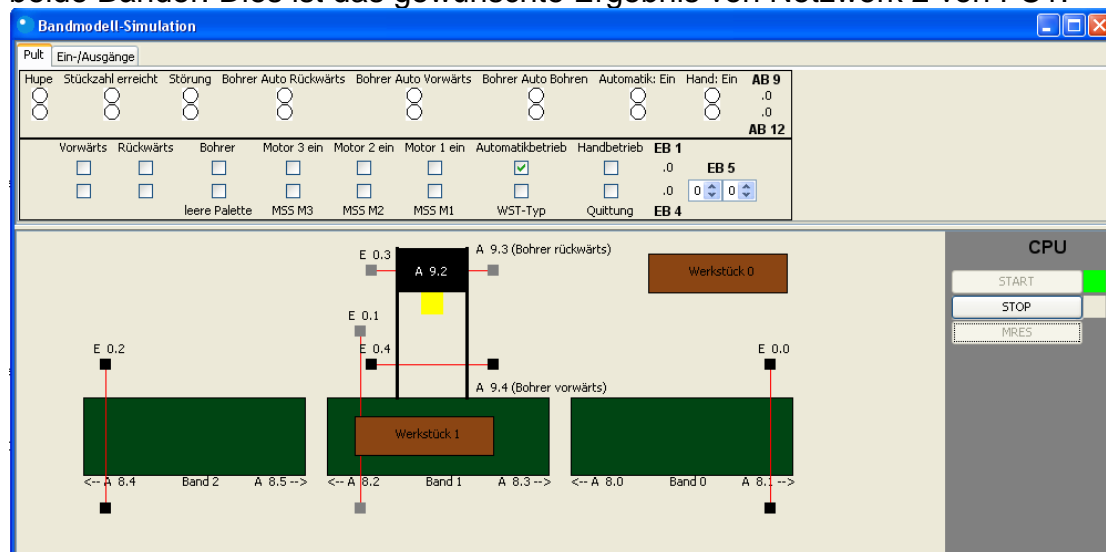


Abbildung 26: Bandmodell - Werkstück stoppt in Bohrposition.

## 1.7 Erweitern des Programms

Nun soll das Programm dahingehend erweitert werden, dass die Lampe „Störung“ aufleuchtet, wenn sowohl der Schalter „Handbetrieb“ als auch der Schalter „Automatikbetrieb“ auf „ein“ stehen.

Die Logik dafür soll in OB1 geschrieben werden. Öffnen Sie dafür den OB1, fügen ein neues Netzwerk hinzu und schreiben:

```
U E 1.0
U E 1.1
= A 9.5
```

Anschließend drücken Sie in der Toolbar auf die Schaltfläche „Laden“ um das aktuell editierte Programm in die im Run-Modus befindliche CPU zu übertragen.

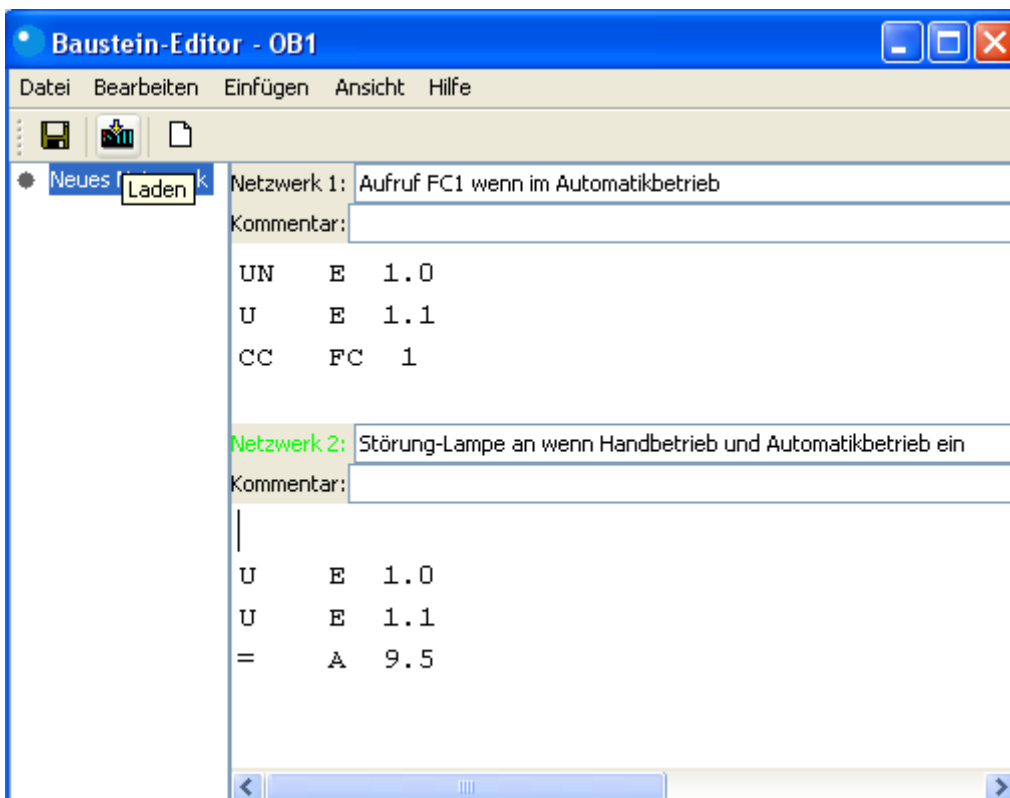


Abbildung 27: Laden des modifizierten Bausteins vom Baustein-Editor in CPU

Wenn in der Bandmodell-Simulation auch der Schalter „Handbetrieb“ auf „ein“ gestellt wird, so leuchtet die Lampe „Störung“ auf.

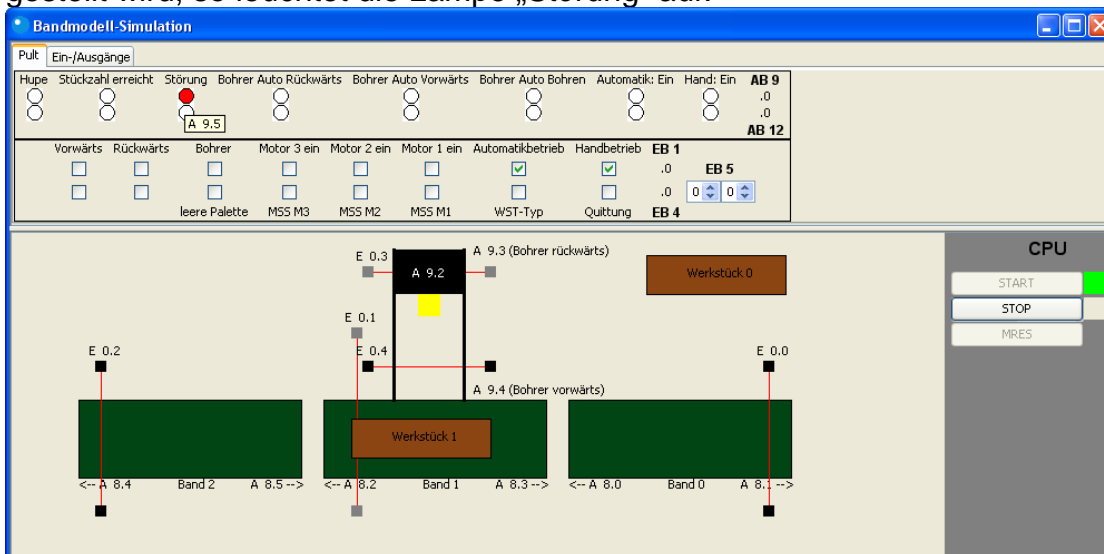


Abbildung 28: Test von Netzwerk 2 von OB1

Beachten Sie dass, wenn die Maus über die Lampe „Störung“ steht, dass dort ein Hilfetext (Tooltip) mit der entsprechenden Adresse erscheint. Ein entsprechendes Verhalten gibt es bei allen anderen Schaltern und Lampen.

Wechseln Sie nun vom Reiter „Pult“ zum Reiter „Ein-/Ausgänge“. Sie können hier die Ein- und Ausgänge kontrollieren und auch steuern.

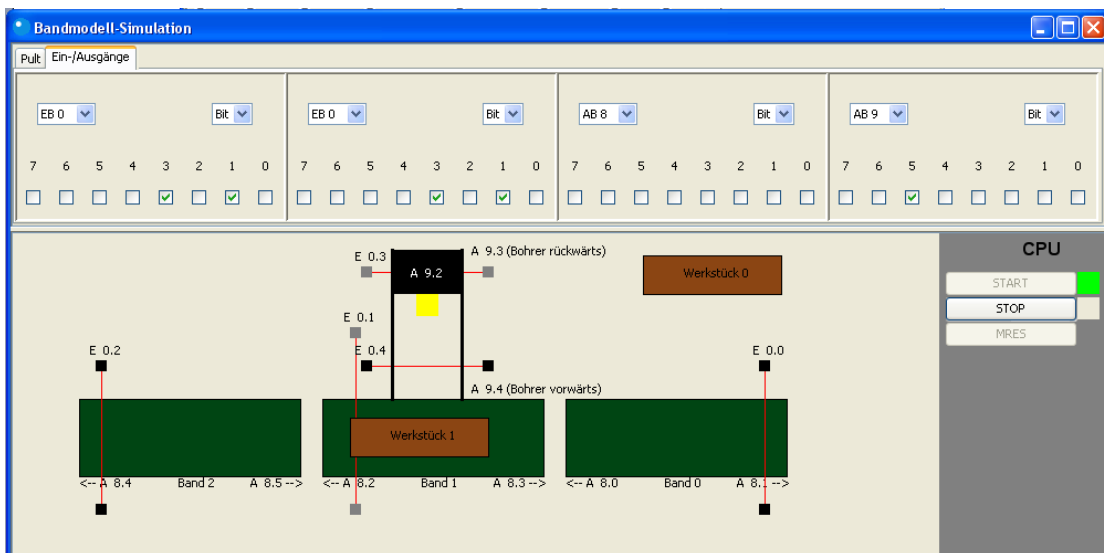


Abbildung 29: Bandmodell Simulation Ansicht Ein-/Ausgänge

Beachten Sie dabei Folgendes:

Eingänge, die vom Bandmodell verwendet werden, sind für die Eingabe „blockiert“. Sie werden vor jedem Zyklus der CPU aus dem Bandmodell eingelesen. Wenn Sie beispielsweise versuchen in der oben (Abbildung 29) dargestellten Situation den Eingang E 0.1 auf 0 zu setzen, funktioniert dies nicht, weil der Eingang durch den Licht-Taster belegt ist.

Wenn hingegen ein Ausgang vom Programm nicht angesteuert wird, so ist es möglich, diesen zu verändern. Wenn sie beispielsweise Ausgang 9.2 im „Handbetrieb“ auf 1 setzen, so läuft das Band 1 rückwärts.

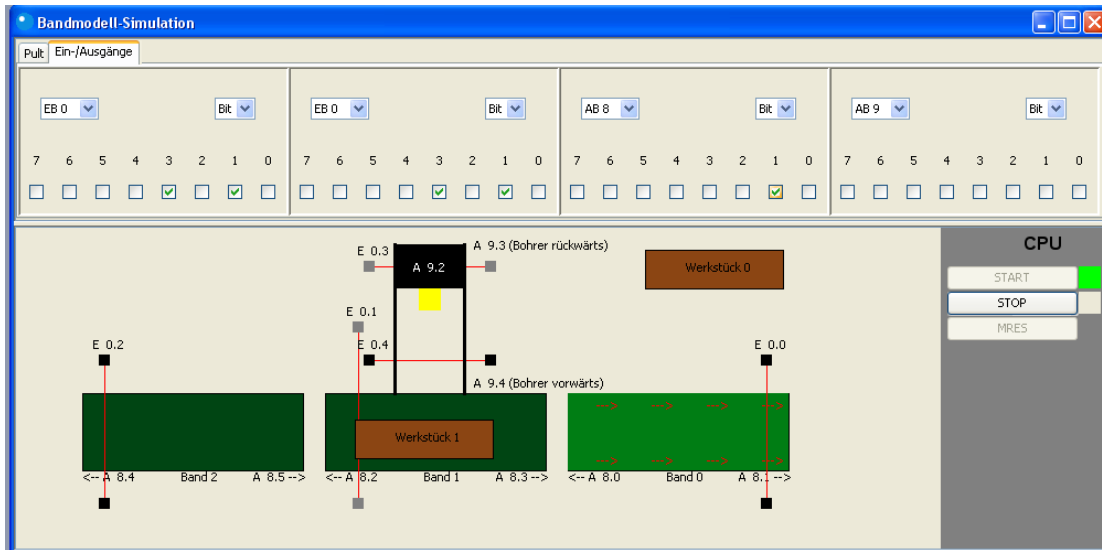


Abbildung 30: Ansteuerung Ein- und Ausgänge über Checkboxes.

## 1.8 Speichern und Laden des Projektes

Im letzten Teil dieses Tutorials soll gezeigt werden, wie Sie das Projekt speichern und wieder laden können.

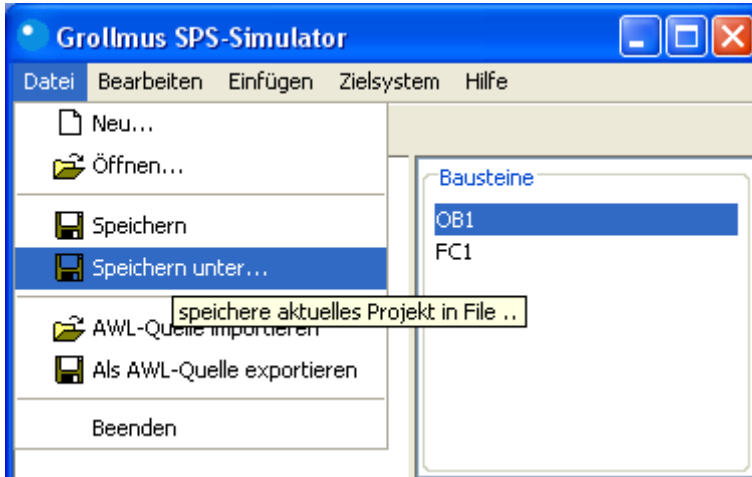


Abbildung 31: Speichern des Projektes

Zum Speichern schließen Sie den Baustein-Editor und rufen entweder die Schaltfläche [Speichern] in der Toolbar oder den Menüeintrag „Datei -> Speichern“ auf. Vergeben sie einen passenden Namen wie z.B. „tutorial.xml“.

Schließen Sie nun alle noch ggf. offenen Fenster des Simulators. (Sie brauchen dazu nur das „Hauptfenster“ zu schließen)

Wenn Sie den SPS-Simulator erneut starten, können Sie über das Menü „Datei -> Öffnen“ Ihre gespeicherte Datei auswählen und damit Ihr bisheriges Projekt laden.

## 2. Zulässige AWL-Befehle

Der SPS-Simulator ermöglicht sowohl die Entwicklung in AWL als auch in FUP. Die simulierte CPU jedoch kann nur bestimmte AWL-Befehle interpretieren. Nicht alle existierende AWL-Befehle können interpretiert werden.

Nachfolgend eine Liste der implementierten Befehle:

<b>Bitverknüpfungsbefehle</b>	
U	UND-Verknüpfung
UN	UND-NICHT-Verknüpfung
O	ODER-Verknüpfung
ON	ODER-NICHT-Verknüpfung
X	EXKLUSIV-ODER-Verknüpfung
XN	EXKLUSIV-ODER-NICHT-Verknüpfung
U(	UND-Klammer-Auf
UN(	UND-NICHT-Klammer-Auf
O(	ODER-Klammer-Auf
ON(	ODER-NICHT-Klammer-Auf
X(	EXKLUSIV-ODER-Klammer-Auf
XN(	EXKLUSIV-ODER-NICHT-Klammer-Auf
)	Klammer-Zu
O	ODER-Verknüpfung von UND-Funktionen
=	Zuweisung des VKE (Binäre Zuweisung)
S AB <ByteNr.BitNr>	Setze Ausgangs-Bit-Operation (Binäre bedingte Zuweisung)
R M 1.1	Rücksetz-Operation (Binäre bedingte Zuweisung)
FP	Positive Flankenauswertung
FN	Negative Flankenauswertung
CLR	Setzt VKE auf 0
SET	Setzt VKE auf 1
NOT	Negiert das VKE
SAVE	Rettet das VKE in das BIE-Bit

<b>Zeitfunktionen</b>	
SI	Impuls
SV	Verlängerter Impuls
SE	Einschaltverzögerung
SS	Speichernde Einschaltverzögerung
SA	Ausschaltverzögerung

<b>Zählerbefehle</b>	
ZV	Zähle vorwärts (bei positiver Flanke)
ZR	Zähle rückwärts (bei positiver Flanke)
S Z <nr>	Setze Zählerwert auf ... (Akku1) (bei positiver Flanke)
R Z <nr>	Setze Zählerwert auf 0 (bei VKE = 1)
L Z <nr>	Lade Zählerstand in Akku1 (dual-codiert)
LC Z <nr>	Lade Zählerstand in Akku1 (bcd-codiert)

<b>Lade- und Transfer-Operationen</b>	
L [E A M][B W D] <ByteNr>	Lade Bit-Muster aus Quell-Adresse in Akku 1
T [E A M][B W D] <ByteNr>	Transferiere Bit-Muster aus Akku 1 in Zieladresse
L <nr>	Lade Ganzzahl (Konstante) in Akku 1
L C#<nr>	Lade Zählerkonstante in Akku 1 (BCD-codiert)
L S5T#<...>	Lade Timer-Konstante in Akku 1
L <realzahl>	Lade Gleitkommazahl in Akku 1
L B#(nr,nr)	Lade eine Konstante als 2 Bytes in Akku 1 z.B. ‚L B#(1,10)‘
L B#(nr,nr,nr,nr)	Lade eine Konstante als 4 Bytes in Akku 1 z.B. ‚L B#(1,10,5,4)‘
L W#16#<hex>	Lade Hexadezimalkonstante (16 Bit) in Akku 1
L DW#16#<hex>	Lade Hexadezimalkonstante (32 Bit) in Akku 1

<b>Akku-Vergleichsoperationen (Akku 1 &lt;-&gt; Akku 2)</b>	
==I	Akku 1 gleich Akku 1 (16 Bit Integer)
<>I	... ungleich
<I	kleiner
<=I	kleiner gleich
>I	größer
>=I	größer gleich
==D	Akku 1 gleich Akku 1 (32 Bit Integer)
<>D	... ungleich
<D	kleiner
<=D	kleiner gleich
>D	größer
>=D	größer gleich
==R	Akku 1 gleich Akku 1 (32 Bit Real)
<>R	... ungleich
<R	kleiner
<=R	kleiner gleich
>R	größer
>=R	größer gleich

<b>Arithmetische Operationen (Akku 2 &lt;op&gt; Akku 1)</b>	
+I	Addiere zwei 16 Bit Integerzahlen (Akku 2 + Akku 1)
-I	Subtrahiere zwei 16 Bit Integerzahlen (Akku 2 - Akku 1)
*I	Multipliziere zwei 16 Bit Integerzahlen (Akku 2 * Akku 1)
/I	Dividiere zwei 16 Bit Integerzahlen (Akku2 / Akku 1)
+D	Addiere zwei 32 Bit Integerzahlen (Akku 2 + Akku 1)
-D	Subtrahiere zwei 32 Bit Integerzahlen (Akku 2 - Akku 1)
*D	Multipliziere zwei 32 Bit Integerzahlen (Akku 2 * Akku 1)
/D	Dividiere zwei 32 Bit Integerzahlen (Akku2 / Akku 1)
MOD	Rest der Division zweier 32 Bit Integerzahlen (Akku2 % Akku1)
+R	Addiere zwei 32 Bit Realzahlen (Akku 2 + Akku 1)
-R	Subtrahiere zwei 32 Bit Realzahlen (Akku 2 - Akku 1)
*R	Multipliziere zwei 32 Bit Realzahlen (Akku 2 * Akku 1)
/R	Dividiere zwei 32 Bit Realzahlen (Akku2 / Akku 1)

<b>Arithmetische Operationen 32 Bit Realzahlen</b>	
NEGR	Negation einer Realzahl
ABS	Betrag einer Realzahl
SQRT	Quadratwurzel einer Realzahl
SQR	Quadrat einer Realzahl

<b>Akku1 Umwandlungs-Operationen</b>	
BTI	wandelt eine BCD-Zahl in eine 16 Bit Ganzzahl
ITB	wandelt eine 16 Bit Ganzzahl in eine BCD-Zahl
BTD	wandelt eine BCD-Zahl in eine 32 Bit Ganzzahl
ITD	wandelt eine 16 Bit Ganzzahl in eine 32 Bit Ganzzahl
DTB	wandelt eine 32 Bit Ganzzahl in eine BCD-Zahl um
DTR	wandelt eine 32 Bit Ganzzahl in eine Gleitpunktzahl um

<b>Call -Operationen</b>	
CALL	Unbedingter Aufruf Baustein
UC	Unbedingter Aufruf Baustein
CC	Conditional Call (Bedingter Aufruf eines Bausteins)

<b>Baustein-Ende-Operationen</b>	
BE	Baustein Ende
BEA	Baustein Ende (Absolut)
BEB	Bedingtes Baustein-Ende

<b>Strukturierungs-Befehle (Notwendig für Umwandlung AWL &lt;-&gt; FUP)</b>	
NOP 0	No Operation / Strukturierung
BLD 101	Strukturierung

### 3. Belegung der Ein- und Ausgänge des Bandmodells

<b>Ein- bzw. Ausgang</b>	<b>Bit-Adresse</b>
Licht-Taster, Position 1	E 0.0
Licht-Taster, Position 2	E 0.1
Licht-Taster, Position 3	E 0.2
Endschalter, Bohrwerk hinten	E 0.3
Endschalter, Bohrwerk vorne	E 0.4
Handbetrieb	E 1.0
Automatikbetrieb	E 1.1
Motor 1 Ein	E 1.2
Motor 2 Ein	E 1.3
Motor 3 Ein	E 1.4
Bohrer bohren im Handbetrieb	E 1.5
Bohrer rückwärts im Handbetrieb	E 1.6
Bohrer vorwärts im Handbetrieb	E 1.7
Quittung	E 4.0
WSt Typ	E 4.1
MSS M1	E 4.2
MSS M2	E 4.3
MSS M3	E 4.4
Leere Palette	E 4.5
Dezimalvorgabe, 1er-Stelle	E 5.0...E5.3
Dezimalvorgabe, 10er-Stelle	E 5.4...E5.7
Band 1 vorwärts	A 8.0
Band 2 vorwärts	A 8.2
Band 3 vorwärts	A 8.4
Bohrer ein	A 8.5
Bohrer rückwärts	A 8.6
Bohrer vorwärts	A 8.7
Hand ein	A 9.0
Automatik ein	A 9.1
Band 1 rückwärts	A 9.2
Band 2 rückwärts	A 9.3
Band 3 rückwärts	A 9.4
Störung	A 9.5
Stückzahl erreicht	A 9.6
Hupe	A 9.7