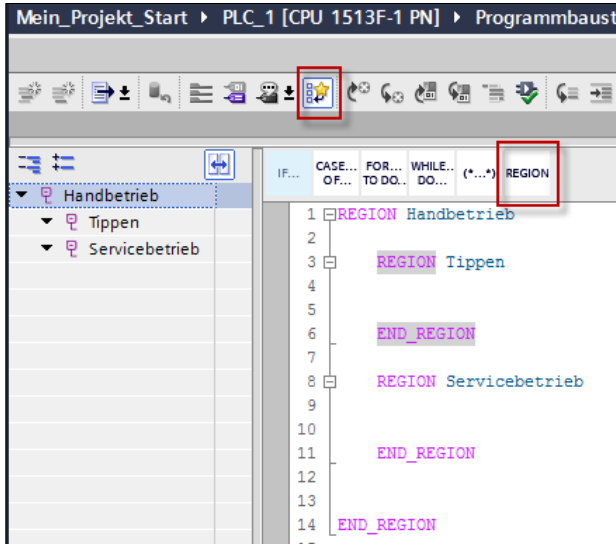
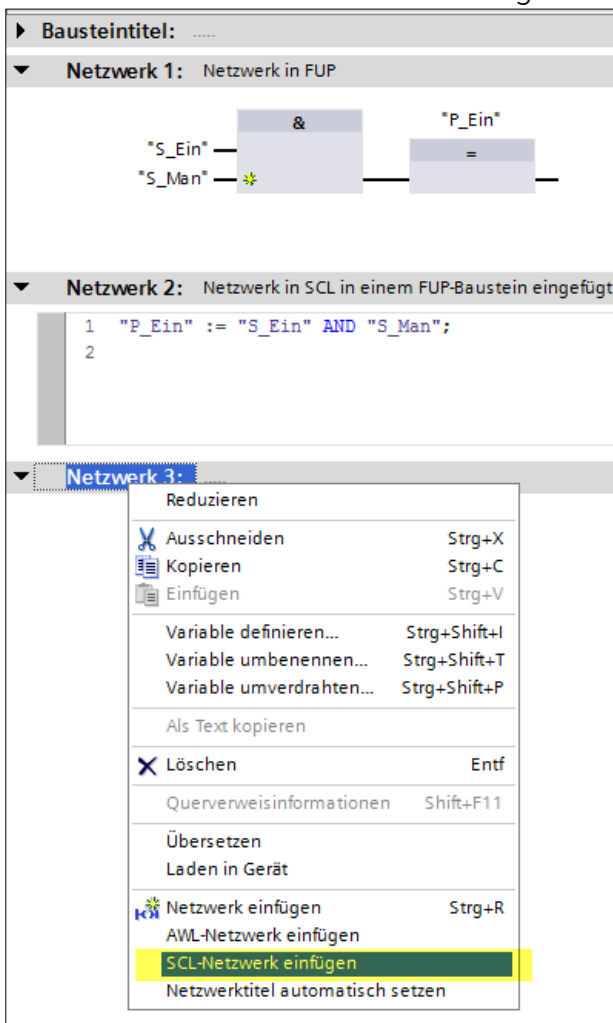


## Neuerungen im SCL ab V14 SP1

### 1. "Regions" zur besseren Strukturierung und Navigation



### 2. SCL-Netzwerke in KOP / FUP einfügen



3. strukturierte Variablen und ARRAY-Elemente per Drag&Drop ersetzen

```

1
2 "Wertespeicher_10"."THIS"[2] := "Wertespeicher"."THIS"[0];
3
4

```

2.warten 1.warten

4. Mehrfachzuweisungen

```

1 //Reset values on startup
2 #FillLevelCurrent := #FillLevelSetpoint := #TempCurrent := #TempSetpoint := 0.0;

```

5. Umschalten zwischen Einfüge- und Überschreibemodus

Einfügemodus aktiv (normal)

```
"P_Ein" := "S_Ein" AND "S_Man";
```

Überschreibemodus aktiv

```
"P_Ein" := "S_Ein" AND "S_Man";
```

Die Umschaltung erfolgt mit der Taste „Insert“

6. Selektierten Programmcode mit Strukturelementen umschliessen.

Netzwerk 2: Netzwerk in SCL in einem FUP-Baustein eingefügt.

Kommentar

```

1 "P_Ein" := "S_Ein" AND "S_Man";
2 "P_Ein" := "S_Ein" AND "S_Man";

```

```

1 IF condition THEN
2     "P_Ein" := "S_Ein" AND "S_Man";
3     "P_Ein" := "S_Ein" AND "S_Man";
4 END_IF;
5

```

7. STRG + Klick für Gehe zu Definition

```

8 END_IF;
9
10 #temp1 := "DeInkrement"(iValue := #temp1 , bDekrement := #toggle);
11 "dq_outb0" := #temp1;
12 "dq_outb0" := MIN(IN1 := "dq_outb0", IN2 := #temp1);
    
```

8. Array(\*) zur Übergabe variabel langer Arrays

**Funktionen**

- Array(\*) als neuer Parameter-Datentyp
- Neue Funktionen LOWER\_BOUND, UPPER\_BOUND zur Ermittlung der Array-Grenzen
- Für AWL, SCL, KOP, FUP

**Nutzen**

- Keine Pointer-Programmierung für Arrays unterschiedlicher Länge mehr notwendig
- Bessere Laufzeit Performance als Pointer (Any/Variant) und kopieren mittels MOVE\_BLK
- Vollsymbolisch und bessere Lesbarkeit
- Ermöglicht generische Standardfunktionen für Arrays unterschiedlicher Länge.

The screenshot shows a variable declaration table for 'InitArray' with columns for Name, Data type, and Default value. 'QuantityArray' is listed with data type 'Array[\*] of Int'. Below this, a code block shows the declaration of 'low' and 'up' bounds and a loop: 'FOR #i := #low TO #up DO #QuantityArray[#i] := 0; END\_FOR;'. At the bottom, a ladder logic diagram shows two 'InitArray' function blocks. The first block has an input '#Belt1Array' of type 'QuantityArray'. The second block has an input '#Belt2Array' of type 'QuantityArray'. Blue arrows point from the variable declarations to the corresponding inputs in the ladder logic.

9. Neue Arithmetik-Funktionen

+=, -=, \*=, /=

```

#PieceCount := #PieceCount + 1;
#PieceCount += 1;

#Factor := #Factor / 2;
#Factor /= 2;
    
```

## 10. Konstanten für Strings

Bei der Deklaration eines Strings können jetzt auch Konstanten (lokal, global) für die Länge verwendet werden.



## 11. Vergleich von UDT-Variablen

12			BeltMotor1	"MotorData"
13			BeltMotor2	"MotorData"

```

1 IF #BeltMotor1 = #BeltMotor2 THEN
2   (* do something *);
3 END_IF;
  
```



## 12. Quellen exportieren

