

# TUTORIAL STEP7

## TIA PORTAL V14

### MET S7-300

**DE HAAGSE**  
HOGESCHOOL

J.E.J. op den Brouw  
De Haagse Hogeschool  
Opleiding Elektrotechniek  
8 februari 2019

[J.E.J.opdenBrouw@hhs.nl](mailto:J.E.J.opdenBrouw@hhs.nl)

# VERSIEHISTORIE

Rev.	Datum	Aut.	Beschrijving
0.1	08-12-2017	JodB	Eerste opzet, concept, tutorial LAD en Graph
0.2	12-12-2017	JodB	Aanpassingen na opmerkingen Gerard
0.3	03-01-2018	JodB	Tutorial SCL, peripheral access, plaats van bestanden herschreven, bijlage met formules voor temperatuurmeting, schakelen tussen programmeertalen, import en export van bronbestanden, dupliceren project, flankdetectie in SCL
0.4	09-01-2018	JodB	Timers in SCL, counters in SCL, extra info formules temperatuurmeting, aanpassingen na opmerkingen Gerard en Ben
0.5	27-01-2018	JodB	Sequencers in SCL, kleine aanpassingen in de tekst, aanpassingen na opmerkingen Gerard
0.6	16-02-2018	JodB	Aanpassingen na opmerkingen Ben
0.61	04-02-2019	jodb	Tikfouten, logo aangepast
0.62	08-02-2019	JodB	Tekst over voorbeeld in bijlage A in rood afgedrukt

©2019, J. op den Brouw

Deze tutorial is tot stand gekomen dankzij de medewerking van Ben Kuiper (Elektrotechniek), Gerard Tuk (Mechatronica), Ernst Kouwe (Mechatronica) en Jon van den Helder (Technische Informatica).

Voor suggesties en/of opmerkingen over deze tutorial kan je je wenden tot J. op den Brouw, kamer D1.047, of je kunt email versturen naar [J.E.J.opdenBrouw@hhs.nl](mailto:J.E.J.opdenBrouw@hhs.nl).

# INHOUDSOPGAVE

<b>1</b>	<b>Inleiding</b>	<b>10</b>
<b>2</b>	<b>Practicumomgeving</b>	<b>12</b>
<b>3</b>	<b>De PLC</b>	<b>14</b>
3.1	Beschrijving PLC . . . . .	14
3.1.1	Processormodule CPU315F-2 PN/DP . . . . .	15
3.1.2	Voedingsmodule PS307 . . . . .	15
3.1.3	Digitale I/O module SM323 . . . . .	16
3.1.4	Analoge I/O module SM334 . . . . .	16
3.1.5	Simulatie I/O module SM374 . . . . .	17
3.2	Programmeertalen . . . . .	18
3.3	Gebruik PLC . . . . .	19
<b>4</b>	<b>Tutorial LAD</b>	<b>21</b>
4.1	Aanmaken nieuw project . . . . .	22
4.2	PLC configureren . . . . .	22
4.3	PLC-configuratie downloaden . . . . .	25
4.4	PLC tags invoeren . . . . .	28
4.5	Programma invoeren . . . . .	29
4.6	Laden PLC met programma . . . . .	31
4.7	Monitoren van het programma en variabelen . . . . .	32
<b>5</b>	<b>Tutorial S7-Graph</b>	<b>35</b>
5.1	Aanmaken nieuw project, PLC configuratie & download . . . . .	35
5.2	PLC tags invoeren . . . . .	35
5.3	Een blok vooraf . . . . .	35
5.4	Blokken, stappen, overgangen en acties . . . . .	35
5.5	Aanmaken functieblok . . . . .	37
5.6	Starten S7-Graph editor . . . . .	37
5.7	Invoeren eerste stap . . . . .	38
5.8	Invoeren volgende stappen . . . . .	39
5.9	Invoeren OB1 . . . . .	40
5.10	Laden PLC met programma . . . . .	43
5.11	Monitoren van het programma en variabelen . . . . .	44
<b>6</b>	<b>Tutorial SCL</b>	<b>46</b>
6.1	Analoge ingang en temperatuur meten . . . . .	46
6.2	Aanmaken nieuw project, PLC configuratie & download . . . . .	47
6.3	PLC tags invoeren . . . . .	47
6.4	De functie ConvertADCToTemp . . . . .	48

6.5	Compileren SCL-code . . . . .	50
6.6	De functie CheckMinMax . . . . .	51
6.7	De functie Alarms . . . . .	52
6.8	Het functieblok AverageTemp . . . . .	52
6.9	OB1 invoeren . . . . .	55
6.10	Laden PLC met programma . . . . .	57
6.11	Monitoren van programma en variabelen . . . . .	57
6.12	Variabelen manipuleren met watch table . . . . .	59
6.13	Automatische temperatuurregistratie . . . . .	61
6.14	Foutdetectie NTC-weerstand . . . . .	62
<b>7</b>	<b>Sequencers in SCL</b>	<b>64</b>
7.1	Opzet sequencer . . . . .	64
7.2	Voorbeeld van een sequencer . . . . .	65
7.3	Eerste binnenkomst en laatste verlaten van een stap . . . . .	67
7.4	Het gebruik van timers in een sequencer . . . . .	68
7.5	Parallele verwerking van sequencers . . . . .	72
7.6	Parallele verwerking binnen een sequencer . . . . .	73
<b>8</b>	<b>Tips, tricks &amp; troubleshoot</b>	<b>77</b>
8.1	Projecten archiveren . . . . .	77
8.2	Projecten inlezen . . . . .	78
8.3	PLC diagnostiek . . . . .	80
8.4	Clock Memory Byte . . . . .	82
8.5	Instellen maximum scan cycle tijd . . . . .	83
8.6	Peripheral access . . . . .	83
8.7	Schakelen tussen programmeertalen . . . . .	84
8.8	Importeren extern bronbestand . . . . .	85
8.9	Exporteren naar een extern bronbestand. . . . .	88
8.10	Dupliceren van een project . . . . .	88
8.11	Flankdetectie in SCL . . . . .	88
8.12	Timers in SCL . . . . .	90
	8.12.1 Simatic timers . . . . .	90
	8.12.2 IEC timers . . . . .	91
8.13	Counters in SCL . . . . .	93
	8.13.1 Simatic counters . . . . .	93
	8.13.2 IEC counters . . . . .	94
8.14	Wijzigen van de programmeertaal in Graph . . . . .	96
	<b>Bibliografie</b>	<b>97</b>
<b>A</b>	<b>LAD-programma verkeerslichten</b>	<b>98</b>
<b>B</b>	<b>Functie FC1 "ConvertADCToTemp"</b>	<b>102</b>
<b>C</b>	<b>Functie FC2 "CheckMinMax"</b>	<b>104</b>
<b>D</b>	<b>Functie FC3 "Alarms"</b>	<b>105</b>

E	Funcatieblok FB1 "AverageTemp"	106
F	OB35 "CyclicInterrupt"	108
G	PLC Blocks, adressen, memory	109
H	Formules temperatuurmeting	110

# LIJST VAN FIGUREN

1.1	Opzet PLC-systeem. . . . .	10
2.1	Engineering layout van de PC. . . . .	12
3.1	De PLC S7-315 met I/O-modules. . . . .	14
3.2	De processormodule CPU315F. . . . .	15
3.3	De voedingsmodule PS307 5 A. . . . .	15
3.4	Digitale I/O module SM323. . . . .	16
3.5	Bitverdeling van de 8-bits analoge module SM334. . . . .	16
3.6	Analoge I/O module SM334. . . . .	17
3.7	Simulatiemodule SM374. . . . .	18
3.8	De operating modes van de PLC. . . . .	19
4.1	Pictogram Tia Portal V14. . . . .	21
4.2	Openingsscherm Tia Portal V14. . . . .	21
4.3	Aanmaken nieuw project. . . . .	22
4.4	Nieuw project wordt aangemaakt. . . . .	22
4.5	Nieuw apparaat configureren. . . . .	22
4.6	Selecteren CPU-module. . . . .	23
4.7	Project view van de PLC-configuratie. . . . .	23
4.8	Volledige module-configuratie van de PLC. . . . .	24
4.9	Instellen IP-adres. . . . .	25
4.10	Instellen MPI-adres. . . . .	25
4.11	Compileren PLC-configuratie. . . . .	26
4.12	PLC-configuratie wordt gecompileerd. . . . .	26
4.13	Selecteren download naar PLC. . . . .	26
4.14	Selecteren PLC voor download. . . . .	27
4.15	Preview van de acties tijdens download naar PLC. . . . .	27
4.16	Resultaat van de download naar de PLC en de mogelijkheid om de PLC te starten. . . . .	28
4.17	Starten invoer PLC tags. . . . .	28
4.18	Alle PLC tags ingevoerd. . . . .	29
4.19	Aanmaken functie. . . . .	29
4.20	Functie FC1 aangemaakt en geopend. . . . .	30
4.21	Meest voorkomende componenten. . . . .	30
4.22	Invoeren PLC tags. . . . .	31
4.23	Het eerste netwerk volledig. . . . .	31
4.24	OB1 roept FC1 aan. . . . .	32
4.25	Selecteer OB1 en FC1 voor download. . . . .	32
4.26	Download-dialog. . . . .	32
4.27	Monitoren eerste netwerk. . . . .	33

4.28	Monitoren PLC tags. . . . .	34
5.1	PLC tags. . . . .	36
5.2	Aanmaken functie DrukknopConditioner (FC1). . . . .	36
5.3	Laddernetwerken FC1. . . . .	37
5.4	Aanmaken FB1. . . . .	38
5.5	Openingsscherm Graph editor. . . . .	38
5.6	Invoeren actie. . . . .	39
5.7	Invoeren overgangsconditie. . . . .	39
5.8	Eerste stap volledig. . . . .	40
5.9	Aanmaken nieuwe stap. . . . .	40
5.10	Volledige Graph-sequencer. . . . .	41
5.11	Compilatie is gelukt. . . . .	42
5.12	Invoeren OB1. . . . .	42
5.13	Aanmaken Data Block voor FB1. . . . .	42
5.14	Volledige OB1 voor aanroepen FC1 en FB1. . . . .	43
5.15	Download alle blokken. . . . .	44
5.16	Download preview. . . . .	44
5.17	Download preview. . . . .	45
5.18	PLC tags online bekijken. . . . .	45
6.1	Aansluitschema van de NTC- en serieweerstand. . . . .	46
6.2	PLC tags. . . . .	47
6.3	Aanmaken nieuwe functie ConvertADCToTemp (FC1). . . . .	48
6.4	Functie ConvertADCToTemp aangemaakt en de SCL-editor is gestart. . . . .	48
6.5	De interface van functie ConvertADCToTemp. . . . .	49
6.6	De SCL-code van functie ConvertADCToTemp. . . . .	49
6.7	Selecteren compilatie. . . . .	50
6.8	Voortgang compilatie. . . . .	50
6.9	Compilatie geslaagd. . . . .	50
6.10	Voorbeeld van een mislukte compilatie. . . . .	51
6.11	De SCL-code van functie CheckMinMax (FC2). . . . .	51
6.12	De SCL-code van functie Alarms (FC3). . . . .	52
6.13	Aanmaken van een array van structures. . . . .	53
6.14	Invullen van de structure. . . . .	53
6.15	De complete interface van functieblok AverageTemp (FB1). . . . .	54
6.16	De SCL-code van functieblok AverageTemp (FB1). . . . .	54
6.17	Functie ConvertADCToTemp toevoegen aan eerste rung OB1. . . . .	55
6.18	Functie ConvertADCToTemp geplaatst in eerste rung van OB1. . . . .	55
6.19	Functies CheckMinMax en Alarms geplaatst in OB1. . . . .	56
6.20	Aanmaken datablok voor functieblok AverageTemp. . . . .	56
6.21	Functieblok AverageTemp is in de vierde rung geplaatst. . . . .	57
6.22	De vierde rung compleet. . . . .	57
6.23	Downloaden van de blokken naar de PLC. . . . .	57
6.24	Download preview. . . . .	58
6.25	Monitoren van functie ConvertADCToTemp. . . . .	58
6.26	Monitoren van de PLC tags. . . . .	59
6.27	Monitoren van datablok AverageTemp_DB. . . . .	59

6.28	Toevoegen nieuwe watch table. . . . .	60
6.29	De ingevulde watch table. . . . .	60
6.30	Watch table monitort de variabelen. . . . .	61
6.31	Wijzigen van de waarde van een variabele. . . . .	61
6.32	Toevoegen cyclische interrupt OB35. . . . .	62
6.33	SCL-code OB35. . . . .	63
7.1	Declaratie van de stappen. . . . .	65
7.2	Declaratie van het datablok van de sequencer met timer. . . . .	70
7.3	Declaratie van het datablok van de sequencer met twee taken. . . . .	74
8.1	Archiveren project. . . . .	77
8.2	Project opslaan. . . . .	77
8.3	Bestandsnaam en doelmap opgeven. . . . .	78
8.4	Project wordt gearchiveerd. . . . .	78
8.5	Inlezen project. . . . .	78
8.6	Inlezen project. . . . .	78
8.7	Huidige project wordt afgesloten. . . . .	79
8.8	Selecteer het opgeslagen project. . . . .	79
8.9	Selecteer de doelmap. . . . .	79
8.10	Overschrijf de doelmap. . . . .	80
8.11	Het project wordt geopend. . . . .	80
8.12	Verbinding opzetten voor diagnostiek. . . . .	81
8.13	De verbinding is opgezet. . . . .	81
8.14	Inhoud diagnostische buffer. . . . .	82
8.15	Instellen clock memory byte. . . . .	82
8.16	Instellen maximum scan cycle tijd. . . . .	83
8.17	Foutieve invoer PLC tag met peripheral access. . . . .	84
8.18	Correcte invoer in de tags-tabel. . . . .	84
8.19	Aanroep van peripheral access in OB1. . . . .	84
8.20	Omzetten van een blok in LAD naar FBD. . . . .	85
8.21	Het blok in FBD. . . . .	85
8.22	Add new external file. . . . .	86
8.23	Selecteer extern bronbestand. . . . .	86
8.24	Omzetten van een extern bronbestand in een blok. . . . .	87
8.25	Waarschuwing overschrijven blok. . . . .	87
8.26	Voortgang conversie extern bronbestand naar blok. . . . .	87
8.27	Het blok is gegenereerd. . . . .	88
8.28	Selectie te exporteren blok. . . . .	89
8.29	Specificeer de naam van het te exporteren blok. . . . .	89
8.30	Huidige en voorgaande waarden bij flankdetectie. . . . .	90
8.31	SCL-code voor het detecteren van een opgaande en neergaande flank. . . . .	90
8.32	Tags-tabel voor gebruik van een timer. . . . .	91
8.33	SCL-code voor het instantiëren en gebruiken van een Simatic timer. . . . .	91
8.34	Tags-tabel voor gebruik van een IEC timer. . . . .	92
8.35	Aanmaken van een datablok voor gebruik van een IEC timer. . . . .	92
8.36	SCL-code voor het instantiëren en gebruiken van een IEC timer. . . . .	92
8.37	Tags-tabel voor gebruik van een counter. . . . .	93



8.38 SCL-code voor het instantiëren en gebruiken van een counter. . . . .	94
8.39 Tags-tabel voor gebruik van een IEC counter. . . . .	94
8.40 Aanmaken van een datablok voor gebruik van een IEC counter. . . . .	95
8.41 SCL-code voor het instantiëren en gebruiken van een IEC counter. . . . .	95
8.42 Selectie van de taal van de overgangscondities bij een Graph-programma. . .	96

## LIJST VAN TABELLEN

3.1 Enkele spanningen en stromen bij analoge modules. . . . .	17
8.1 Frequenties en periodetijden van de clock memory byte. . . . .	83

## LISTINGS

7.1 Opzet sequencer in SCL. . . . .	64
7.2 Code van de functie Sequencer. . . . .	66
7.3 Voorbeeld van het weglaten van de toekenning van een nieuwe stapwaarde. . .	67
7.4 Opzet sequencer met detectie binnenkomst en verlaten. . . . .	68
7.5 Code van de functie SequencerEntryExit. . . . .	69
7.6 Code van het functieblok SequencerIECTimer. . . . .	71
7.7 Opzet parallelle sequencers. . . . .	73
7.8 Code van het functieblok SequencerSimultaneous (deel 1). . . . .	75
7.9 Code van het functieblok SequencerSimultaneous (deel 2). . . . .	76
B.1 Code van de functie ConvertADCToTemp (deel 1). . . . .	102
B.2 Code van de functie ConvertADCToTemp (deel 2). . . . .	103
C.1 Code van de functie CheckMinMax. . . . .	104
D.1 Code van de functie Alarms. . . . .	105
E.1 Code van het functieblok AverageTemp (deel 1). . . . .	106
E.2 Code van het functieblok AverageTemp (deel 2). . . . .	107
F.1 Code van Organization Block CyclicInterrupt. . . . .	108

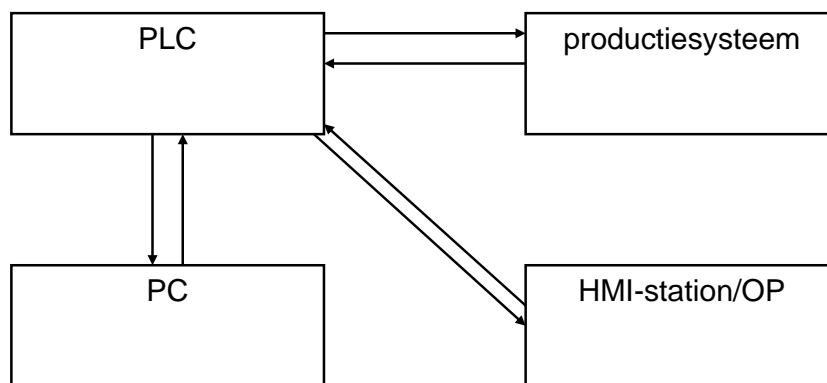
# 1. INLEIDING

Het zal de meeste mensen niet opvallen, maar veel van onze producten worden gemaakt in fabrieken. We denken hierbij aan bijvoorbeeld mobiele telefoons, microprocessorsen maar ook colaflessen (en het vullen ervan).

Het vervaardigen van deze producten wordt, omdat er massaproductie nodig is, gedaan door middel van een productielijn, in de volksmond lopende band genoemd. Vroeger waren hier grote groepen arbeiders voor nodig, tegenwoordig worden veel van deze klussen gedaan door machines.

Deze machines moeten bestuurd worden. Het besturen wordt onder andere gedaan door Programmable Logic Controllers, afgekort tot PLC<sup>1</sup>. Een PLC is te beschouwen als een computer met gespecialiseerde I/O. De computer, bestaande uit een microprocessor en geheugen, is te programmeren. De I/O wordt gebruikt om de machines aan te sturen en informatie betreffende de voortgang van het proces te vergaren.

De Faculteit voor Technologie, Innovatie en Samenleving / Delft beschikt over een schaalmodel van een productiesysteem van Festo en PLC's van Siemens. Deze beide componenten worden verderop toegelicht. Zie figuur 1.1.



**Figuur 1.1:** Opzet PLC-systeem.

Een korte introductie van de blokken:

- PLC (Programmable Logic Controller), de aanstuurder van (een deel van) het productiesysteem;
- Productiesysteem, de te besturen productieomgeving, bv. van de firma Festo,
- PC, de computer waarop de software voor de PLC wordt ontwikkeld, tevens monitoring systeem;

<sup>1</sup> In de Verenigde Staten wordt veelal de term Programmable Controller gebruikt, afgekort tot PC. Dit geeft echter verwarring omdat een ander veel voorkomend apparaat, de Personal Computer, óók wordt afgekort tot PC.

- HMI-station/Operator Panel, een terminal waarop de engineer/operator het productieproces kan volgen.

Tussen de diverse blokken is communicatie mogelijk:

- PLC  $\longleftrightarrow$  Productiesysteem, meetsignalen en stuursignalen (gezien vanuit de PLC);
- PLC  $\longleftrightarrow$  PC, programmeerkabel, communicatie d.m.v. MPI of Ethernet. Deze kabel kan voor meer doeleinden gebruikt worden, maar in eerste instantie voor programmeren;
- PLC  $\longleftrightarrow$  HMI-station/OP, communicatie d.m.v. MPI, Profibus en Profinet.

### Over de tutorials

Deze handleiding begeleidt de lezer aan de hand van diverse tutorials door de veelzijdigheid van de TIA Portal-programmeeromgeving. De eerste tutorial laat een ladderprogramma zien voor een eenvoudig verkeerslichtsysteem. In de tweede tutorial wordt het programma opnieuw ingevoerd, maar nu in S7-Graph. De derde tutorial betreft iets geheel anders; met behulp van een NTC-weerstand en een weerstand wordt een thermometer ontwikkeld. Het bijbehorende programma is in SCL geschreven. Daarna volgt een hoofdstuk over het implementeren van sequencers in SCL.

### Hoe moet dit document gelezen worden

In de eerste twee tutorials wordt de lezer “aan de hand meegenomen”. Van bijna alles wat de lezer moet invullen of aanklikken is een screenshot weergegeven. Vanwege de layout zijn sommige screenshots na de begeleidende tekst geplaatst, bijvoorbeeld op de volgende bladzijde. De lezer kan het best een alinea per keer lezen om een beeld te krijgen van de te verrichten handelingen.

### Noot betreffende de plaats waar de projectbestanden moeten worden opgeslagen

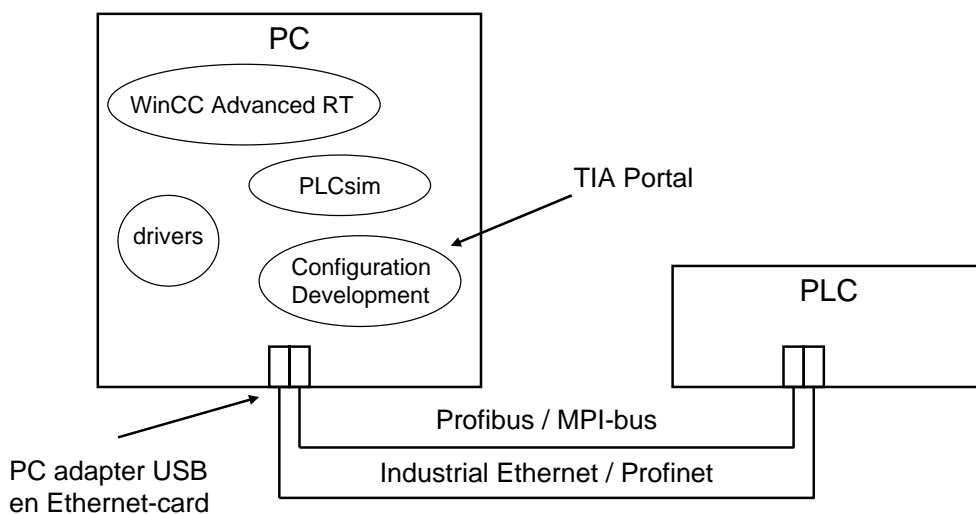
TIA Portal maakt vrij weinig bestanden aan in een project. De bestanden zijn verder ook niet heel erg groot. Het is daarom mogelijk om een project op een USB-stick te zetten. Zorg er wel voor dat de USB-stick snel is. Een langzame stick levert toch wat vertraging op. Het gebruik van de H:-schijf is niet mogelijk. Verder is het mogelijk om een project te archiveren. Archiveren wordt besproken in hoofdstuk 8.

### Boeken, datasheets en manuals

Siemens heeft een keur aan boeken, datasheets en manuals. Veel is te vinden op de support-website van Siemens [1]. Een goed boek dat veel beschrijft over de S7-300 en TIA Portal is van Berger [2].

## 2. PRACTICUMOMGEVING

De PC is een gewone IBM compatible computer en heeft in dit project meerdere functies: het is een ontwikkelstation om de PLC te configureren en programmeren, het is een ontwikkelstation voor de SCADA software en het is een monitoring station voor SCADA. Dit is in figuur 2.1 schematisch weergegeven.



Figuur 2.1: Engineering layout van de PC.

Aan de PC is een Ethernetkaart gekoppeld. Hiervoor zijn diverse drivers geïnstalleerd. De PC is met de PLC verbonden via de Ethernet-interface. De PC kan ook uitgerust worden met een PC Adapter USB van Siemens. Hiermee kan de PC met de PLC verbonden worden via de MPI-bus, een Siemens-eigen bus en protocol. Deze interface kan omgeschakeld worden naar Profibus, een gestandaardiseerd bussysteem dat gebruikt wordt in de automatiseringstechniek. Beide maken gebruik van het RS-485 protocol.

Op de PC is de volgende software geïnstalleerd:

### Windows 7 SP1

Eén van de Operating Systems van Microsoft, aangevuld met de laatste Service Packs.

### Simatic STEP 7 TIA Portal V14 Professional

Dit is de software van Siemens waarmee het PLC-programma ontwikkeld wordt en waarmee dat programma in de PLC geladen wordt. Dit laden gebeurt via de MPI-bus of Ethernet-bus. Deze versie heeft naast de bekende programmeertalen LAD, FBD en STL ook Graph en SCL.

### WinCC Advanced V14<sup>2</sup>

Deze software van Siemens wordt gebruikt voor procesvisualisatie (HMI = Human Machine Interface) en toezichhouderscontrole (SCADA = Supervisory Control And Data

<sup>2</sup> Bij de studentenlicentie wordt alleen WinCC Basic geïnstalleerd.

Aquisition). WinCC Advanced is een onderdeel van het totale pakket waarmee een *runtime* ontworpen wordt. Hier kan je dus knoppen, I/O-velden, etc. aanmaken en acties koppelen aan die knoppen en I/O-velden. Is een runtime ontworpen, dan wordt het gecompileerd en kan het draaien in de WinCC Advanced Runtime (RT) of op een HMI-paneel.

### **WinCC Advanced RT V14<sup>3</sup>**

WinCC Advanced RT is de component die de gecompileerde code uitvoert en daadwerkelijk interactie met de PLC vertoont. Het zal dus gegevens uit de PLC opvragen en bijvoorbeeld in een trenddatabase loggen. Zo kan de gebruiker informatie over langere tijd vergaren en opslaan om bijvoorbeeld historische trends te analyseren, zodat het productieproces verbeterd kan worden. De PC is te gebruiken als HMI-station.

### **PLCSIM V14**

Het is mogelijk om het PLC-programma te testen in een simulator. Dit droogzwemmen is handig als het programma niet op de plek van de machinerie wordt ontwikkeld en getest. PLCSIM bootst een PLC met invoer en uitvoer na. Na ontwikkeling van de programmatuur kan deze in de PLC-simulator geladen worden en getest worden. Let op: PLCSIM V14 ondersteunt alleen de S7-1200 en de S7-1500.

### **PLCSIM V5.4**

Deze versie van PLCSIM ondersteunt de S7-300 en de S7-400. Het is onderdeel van PLCSIM V14.

Alle tutorials kunnen met de studentenversie TIA Portal V14 SP1 en PLCSIM V5.4 uitgevoerd worden.

---

<sup>3</sup> Deze software zit niet bij de studentenlicentie.

## 3. DE PLC

Het aansturen van fabrieksmachines wordt gedaan met een Programmable Logic Controller, afgekort tot PLC. We maken gebruik van één uit de familie van de S7-300 serie van Siemens.

### 3.1 Beschrijving PLC

Het exemplaar dat tijdens deze tutorial wordt gebruikt is de CPU315F-2 PN/DP, gecombineerd met voeding, digitale en analoge I/O-module en een simulatiemodule. Een foto van de configuratie is te zien in figuur 3.1.



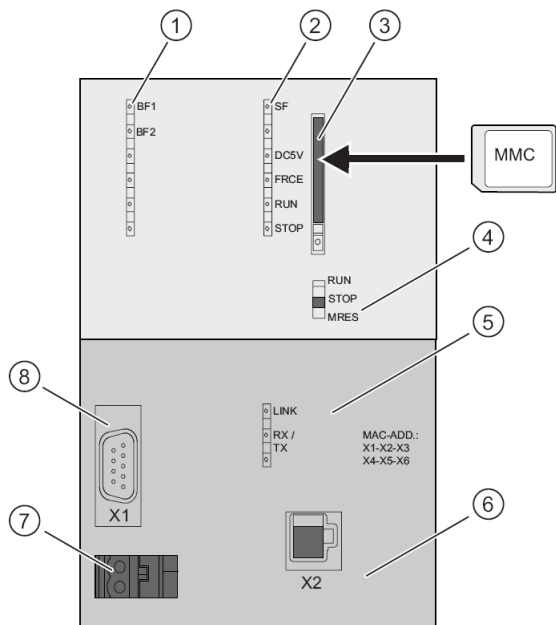
**Figuur 3.1:** De PLC S7-315 met I/O-modules.

Geheel links op de foto is de voedingsmodule te zien. Direct rechts daarvan, met gele sticker is de CPU-module te zien. Rechts van de CPU-module zijn achtereenvolgens de digitale I/O-module, de analoge I/O-module en de simulatiemodule te zien.

Meer informatie over de CPU-module kan gevonden worden in [3]. Meer informatie over de signaalmodules kan gevonden worden in [4].

### 3.1.1 Processormodule CPU315F-2 PN/DP

Dit is het hart van het systeem. Op de foto is deze module in het midden gesitueerd. Het bevat een CPU-eenheid en een FLASH-opslagkaart. De CPU kan worden geprogrammeerd door middel van TIA-Portal software. De module heeft naast een MPI/Profibus-aansluiting ook een Ethernet aansluiting voor Industrial Ethernet of Profinet.



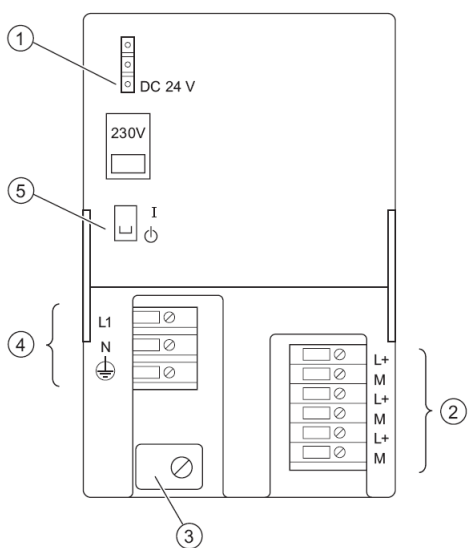
De volgende onderdelen zijn (figuur 3.2):

1. Bus Fail; statusleds betreffende de bussen.
2. Diverse statusleds, o.a. System Fail, Run en Stop mode.
3. Het MMC-slot. Hierin wordt de MMC-kaart geplaatst. Zonder deze kaart werkt de PLC niet.
4. Operation Switch. Zet de PLC in RUN of STOP.
5. Leds van de Ethernet Link.
6. De Ethernet/Profinet-aansluiting.
7. Voedingsspanning-aansluiting (+24V)
8. De MPI/Profibus-aansluiting

Figuur 3.2: De processormodule CPU315F.

### 3.1.2 Voedingsmodule PS307

Het geheel wordt gevoed met een aparte voeding, links op de foto. De voeding kan 5 A leveren en is krachtig genoeg om de CPU en de I/O-modules aan te sturen.



De volgende onderdelen zijn (figuur 3.3):

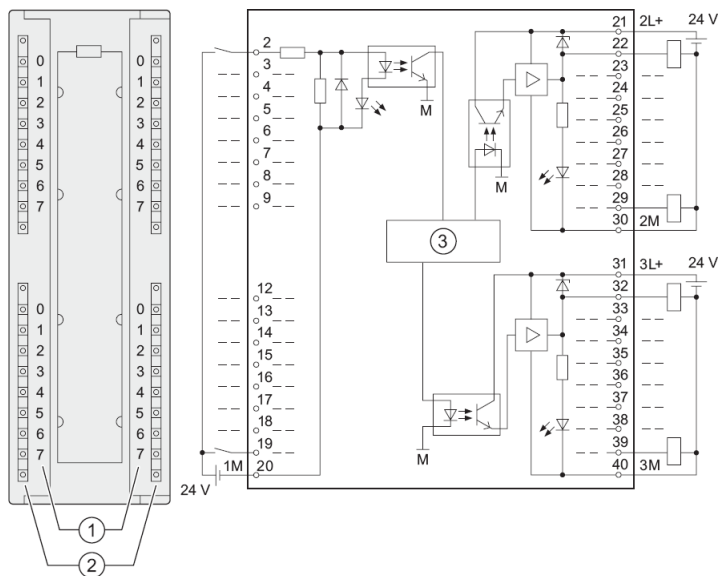
1. "24 Volt aanwezig"-lamp.
2. Aansluitingen voor 24 V DC.
3. Trekontlasting.
4. Spanningsaansluiting 230 V AC en aarde.
5. 24 V DC aan-uit schakelaar.

Figuur 3.3: De voedingsmodule PS307 5 A.



### 3.1.3 Digitale I/O module SM323

Deze module, rechts naast de CPU-module, heeft 16 digitale ingangen en uitgangen. De module werkt op 24 V gelijkspanning. Voor zowel de ingangen als de uitgangen geldt: een logische '0' komt overeen met 0 V, een logische '1' komt overeen met 24 V. De uitgangen kunnen maximaal 0,5 A leveren. Links op figuur 3.4 is de voorkant van de module afgebeeld. Links zijn de ingangen, rechts de uitgangen gesitueerd.



- De onderdelen zijn (figuur 3.4):
1. Kanaalnummer.
  2. Statusled (groen indien logisch '1').
  3. Backplane.

Figuur 3.4: Digitale I/O module SM323.

### 3.1.4 Analoge I/O module SM334

Deze module staat rechts naast de digitale I/O-module op de foto. Het heeft vier analoge ingangen en twee analoge uitgangen. De ingangen kunnen d.m.v. spanning of stroom worden aangestuurd. De uitgangen kunnen een spanning of een stroom leveren. Het spanningsbereik ligt nominaal tussen 0 en 10 V, het stroombereik ligt nominaal tussen 0 en 20 mA. Aangezien de PLC verder een digitaal systeem is, moeten de analoge waarden worden geconverteerd. Er wordt gebruik gemaakt van 8-bits resolutie voor zowel ingangen als uitgangen.

De digitale waarden bestaan allemaal uit 16 bits, ook al worden maar 8 bits effectief gebruikt. De analoge spanningen zijn allemaal positief, het tekenbit is dan ook 0. Daarna volgen de 8 bits. De overige bits zijn allemaal 0. Zie ook figuur 3.5.

Bitnummer	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Waarde	0	v	v	v	v	v	v	v	v	0	0	0	0	0	0	0

Figuur 3.5: Bitverdeling van de 8-bits analoge module SM334.

Voor zowel spanning als stroom geldt een "rated range", het bereik waarin normaal gewerkt wordt. Mocht een spanning of stroom daarboven komen, dan wordt over de "overshoot range" gesproken. Bij een nog hogere spanning of stroom wordt gesproken over "overflow". Zie ook tabel 3.1. Opmerking: deze waarden gelden voor alle analoge modu-



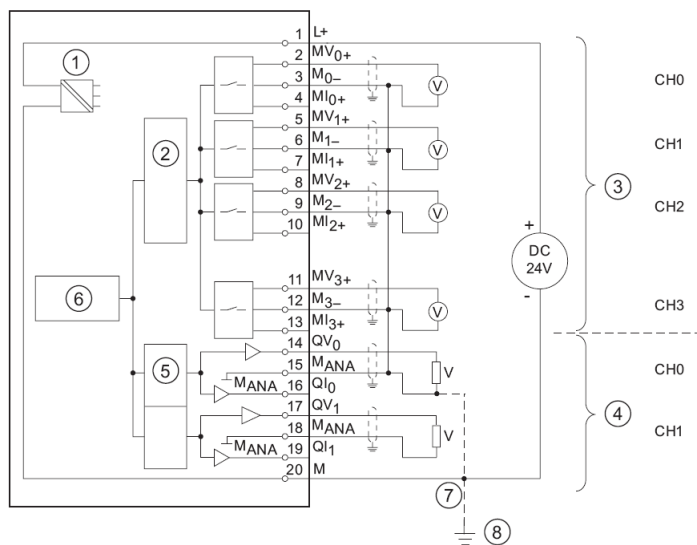
les.

Tabel 3.1: Enkele spanningen en stromen bij analoge modules.

Decimaal	Hexadecimaal	Spanning	Stroom	Opmerking
32767	7FFF	11,852 V	23,70 mA	Overflow
32512	7F00			
32511	7EFF	11,759 V	23,52 mA	Overshoot
27649	6C01			
27648	6C00	10,000 V	20,00 mA	Rated range
20736	5100	7,500 V	15,00 mA	
128	80	46,296 mV	92,59 uA	
0	0	0 V	0 mA	

Merk op dat de SM334 de digitale waarde in stappen van 128 kan aanpassen vanwege de 8-bits resolutie.

Indien een uitgang in de overflow range wordt gestuurd, levert de uitgang een spanning van 0 V of een stroom van 0 mA. Indien een ingang in de overflow range wordt gestuurd levert dit een waarde van 7FFF op.



De onderdelen zijn (figuur 3.6):

1. Interne voeding.
2. Analogue-digitaal converter.
3. Ingangen: spanningsmeting.
4. Uitgangen: spanningsaansluiting
5. Digitaal-analogue converter.
6. Backplane
7. Equipotential bonding.
8. Functionele aarding.

Figuur 3.6: Analoge I/O module SM334.

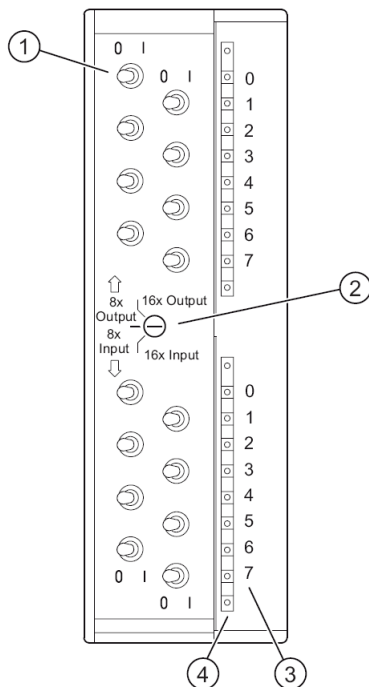
### 3.1.5 Simulatie I/O module SM374

Geheel rechts op de foto is een I/O-module te zien, waarmee digitale ingangen en uitgangen gesimuleerd kunnen worden. Nu is simuleren niet helemaal het juiste woord, het zijn wel gewoon in- en uitgangen. Alleen zijn de ingangen te bedienen door een gebruiker die hiermee de werking van een machine kan simuleren. De module is te gebruiken als:

- 16x ingang via schakelaars,
- 16x uitgang via leds,

- 8x ingang en 8x uitgang gecombineerd.

In deze laatste modus wordt de module gebruikt tijdens de tutorials.



De onderdelen zijn (figuur 3.7):

1. Schakelaars.
2. Modus-selector.
3. Kanaalnummer.
4. Leds

Tijdens de tutorials wordt de 8x input / 8x output modus gebruikt. De selector moet daarvoor in het midden staan. Het bovenste gedeelte werkt als 8x output; de schakelaars hebben geen functie. Het onderste gedeelte werkt als 8x input; de leds geven de stand van de schakelaars weer.

Figuur 3.7: Simulatiemodule SM374.

## 3.2 Programmeertalen

De PLC is ontstaan in de energie- en automatiseringstechniek en niet in de “echte” computerhoek. De programmeurs hadden geen opleiding tot computerprogrammeurs gevolgd. Hierdoor ontbrak in het begin een echte programmeertaal. In de loop der jaren zijn diverse methoden van programmeren ontworpen. Noot: hieronder volgt slechts een korte beschrijving.

De internationale norm IEC 61131-3<sup>4</sup> beschrijft een aantal programmeertalen. Hieronder een overzicht. In de opsomming wordt eerst de IEC-naam gegeven en tussen de haakjes de IEC-afkorting en de afkorting die Siemens gebruikt.

### Ladder Diagram (LD, LAD)

Deze taal bestaat uit een set van symbolische instructies die op een grafische wijze worden gepresenteerd. Er zijn vijf categorieën van instructies: relais<sup>5</sup>, timer/counter, rekenkundig, data-manipulatie en programmabesturing. Deze symbolen kunnen zo worden opgesteld dat de gewenste werking in het geheugen wordt ingevoerd. Ladderdiagrammen worden gebruikt voor het besturen van (contact-)uitgangen gebaseerd op ingangscondities.

### Function Block Diagram (FBD, FBD)

<sup>4</sup> Zie <http://www.plcopen.org>

<sup>5</sup> Vroeger waren de uitgangen van relais voorzien. Tegenwoordig worden de uitgangen ook met transistoren of thyristoren uitgevoerd. Men heeft echter de oude terminologie aangehouden.

Dit is een grafische taal waarbij de logische functies als blokken worden weergegeven. Naast de bekende AND en OR zijn er ook blokken voor tellers, timers enz. Het is een alternatief voor ladderdiagrammen.

### Instruction List (IL, STL)

Dit is het best te omschrijven als een assemblertaal voor PLC's. Diverse instructies zoals L (load) en T (output) zijn voor handen. Hierin is het mogelijk om zeer nauwkeurig de werking van een programma te beschrijven omdat de instructies één op één worden vertaald naar machinetaal. STL staat voor Statement List. STL is niet conform de IEC-norm.

### Structured Text (ST, SCL)

Deze taal lijkt erg op Pascal. Er kan geprogrammeerd worden als een echte programmeertaal, compleet met IF's, functieaanroepen, verschillende datatypes etc. SCL staat voor Structured Control Language. SCL is conform de IEC-norm en heeft enkele uitbreidingen.

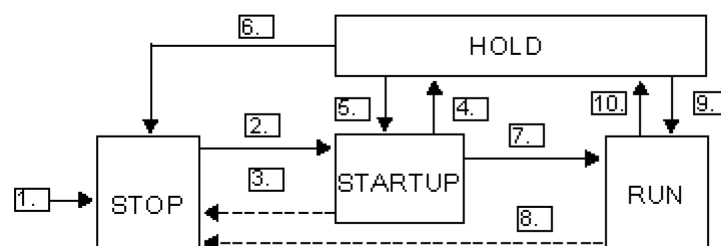
### Sequential Function Chart (SFC, Graph)

Ook dit is weer een grafische taal. Het is te vergelijken met het beschrijven van een toestandsmachine uit de digitale techniek; de condities van de overgangen worden beschreven d.m.v. LD's of FBD's. Aftakkingen (Branch) en parallelisme (Simultaneous Branch) zijn mogelijk. Siemens heeft een variant, S7-Graph genaamd, die voldoet aan de IEC 61131-3 norm.

Siemens levert een softwareapplicatie, genaamd TIA Portal Professional, waarin al deze programmeertalen zijn opgenomen.

## 3.3 Gebruik PLC

De PLC is een robuust apparaat. Dat neemt niet weg dat er voorzichtig met een PLC moet worden omgegaan. De PLC kan tijdens de tutorials en werkzaamheden gewoon onder spanning blijven staan. Wel is het raadzaam om de PLC in STOP te zetten als er wijzigingen in het geheugen worden gedaan. Hierbij valt te denken aan configuratie-aanpassing en nieuwe blokken verzenden. Hieronder staan de operating modes van de PLC afgebeeld. Zie figuur 3.8.



Figuur 3.8: De operating modes van de PLC.

Uitleg van de overgangen:

1. Na aanzetten komt de CPU in STOP.
2. De CPU gaat naar STARTUP als de schakelaar in RUN wordt gezet.

## CONCEPT

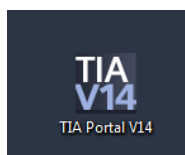
3. De CPU komt in STOP als de schakelaar in STOP wordt gezet of de PLC een fout detecteert.
4. Hold mode (wordt niet besproken).
5. Hold mode (wordt niet besproken).
6. Hold mode (wordt niet besproken).
7. De CPU komt in RUN als alles goed is gegaan.
8. De CPU komt in STOP als de schakelaar in STOP wordt gezet of de PLC een fout detecteert.
9. Hold mode (wordt niet besproken).
10. Hold mode (wordt niet besproken).

## 4. TUTORIAL LAD

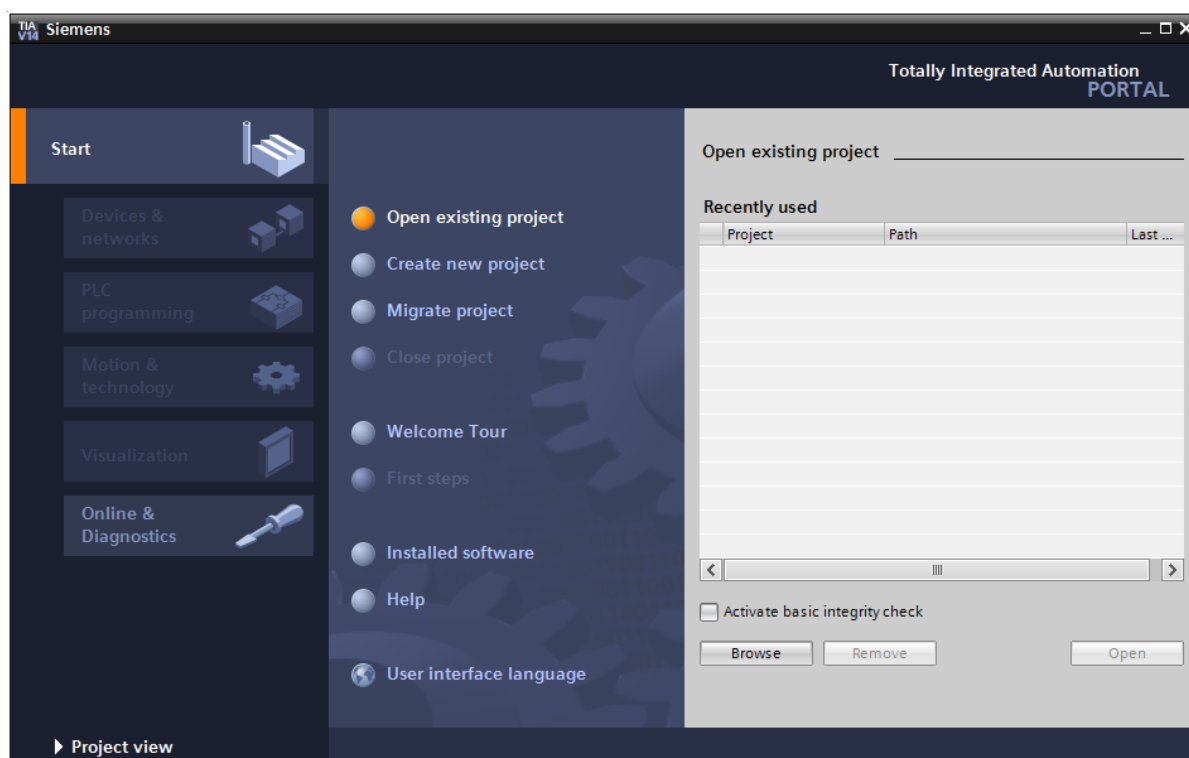
In deze tutorial wordt uitgelegd hoe een project moet worden aangemaakt. Er wordt een eenvoudig verkeerslichtsysteem ingevoerd. Eerst wordt een nieuw project aangemaakt, dan worden de netwerk- en hardwareconfiguratie van de PLC opgesteld. Daarna wordt de PLC tags-lijst aangemaakt zodat het programmeren wat eenvoudiger wordt. Vervolgens wordt het programma ingevoerd en als laatste wordt het programma online gevolgd (monitoren).

TIA Portal is een groot programma met zeer veel mogelijkheden. In deze tutorial wordt slechts een deel van de mogelijkheden getoond.

Start STEP7 Tia Portal V14 door op het pictogram te klikken, zie figuur 4.1. Na enige tijd volgt het openingsscherm, zie figuur 4.2.



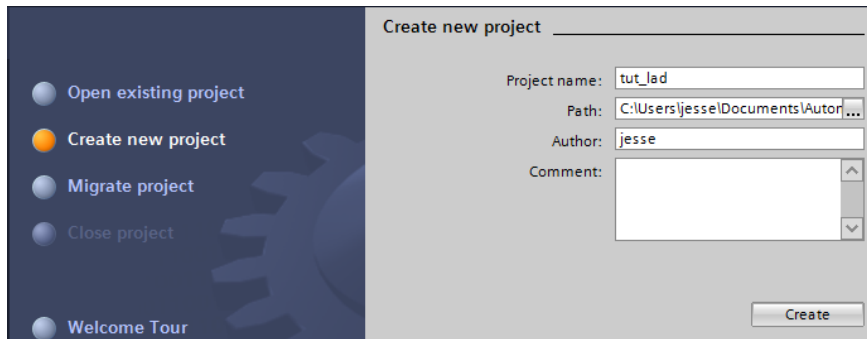
Figuur 4.1: Pictogram Tia Portal V14.



Figuur 4.2: Openingsscherm Tia Portal V14.

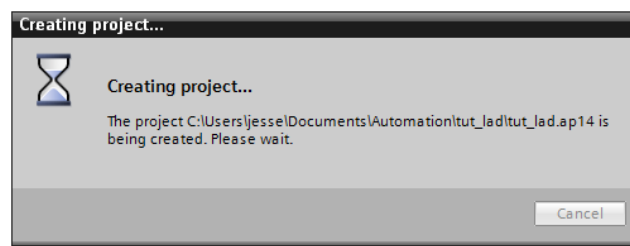
## 4.1 Aanmaken nieuw project

Selecteer in het midden van het TIA Portal-scherf Create new project. Vul als projectnaam tut\_lad in. Selecteer onder Path een geschikte map om het project onder te brengen, eventueel op een USB-stick. Klik daarna op **Create**. Zie figuur 4.3.



Figuur 4.3: Aanmaken nieuw project.

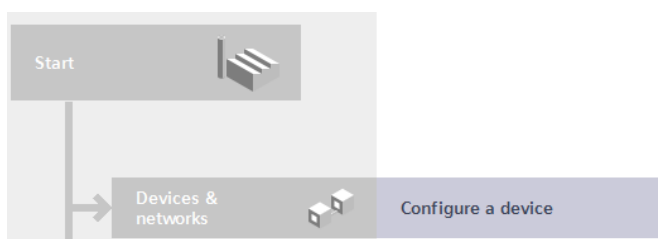
TIA Portal gaat nu een nieuw project opzetten. Dat kan enige tijd duren. Tijdens het creëer-proces wordt een scherm getoond met de voortgang. Zie figuur 4.4.



Figuur 4.4: Nieuw project wordt aangemaakt.

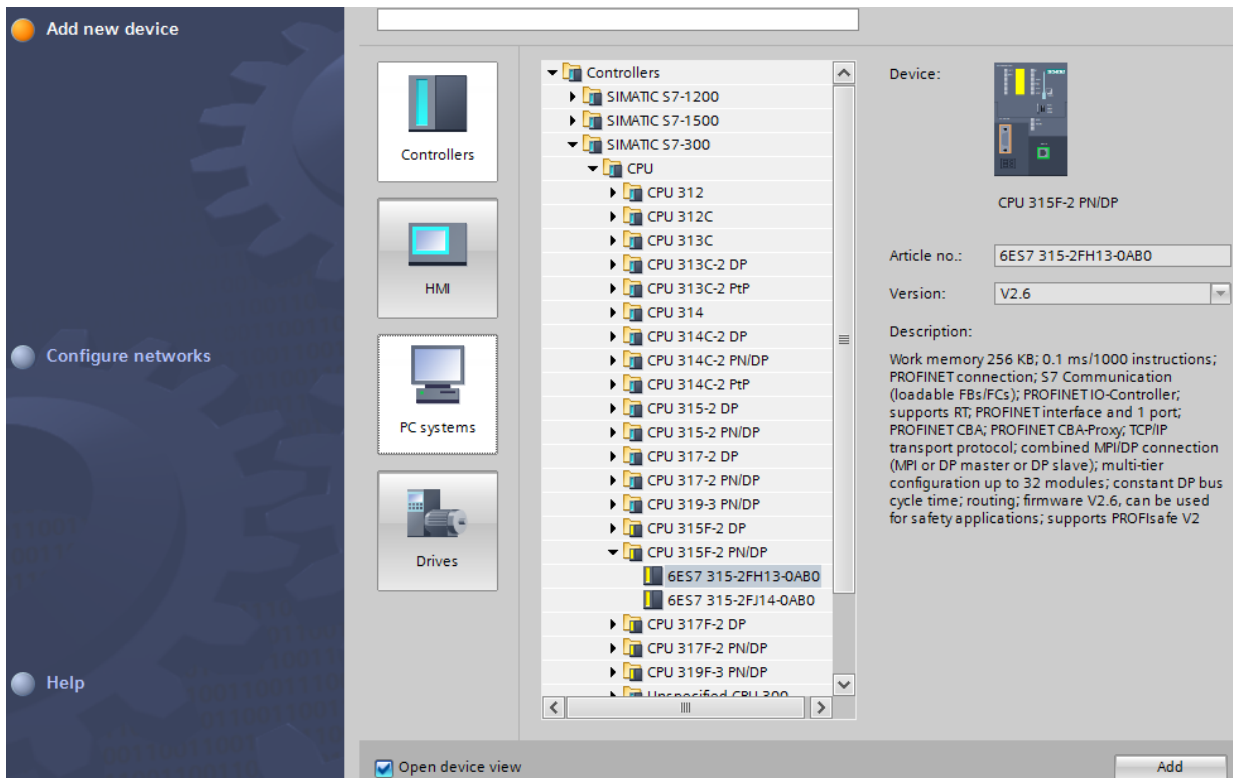
## 4.2 PLC configureren

Nadat het project is aangemaakt verschijnt een scherm (figuur 4.5). Klik hierin op **Configure a device**.

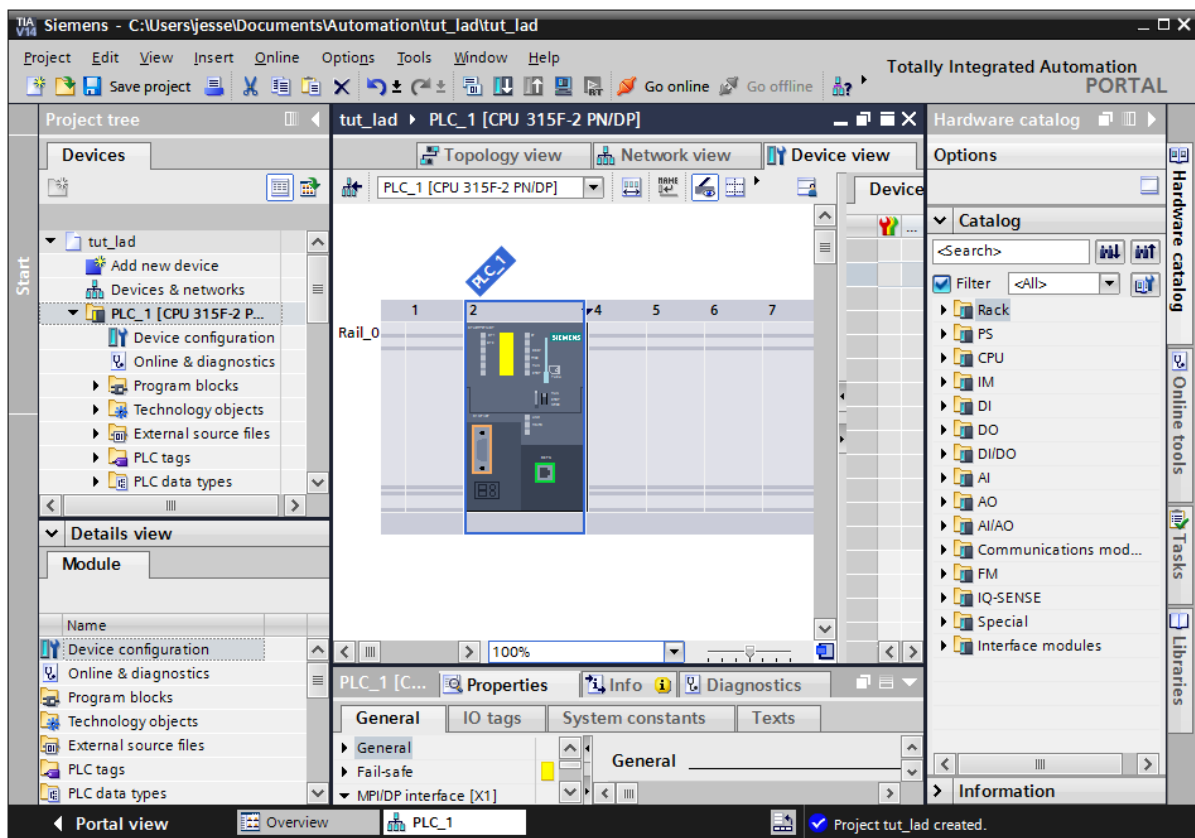


Figuur 4.5: Nieuw apparaat configureren.

Er wordt een scherm geopend (figuur 4.6) Klik links op **Add new device** en daarna op de pictogram **Controllers**. Selecteer in de lijst van PLC's het type CPU 315F-2 PN/DP met ordernummer 6ES7 315-2FH13-0AB0. Dit kost enige tijd. Klik daarna onderaan op de knop **Add**. De schermindeling verandert nu radicaal. Dit is te zien in figuur 4.7.



Figuur 4.6: Selecteren CPU-module.



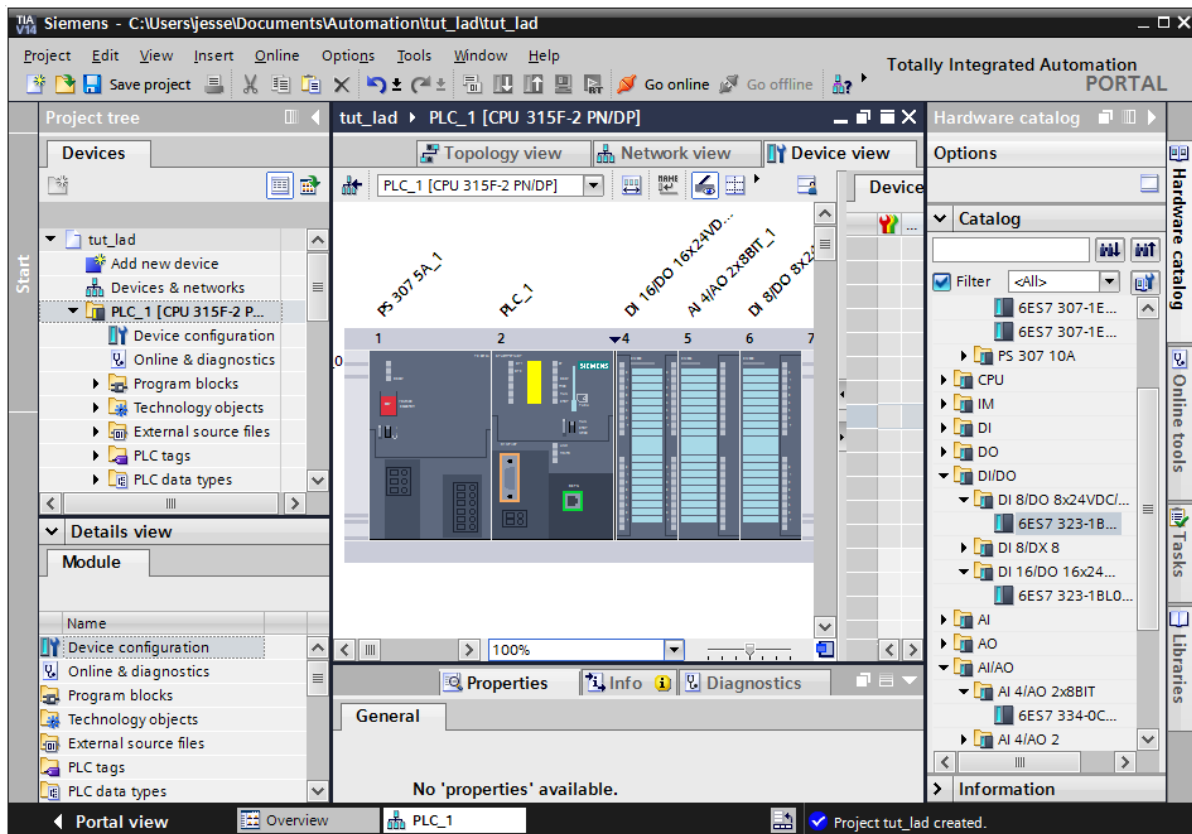
Figuur 4.7: Project view van de PLC-configuratie.

Figuur 4.7 toont de *project view* van alles wat zich in het project bevindt. In het linker paneel, de zogenoemde *project tree*, worden de apparaten en alles wat daarmee samenhangt weergegeven. In het midden is de PLC-configuratie op schematische wijze weergegeven. De CPU-module is geplaatst in slot 2. Nu moeten de voeding, de digitale I/O, analoge I/O en de simulatiemodule worden toegevoegd. Deze zijn te kiezen via het rechter paneel.

Selecteer via de catalogus in het rechter paneel de voedingsmodule via PS→PS 307 5A→6ES7 307-1EA00-0AA0 en plaats deze in slot 1. Let hierbij op de Siemens ordernummers. Plaats vervolgens de drie overige modules. De ordernummers zijn:

- Het ordernummer voor de voedingsmodule is 6ES7 307-1EA00-0AA0 (slot 1)
- Het ordernummer voor de digitale I/O-module is 6ES7 323-1BL00-0AA0 (slot 4).
- Het ordernummer voor de analoge I/O-module is 6ES7 334-0CE01-0AA0 (slot 5).
- Het ordernummer voor de simulatiemodule is 6ES7 323-1BH01-0AA0 (slot 6).

De volledige moduleconfiguratie is te zien in figuur 4.8.

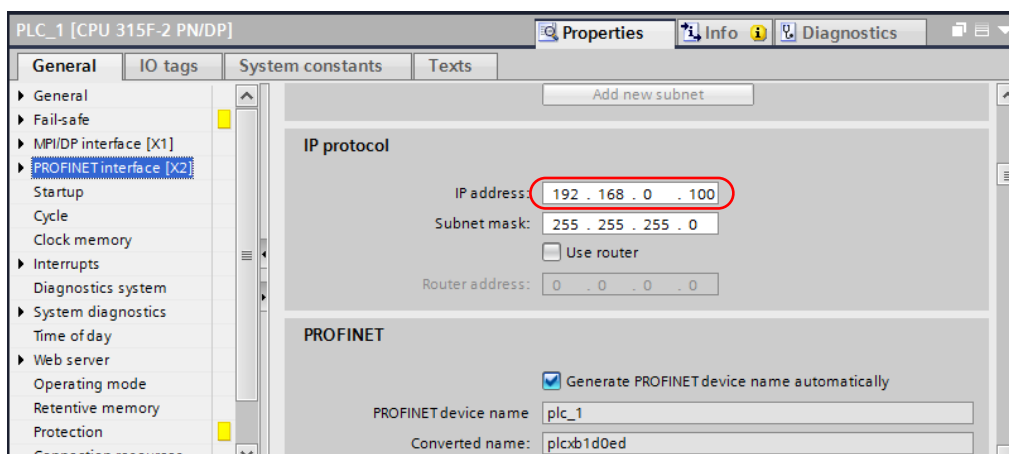


**Figuur 4.8:** Volledige module-configuratie van de PLC.

Nu moeten het IP-adres en het MPI-adres van de PLC geconfigureerd worden. Klik in de PLC-configuratie in het paneel in het midden op het groene vierkantje. Dit is het symbool voor de Profinet-interface. In het paneel midden onderaan verschijnen nu de eigenschappen van de CPU-module. Kies in het paneel onderaan de optie PROFINET interface [X2]. Navigeer in het rechter deel van het paneel naar het veld IP protocol. Vul hier het IP-adres in. Zie figuur 4.9.



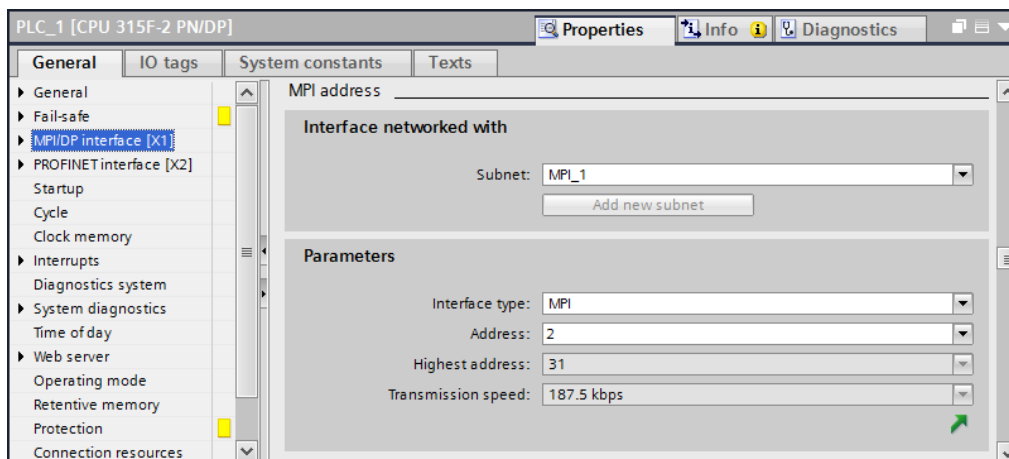
**Let op: voor elke PLC moet een ander IP-adres worden ingevuld.**



**Figuur 4.9:** Instellen IP-adres.

Afhankelijk van de gekozen PLC moet een IP-adres in het bereik van 192.168.0.100 t/m 192.168.0.107 worden ingevuld.

Het MPI-adres kan op vergelijkbare wijze worden ingesteld. Klik in de PLC-configuratie (figuur 4.8) op de oranje rechthoek. Dit is het symbool voor de MPI-interface. Selecteer in het paneel midden onderin op MPI/DP interface [X1]. Navigeer in het rechter deel van het paneel naar het veld Parameters. Vul in het veld Address de waarde 2 in. Zie figuur 4.10.

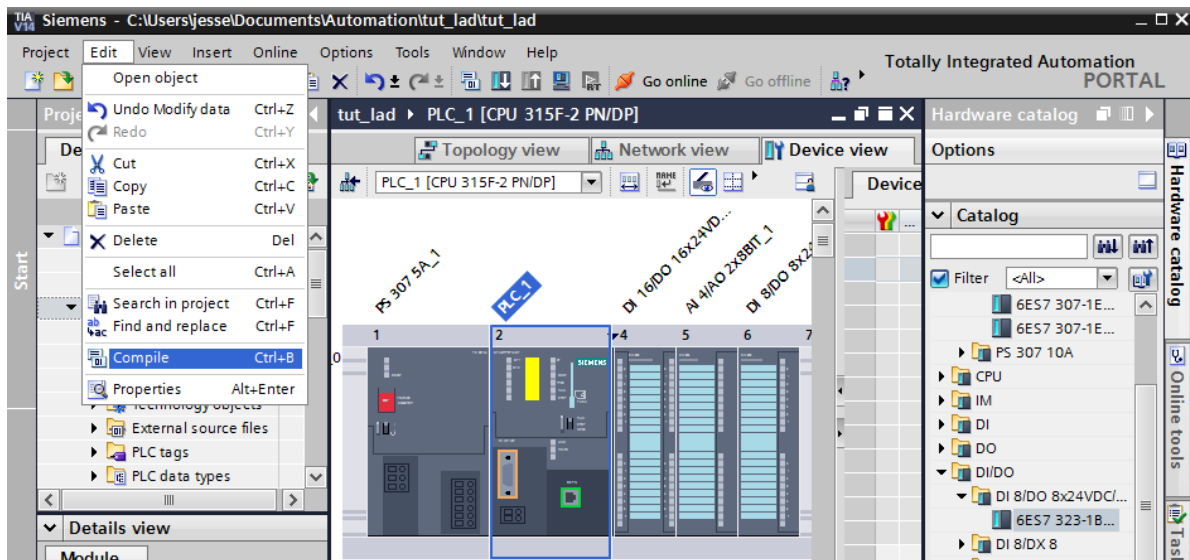


**Figuur 4.10:** Instellen MPI-adres.

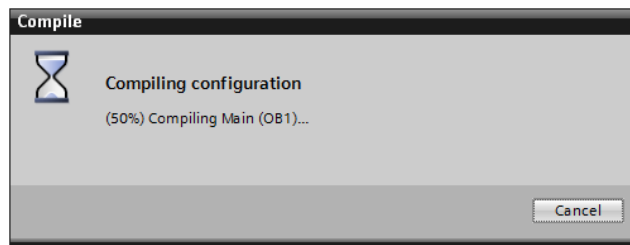
De configuratie is nu klaar en kan in de PLC worden geladen. Klik op de PLC-configuratie en selecteer via het menu **Edit**→**Compile**. Zie figuur 4.11. TIA Portal gaat nu de configuratie compileren. Zie figuur 4.12. Dit kan enige tijd duren.

### 4.3 PLC-configuratie downloaden

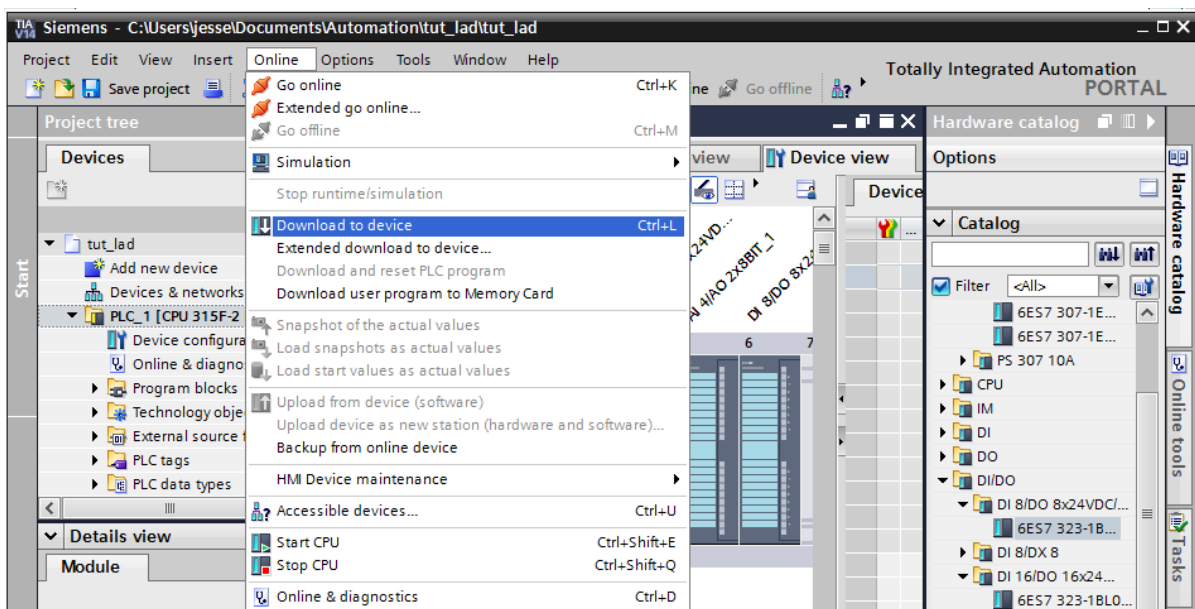
Zet de PLC nu eerst in STOP door de operating mode schakelaar van RUN naar STOP te zetten. Nu kan de configuratie naar de PLC worden gestuurd via het menu **Online**→**Download to device**. Zie figuur 4.13.



Figuur 4.11: Compilieren PLC-configuratie.



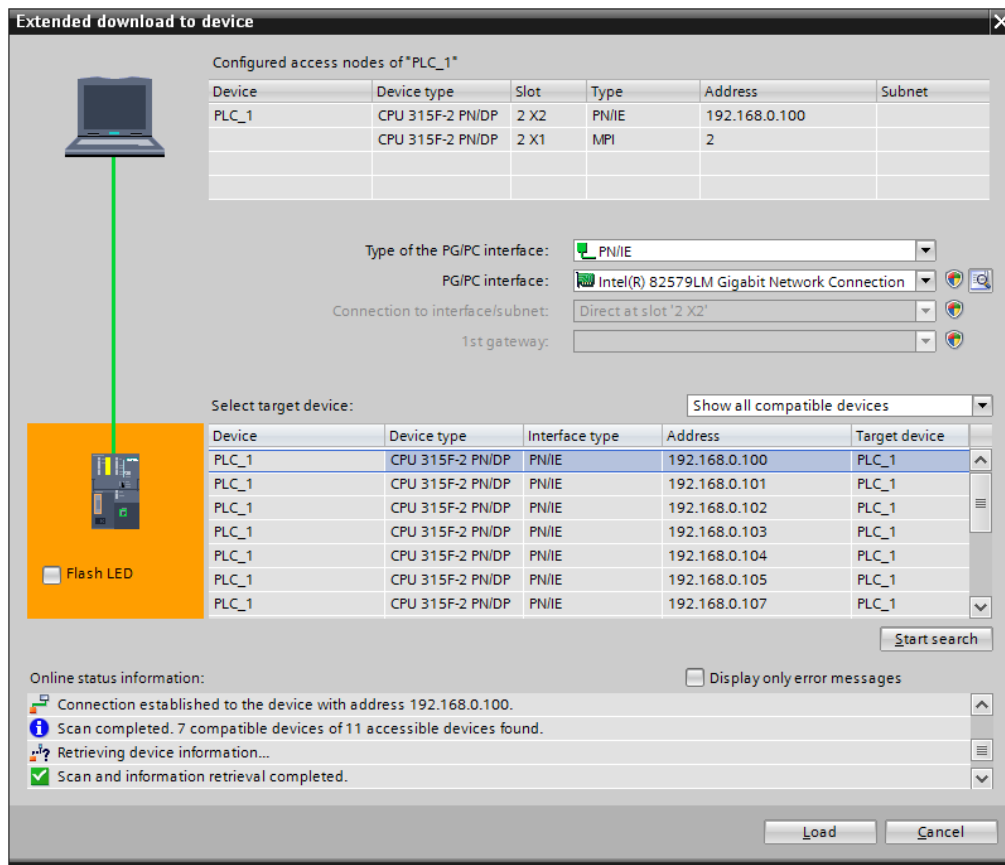
Figuur 4.12: PLC-configuratie wordt gecompileerd.



Figuur 4.13: Selecteren download naar PLC.

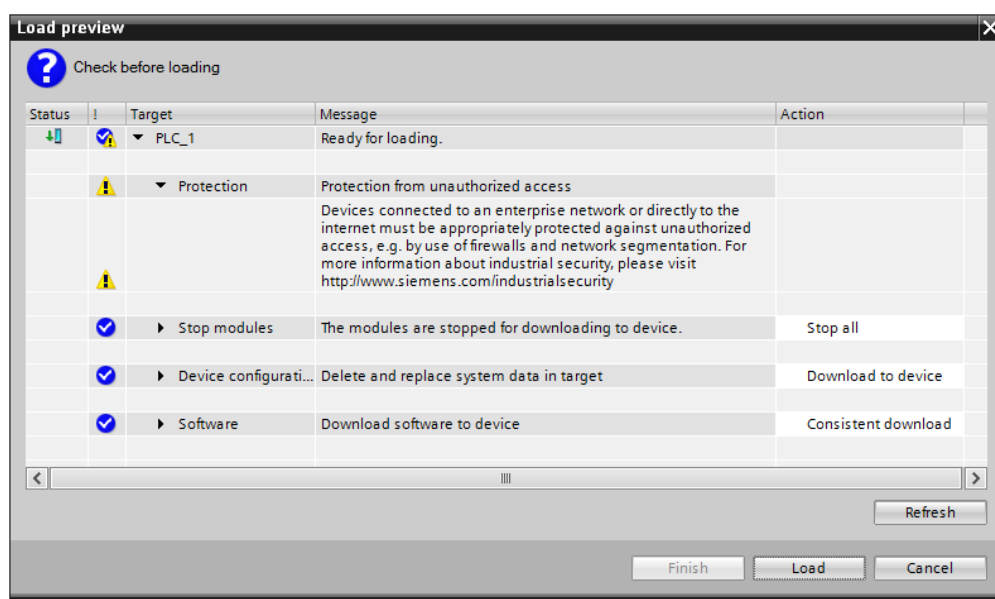
TIA Portal laat nu een scherm zien waarin de PLC geselecteerd kan worden. Zie figuur 4.14. Via de knop **Start search** wordt een lijst samengesteld met daarin alle adresseerbare PLC's. Selecteer de PLC met het juiste IP-adres. In dit voorbeeld is dat de PLC

met IP-adres 192.168.0.100 en klik op de knop **Load**. Raadpleeg de docent als het juiste IP-adres niet te zien in.



Figuur 4.14: Selecteren PLC voor download.

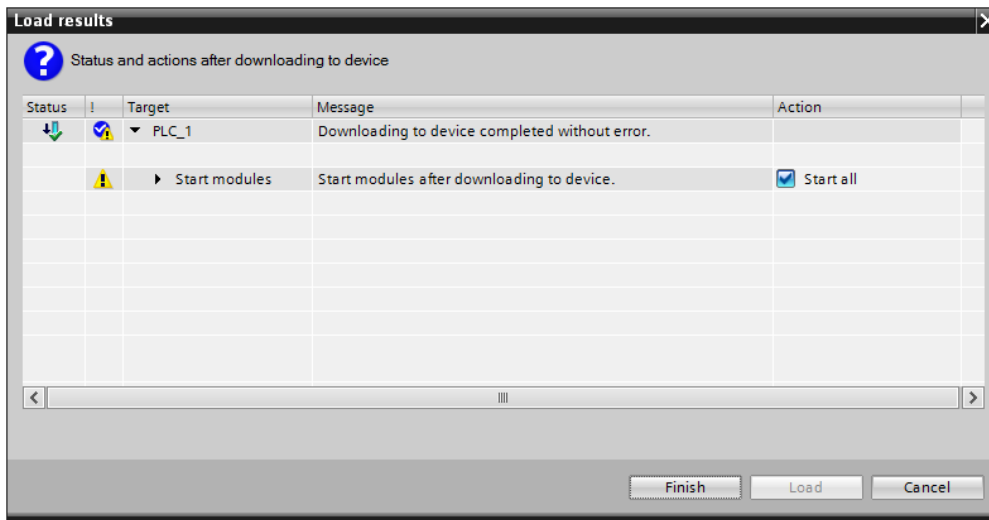
Daarna wordt een scherm zichtbaar waarin de acties vermeld staan die worden uitgevoerd. Zie figuur 4.15. Klik nogmaals op de knop **Load** en de configuratie wordt naar de



Figuur 4.15: Preview van de acties tijdens download naar PLC.

PLC gestuurd (eigenlijk naar de CPU-module).

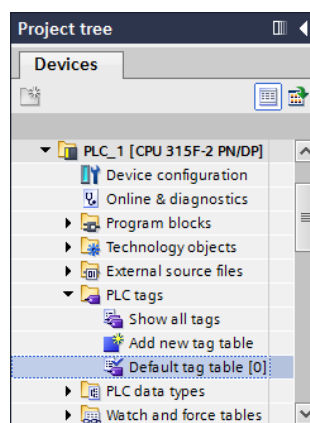
Als de PLC niet in STOP is gezet voordat de configuratie geladen werd, verschijnt figuur 4.16. Deze figuur geeft het resultaat weer van de download naar de PLC. Er wordt een mogelijkheid gegeven om de PLC te starten, zie de checkbox **Start all**. Klik op de knop **Finish** om de PLC te starten.



Figuur 4.16: Resultaat van de download naar de PLC en de mogelijkheid om de PLC te starten.

## 4.4 PLC tags invoeren

Het onthouden van de adressen van ingangen, uitgangen en timers, is een lastige en vermoeiende bezigheid. Gelukkig kan een symbolenlijst, in TIA Portal PLC tags genoemd, worden opgesteld. Er kan dan gerefereerd worden met de symbolische naam in plaats van het eigenlijke adres. Open in het linker paneel Devices het onderdeel PLC tags. Dubbelklik daarna op het onderdeel Default tag table. Zie figuur 4.17.



Figuur 4.17: Starten invoer PLC tags.

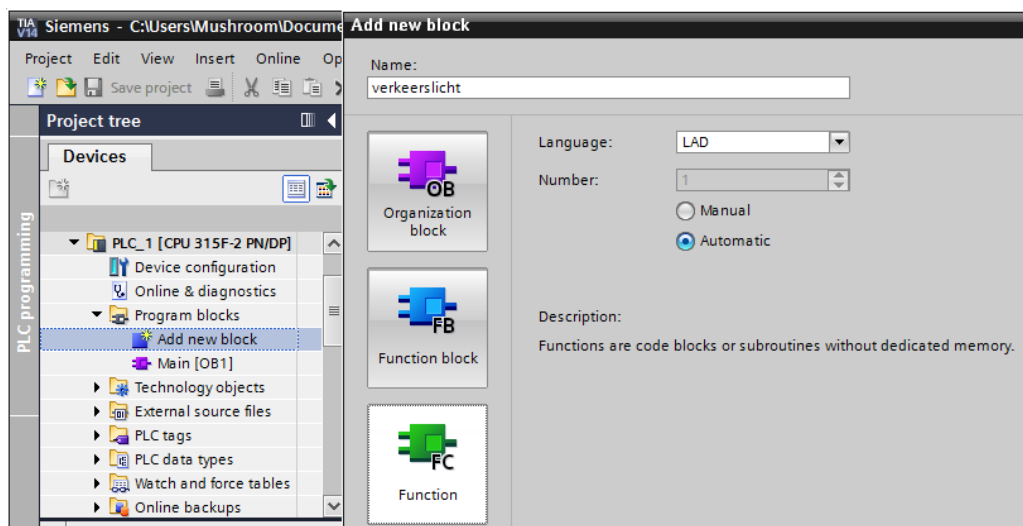
De PLC tags-editor wordt dan gestart. Voer de tags in zoals is weergegeven in figuur 4.18 (het veld Data Type wordt door de editor zelf ingevuld). Sluit de editor af, de definities worden automatisch opgeslagen.

	Name	Data type	Address	Retain	Acces...	Visibl...	Comment
1	drukknop1	Bool	%I8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Drukknop 1
2	drukknop2	Bool	%I8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Drukknop 2
3	reset	Bool	%I8.7		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Reset systeem
4	voet_gezien	Bool	%M8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Voetganger gezien
5	timer_geel_klaar	Bool	%M8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Timer voor geel klaar met tijdmeter
6	timer_groen_klaar	Bool	%M8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Timer voor groen klaar met tijdmeter
7	voet_gezien_flank	Bool	%M8.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Flankdetector voet_gezien
8	voet_lampje	Bool	%Q8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Voetganger gezien feedback lamp
9	auto_rood	Bool	%Q8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht auto rood
10	auto_geel	Bool	%Q8.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht auto geel
11	auto_groen	Bool	%Q8.4		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht auto groen
12	voet_rood	Bool	%Q8.6		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht voetganger rood
13	voet_groen	Bool	%Q8.7		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht voetganger groen
14	timer_geel	Timer	%T0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Timer voor auto geel
15	timer_groen	Timer	%T1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Timer voor auto groen
16	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figuur 4.18: Alle PLC tags ingevoerd.

## 4.5 Programma invoeren

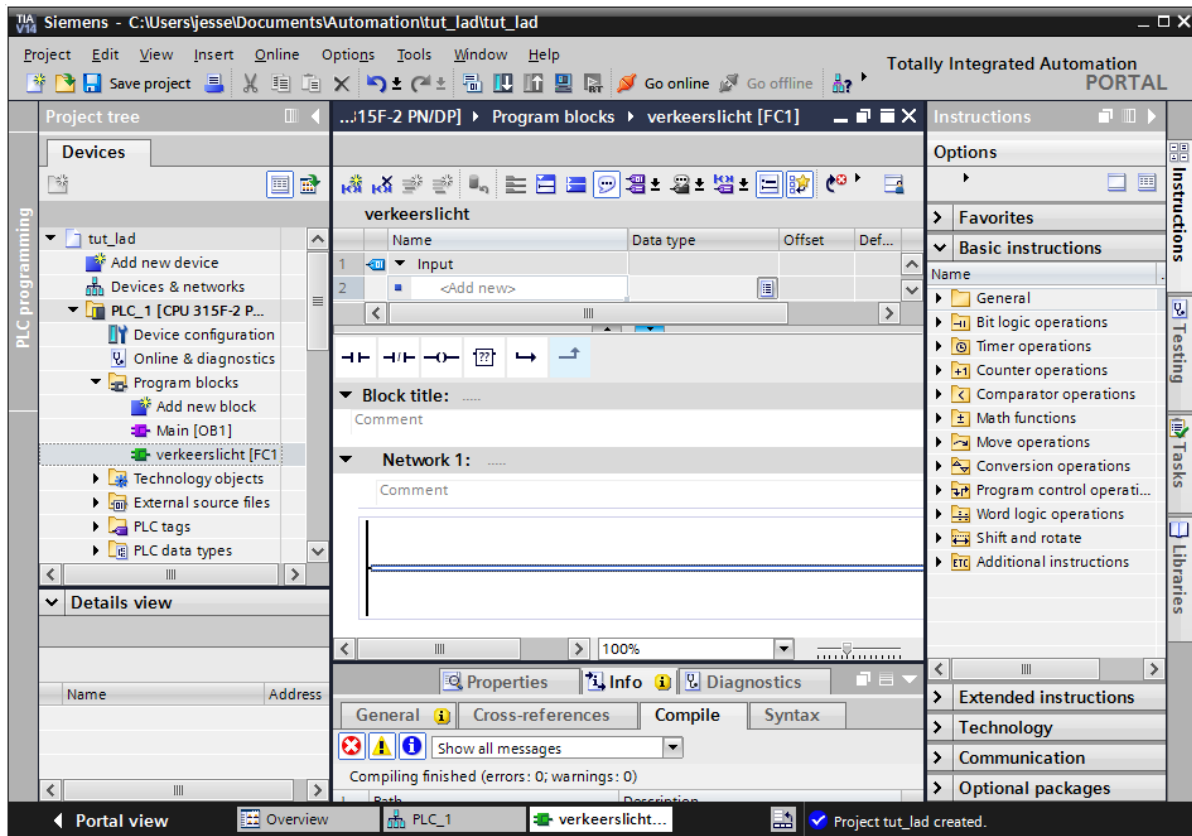
Zoals bekend moet een PLC-programma altijd Organization Block 1 (OB1) bevatten, ook al roept deze niets anders aan dan Functions (FC) en Function Blocks (FB). Eerst wordt FC1 geprogrammeerd met het eigenlijke verkeerslichtprogramma, vervolgens wordt OB1 geprogrammeerd met daarin een aanroep van FC1. Deze volgorde is van belang. Vanuit TIA Portal moet eerst FC1 aangemaakt worden. Selecteer in het linker paneel onder Program blocks de optie Add new block. Er wordt een scherm geopend waarmee een programmablok kan worden toegevoegd. Klik op de pictogram Function en vul bij Name het woord verkeerslicht in. Zet de programmeertaal op LAD. Zie figuur 4.19. Klik op de knop **Add** (niet in de figuur weergegeven).



Figuur 4.19: Aanmaken functie.

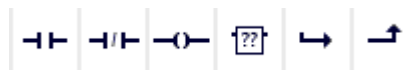
TIA Portal zal nu functie FC1 aanmaken en openen. Dit is te zien in figuur 4.20.

Een ladderdiagram bestaat uit één of meerdere netwerken (ook wel een *rung* genoemd).



Figuur 4.20: Functie FC1 aangemaakt en geopend.

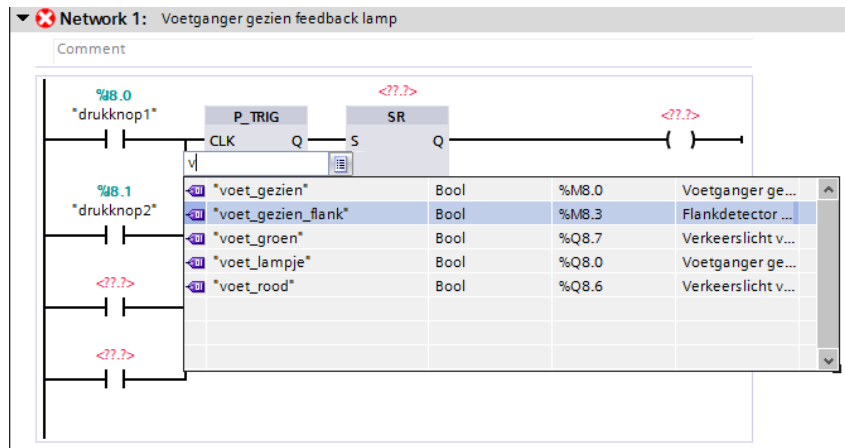
Per netwerk kan één logische schakeling geprogrammeerd worden met één of meerdere uitgangen. Als ingangen kunnen PLC-ingangen en merkers dienen. Zie voor de pictogrammen figuur 4.21. De eerste en tweede pictogrammen zijn respectievelijk *normally open* en *normally closed* contacten. Als uitgangen (*coils*) kunnen PLC-uitgangen of merkers gebruikt worden. Dit is te zien in het derde pictogram. Een logische OR wordt gemaakt met behulp van de laatste twee pictogrammen.



Figuur 4.21: Meest voorkomende componenten.

Het programmeren vergt enige routine. Om ingangen in te voegen moet b.v. eerst een rung van een netwerk worden geselecteerd. Dan worden de pictogrammen uit figuur 4.21 geactiveerd en kunnen ze worden ingevoegd. Niet alle mogelijkheden kunnen via de pictogrammen worden ingevoerd. Aan de linkerkant staan alle mogelijkheden, zoals SR-elementen, timers, counters en rekenkundige operaties. Tijdens het invoeren van symbolische namen als *voet\_gezien* verschijnt de PLC tags-lijst. Dan kan eenvoudig de juiste tag worden gekozen. Zie figuur 4.22.

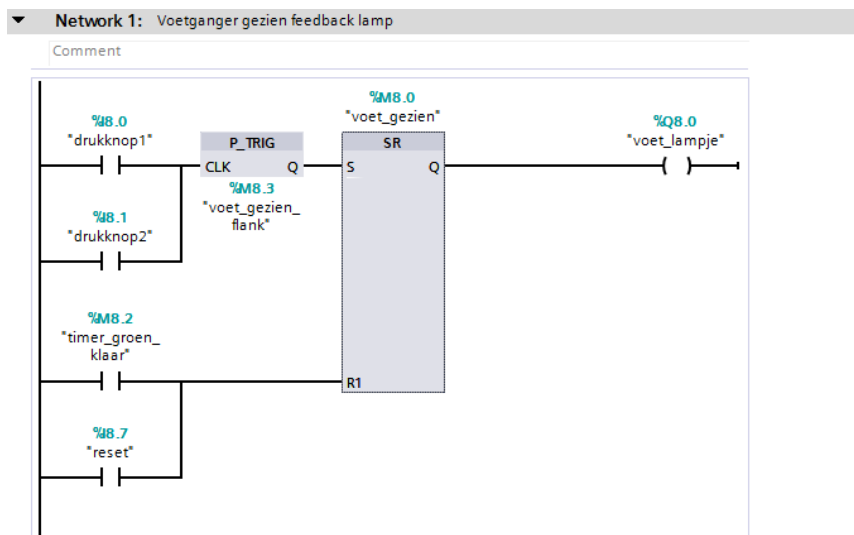
Indien iets niet ingevoerd kan worden, zoals een BYTE-adres bij een contact, wordt dit rood afgebeeld en kan de laddereditor niet worden afgesloten. Ook als adressen niet zijn ingevuld, zoals in figuur 4.22 zichtbaar is, kan de editor niet worden afgesloten.



Figuur 4.22: Invoeren PLC tags.



Als voorbeeld is het eerste netwerk in figuur 4.23 afgebeeld. Het complete programma is opgenomen in bijlage A. Voor correcte werking moet het hele programma ingevoerd worden.



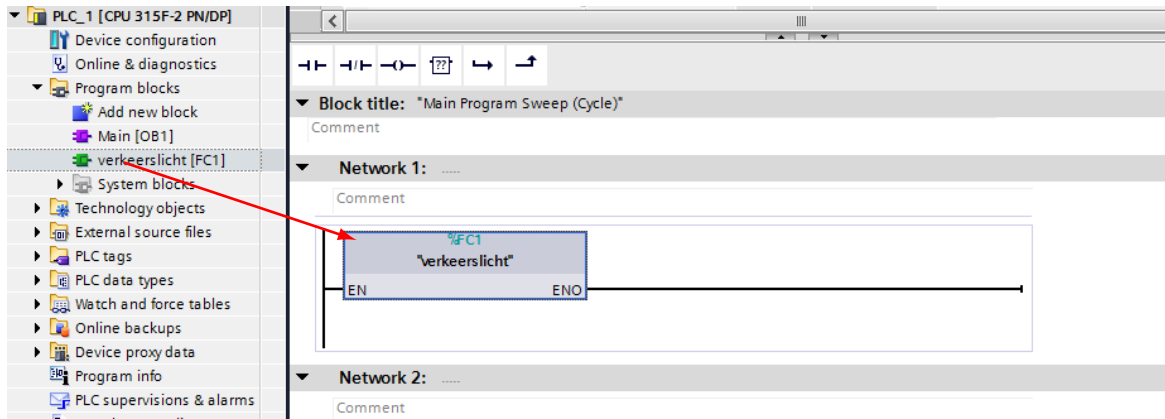
Figuur 4.23: Het eerste netwerk volledig.

Nu moet in OB1 nog een aanroep naar FC1 worden geprogrammeerd. Dat kan vrij eenvoudig. Dubbelklik in het linker paneel op Main (OB1). Sleep vervolgens FC1 naar het eerste netwerk in de editor. FC1 zal daar geplaatst worden. Zie figuur 4.24.

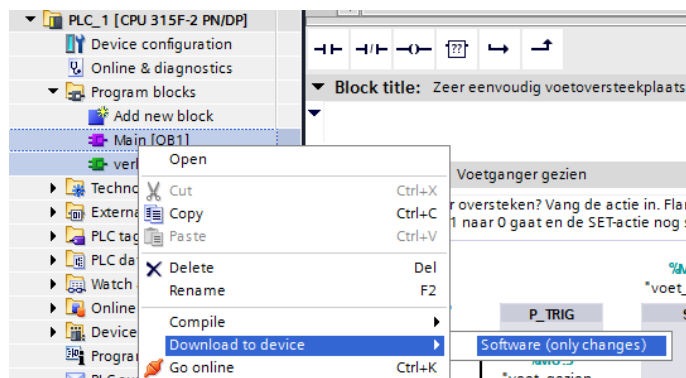
## 4.6 Laden PLC met programma

Nu moeten de diverse onderdelen in de PLC geladen worden. Zet de PLC eerst in STOP. Selecteer in het linker paneel de blokken OB1 en FC1. Klik op de rechter muisknop en selecteer de menu-optie **Download to device** → **Software (only changes)**. Zie figuur 4.25.

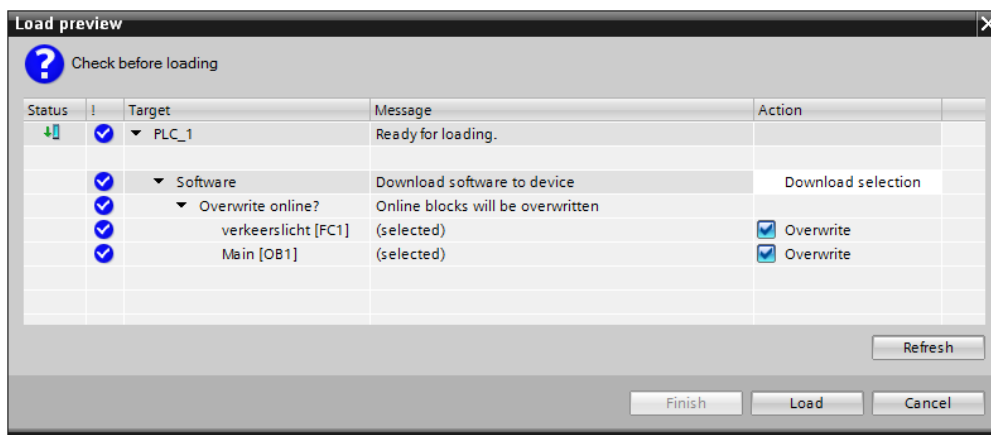
Er wordt een dialoog geopend, te zien in figuur 4.26. Klik op de knop **Load**. Het PLC-programma kan nu getest worden.



Figuur 4.24: OB1 roept FC1 aan.



Figuur 4.25: Selecteer OB1 en FC1 voor download.



Figuur 4.26: Download-dialogoog.

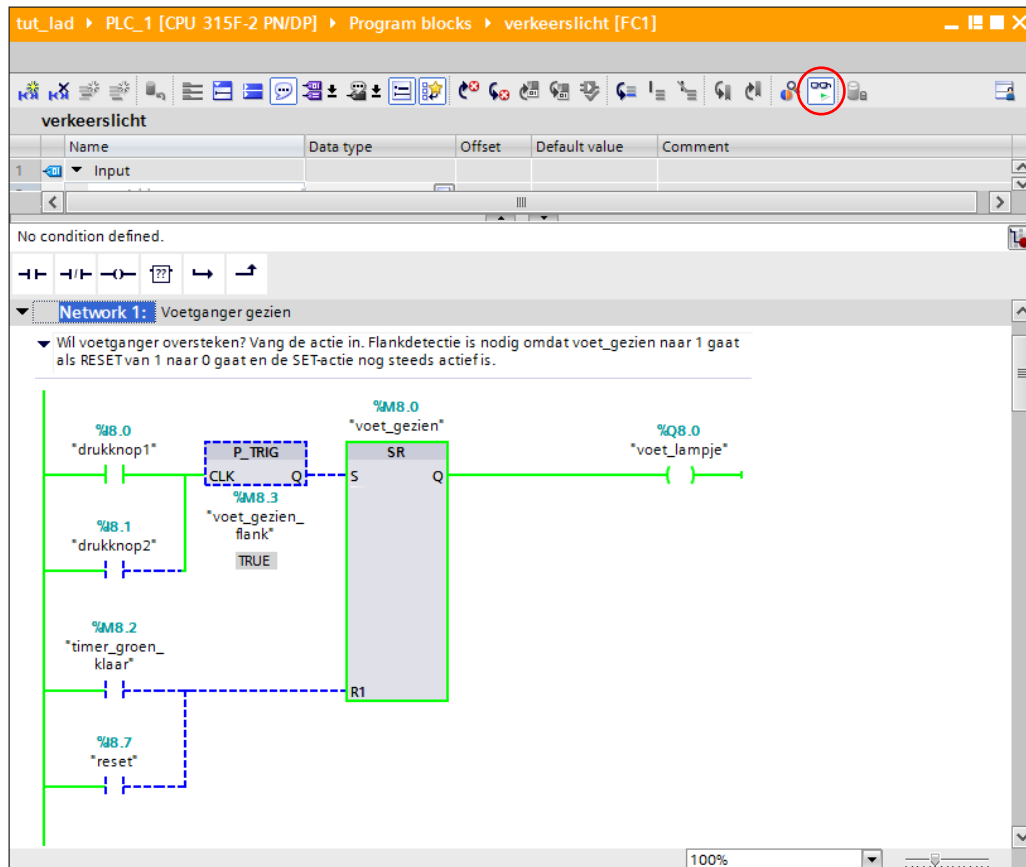
## 4.7 Monitoren van het programma en variabelen

Het programma is nu geladen; de werking kan worden getest. Start de PLC door de operatng mode schakelaar van STOP naar RUN te schakelen. De PLC gaat nu starten. Daarna kunnen de schakelaars op de simulatiemodule gebruikt worden om het programma te testen.

Het programma kan ook gevolgd (*monitor*) worden via de ladder-editor. Dubbelklik hiervoor op FC1 om deze bouwsteen te openen in de editor.



FC1 is nu geopend. Klik op het brilletje rechtsboven in de editor. De kleur van de titelbalk van de editor wordt nu oranje ten teken dat het hier om een online verbinding gaat. Zie figuur 4.27. In deze figuur is te zien dat contact drukknop1 is geactiveerd. Verder is het SR-element ook geactiveerd en daarmee ook de coil voet\_lampje.



Figuur 4.27: Monitoren eerste netwerk.

Het is ook mogelijk om de PLC tags te monitoren. Open de PLC tags-editor en klik op het brilletje linksboven. De PLC-tags worden nu real-time gemonitord. Zie figuur 4.28.

De tutorial is hiermee ten einde. De cursist wordt uitgedaagd het eenvoudige verkeerslichtsysteem uit te breiden met bijvoorbeeld een mogelijkheid tot het laten knipperen van het gele licht indien het systeem niet operationeel is. Hiervoor is een extra schakelaar nodig.

	Name	Data type	Address	Retain	Acces...	Visibl...	Monitor value	Comment
1	drukknop1	Bool	%I8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TRUE	Drukknop 1
2	drukknop2	Bool	%I8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALSE	Drukknop 2
3	reset	Bool	%I8.7		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALSE	Reset systeem
4	voet_gezien	Bool	%M8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TRUE	Voetganger gezien
5	timer_geel_klaar	Bool	%M8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TRUE	Timer voor geel klaar met tijdmeten
6	timer_groen_klaar	Bool	%M8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALSE	Timer voor groen klaar met tijdmeten
7	voet_gezien_flank	Bool	%M8.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TRUE	Flankdetector voet_gezien
8	voet_lampje	Bool	%Q8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TRUE	Voetganger gezien feedback lamp
9	auto_rood	Bool	%Q8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TRUE	Verkeerslicht auto rood
10	auto_geel	Bool	%Q8.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALSE	Verkeerslicht auto geel
11	auto_groen	Bool	%Q8.4		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALSE	Verkeerslicht auto groen
12	voet_rood	Bool	%Q8.6		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FALSE	Verkeerslicht voetganger rood
13	voet_groen	Bool	%Q8.7		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TRUE	Verkeerslicht voetganger groen
14	timer_geel	Timer	%T0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	SST#0MS	Timer voor auto geel
15	timer_groen	Timer	%T1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	SST#9S_800MS	Timer voor auto groen
16	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

Figuur 4.28: Monitoren PLC tags.

## 5. TUTORIAL S7-GRAPH

Veel productieprocessen kunnen worden opgelost met een *afloop*. Een (deel-)product volgt een vast aantal stappen waarin iets gebeurt. Deze stappen zijn goed van elkaar te onderscheiden. Zo'n programma kan goed worden geprogrammeerd als een *stappendiagram*. S7-Graph is een taal waarin zo'n stappendiagram kan worden geprogrammeerd.

In deze tutorial wordt uitgelegd hoe een S7-Graph programma ingevoerd moet worden. Eerst wordt de configuratie van de PLC ingevoerd, vervolgens een klein ladder-programma voor conditionering van de ingangen, dan het S7-Graph programma. Als laatste wordt dit programma en enkele variabelen gevolgd met de monitor.

### 5.1 Aanmaken nieuw project, PLC configuratie & download

Het aanmaken van een nieuw project, het opstellen van de PLC-configuratie en het downloaden van de configuratie gaat op identieke wijze zoals is beschreven in de paragrafen [4.1](#) t/m [4.3](#).

Opmerking: in de oudere versies van STEP7, voor het TIA Portal-tijdperk, was het mogelijk om de configuratie uit de PLC op te halen. In TIA portal is dat voor de S7-300 en S7-400 niet meer mogelijk.

Maak een nieuw project aan met de naam `tut_graph`, stel de configuratie van de PLC op en download de configuratie naar de PLC.

### 5.2 PLC tags invoeren

Natuurlijk moeten ook voor deze tutorial PLC tags ingevoerd worden. In figuur [5.1](#) zijn de PLC tags vermeld. Let vooral op `aantal_gedrukt`, dit is een merkerwoord van het type INT.

### 5.3 Een blok vooraf

Eerst wordt een ladderprogramma ingevoerd dat samenwerkt met het S7-Graph programma. Het conditioneert de ingangen, zodat de invoer via de drukknoppen makkelijk kan worden verwerkt. Maak hiervoor functie DrukknopConditioner (FC1) aan, zie figuur [5.2](#). Klik daarna op **OK**.

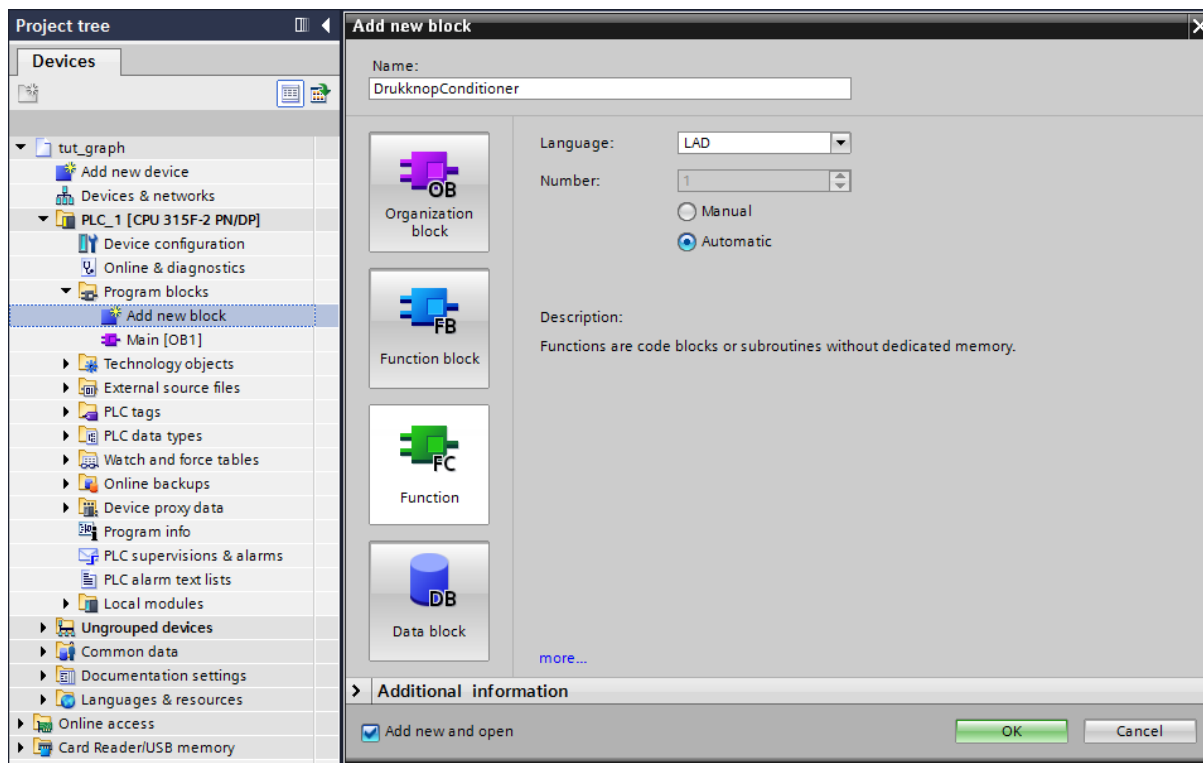
Voer het ladderprogramma in figuur [5.3](#) in.

### 5.4 Blokken, stappen, overgangen en acties

Naar de gebruiker toe laat de CPU en bijbehorend geheugen zich afbeelden als programmeerbare blokken. Er zijn verschillende typen, zie hiervoor bijlage [G](#). Voorlopig is het ge-

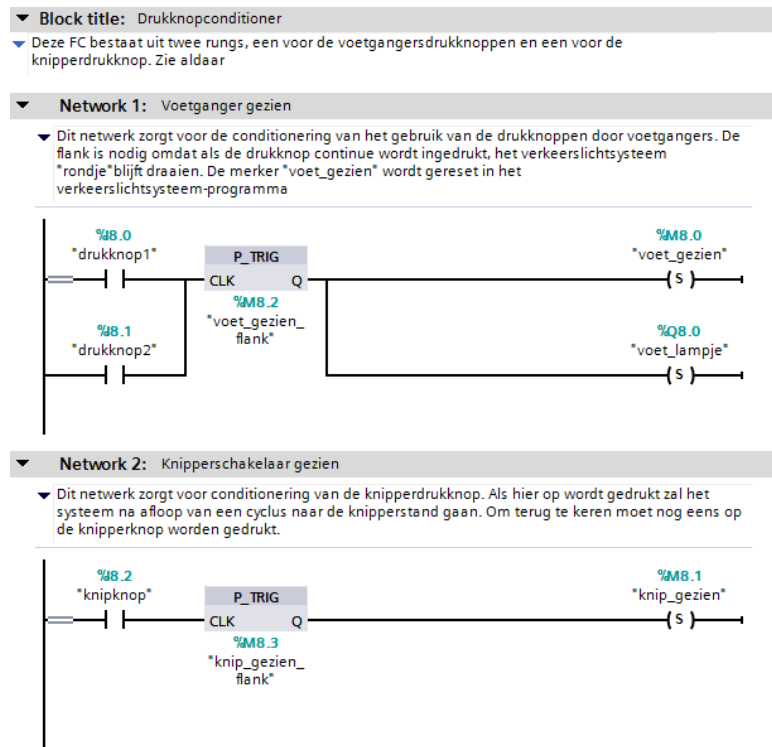
	Name	Data type	Address	Retain	Acces...	Visibl...	Comment
1	drukknop1	Bool	%I8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Drukknop 1
2	drukknop2	Bool	%I8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Drukknop 2
3	knipknop	Bool	%I8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Drukknop voor knippen geel licht
4	reset	Bool	%I8.7		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Reset systeem
5	voet_gezien	Bool	%M8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Voetganger gezien
6	knip_gezien	Bool	%M8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Knipperschakelaar gezien
7	voet_gezien_flank	Bool	%M8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Flankdetector voet_gezien
8	knip_gezien_flank	Bool	%M8.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Flankdetector knip_gezien
9	aantal_gedrukt	Int	%MW20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Aantal keer dat er op een drukknoop is gedrukt
10	voet_lampje	Bool	%Q8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Voetganger gezien feedback lamp
11	auto_rood	Bool	%Q8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht auto rood
12	auto_geel	Bool	%Q8.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht auto geel
13	auto_groen	Bool	%Q8.4		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht auto groen
14	voet_rood	Bool	%Q8.6		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht voetganger rood
15	voet_groen	Bool	%Q8.7		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Verkeerslicht voetganger groen
16	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figuur 5.1: PLC tags.



Figuur 5.2: Aanmaken functie DrukknopConditioner (FC1).

noeg te weten dat elk gebruikersprogramma moet starten in Organization Block 1 (OB1) en dat S7-Graph programma's worden opgeslagen in een functieblok (FB) met bijbehorend datablok (DB). Een S7-Graph programma bestaat uit een aantal stappen (*steps*), overgangen (*transitions*) tussen de stappen en acties (*actions*). Dit geheel wordt in S7-Graph een *sequencer* genoemd. In een stap kunnen diverse operaties worden uitgevoerd, zoals het aanzetten van een uitgang of het starten van een teller. Aan de overgangen kunnen voorwaarden worden verbonden zoals "ga verder als ingang 1 actief is". De oplettende lezer



Figuur 5.3: Laddernetwerken FC1.

ziet in dit alles een overeenkomst met toestandsmachines uit de digitale techniek.

Eigenlijk is S7-Graph niets anders dan een frontend die de ingevoerde sequencer omzet naar STL. Het is dus eigenlijk een compiler.

## 5.5 Aanmaken functieblok

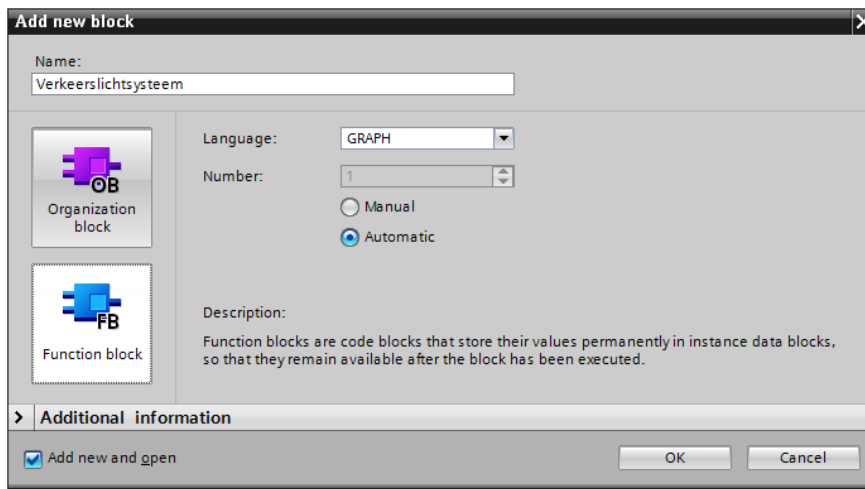
Een S7-Graph programma kan alleen in een functieblok (FB) ingevoerd worden. Aan een functieblok wordt een datablok gekoppeld. Eerst moet een FB worden aangemaakt waarin het S7-Graph programma kan "leven". Klik in TIA Portal links op Program Blocks. Dubbelklik daarna op Add new block om een functieblok in te voegen. Er wordt een dialoogvenster geopend waarin de eigenschappen van de FB kunnen worden vastgelegd (figuur 5.4). Kies verder als taal GRAPH. Klik op **OK** om de FB aan te maken.

## 5.6 Starten S7-Graph editor

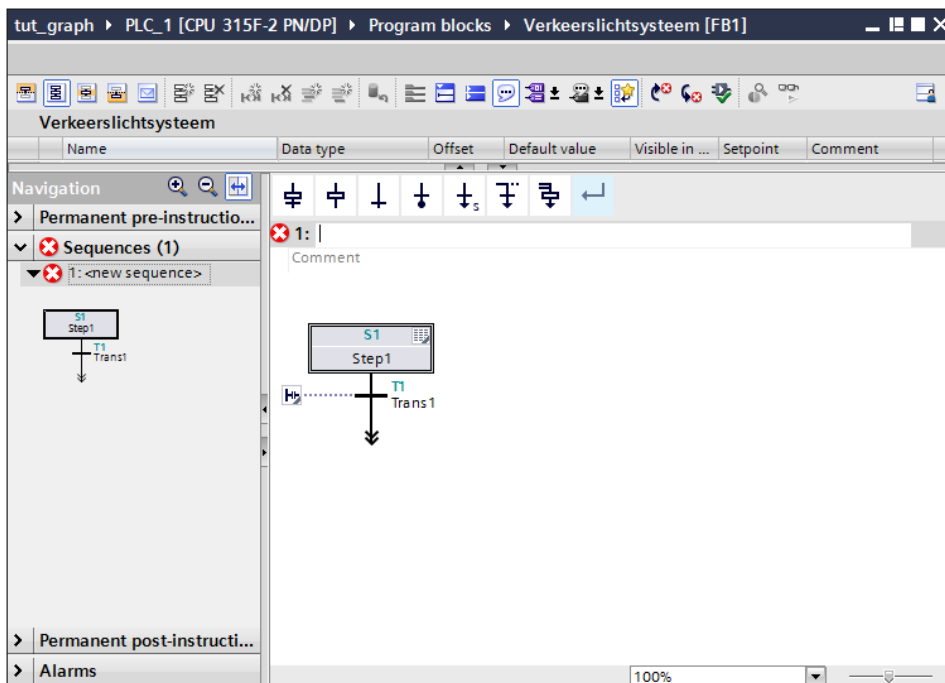
Nu is onder Program Blocks Verkeerslichtsysteem (FB1) te vinden. Dubbelklik hierop om de S7-Graph editor met FB1 te openen. Het openingsscherm is afgebeeld in figuur 5.5.

Het scherm is in twee panelen verdeeld. Links is een overzicht van de sequencer inclusief de permanente instructies te zien (tabblad Sequences). Er kan echter ook voor andere tabbladen gekozen worden. Rechts is het edit-paneel. Hierin worden de aanpassingen aan de sequencer verricht. In het edit-paneel wordt gelijk de eerste stap geplaatst. De dubbele ring geeft aan dat het hier om een beginstap (*initial step*) gaat. Als de sequencer wordt herstart, is dit de eerste stap.

Let op: er moet minstens één initial step in het programma aanwezig zijn. Een veel voor-



Figuur 5.4: Aanmaken FB1.

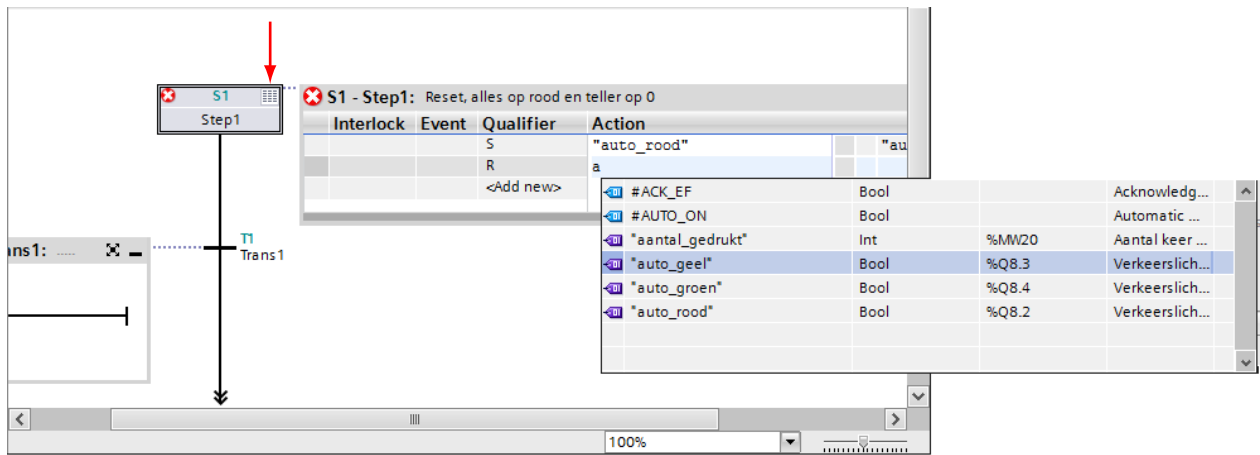


Figuur 5.5: Openings scherm Graph editor.

komend probleem is dat tijdens het programmeren de initial step wordt verwijderd, en er geen nieuwe wordt aangemerkt. De sequencer start dan niet als de PLC het programma begint te verwerken.

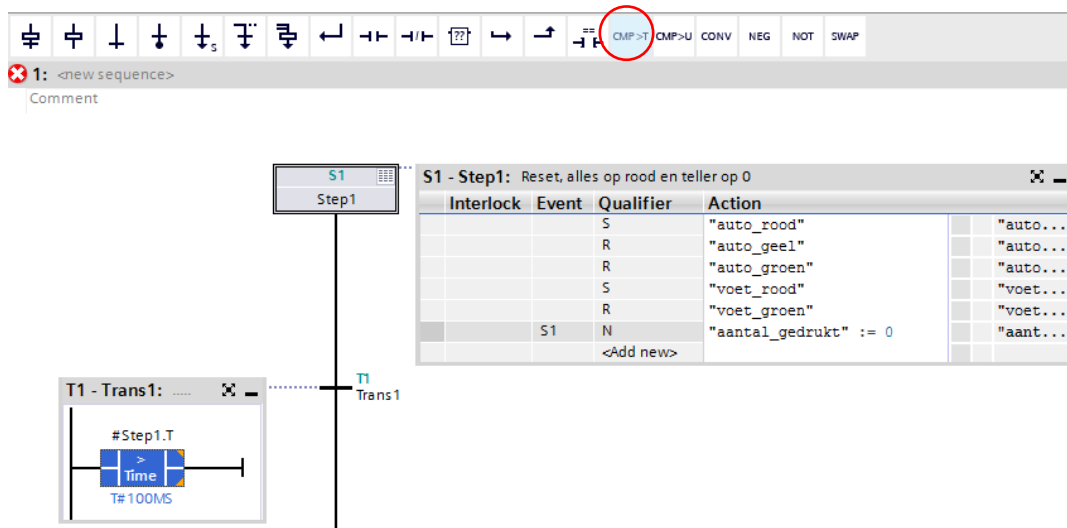
### 5.7 Invoeren eerste stap

Het gebruik van de editor vergt, net als bij de laddereditor, enige routine. Als vuistregel geldt: als iets veranderd, ingevoegd of verwijderd moet worden, selecteer dan eerst het onderdeel en voer dan de operatie uit. Namen van variabelen, zoals auto\_rood, kunnen zonder aanhalingstekens worden ingevoerd, de editor plaatst ze zelf. In figuur 5.6 is te zien hoe een actie wordt ingevoerd. Klik op het tabel-pictogram (rode pijl) om de actielijst te openen.



Figuur 5.6: Invoeren actie.

In figuur 5.7 is te zien hoe een overgangsconditie, in dit geval een vergelijkfunctie, wordt ingevoerd. Ook dit vergt enige routine. Selecteer in de overgangsconditie de horizontale ruiting en klik daarna op het pictogram CMP>T boven in het paneel. Zie de rode cirkel.



Figuur 5.7: Invoeren overgangsconditie.

Merk op dat de overgangscondities in LAD worden ingevoerd. Het is ook mogelijk om de overgangscondities in FBD in te voeren. Hiervoor moet de programmeertaal gewijzigd worden. Zie paragraaf 8.14.

In figuur 5.8 is de complete stap met acties en overgangsconditie te zien. Programmeer de eerste stap precies zoals in deze figuur is weergegeven.

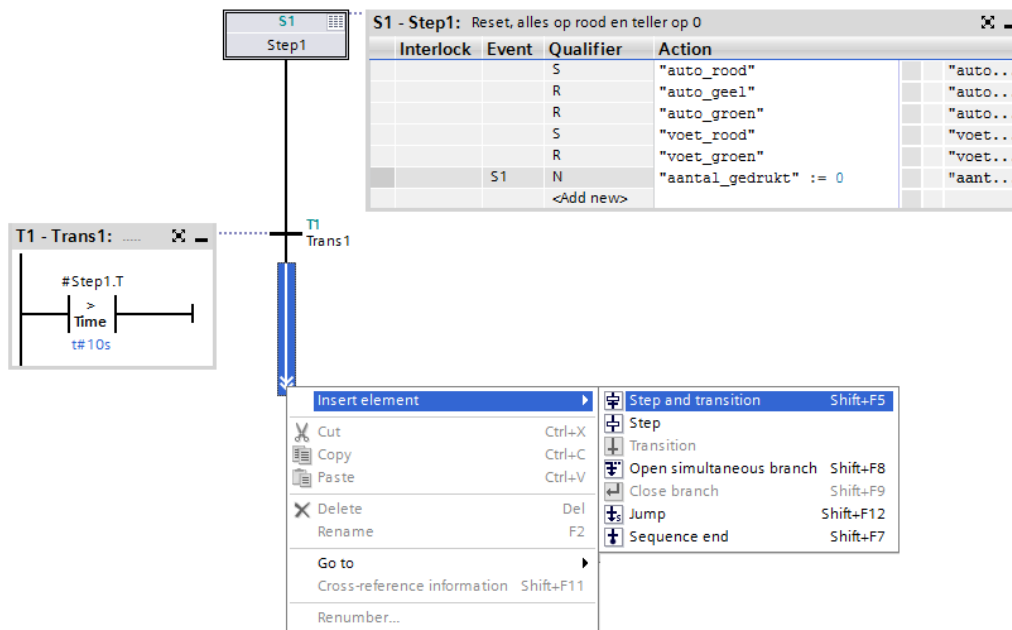
## 5.8 Invoeren volgende stappen

Aan de eerste stap moet nu de vervolgstap gekoppeld worden. Selecteer hiervoor de overgang (niet de conditie), klik op de rechter muisknop en selecteer **Insert element** → **Step and transition**. Zie figuur 5.9. Een nieuwe stap wordt nu toegevoegd.

In figuur 5.10 zijn alle stappen, acties en overgangscondities afgebeeld. Voer deze zo in.



Figuur 5.8: Eerste stap volledig.



Figuur 5.9: Aanmaken nieuwe stap.

Let op de laatste stap (stap 6). Daar wordt een *jump* gemaakt naar stap 2. Boven stap 2 is dan een horizontale pijl te zien. Deze is pas te zien na de plaatsing van de jump.

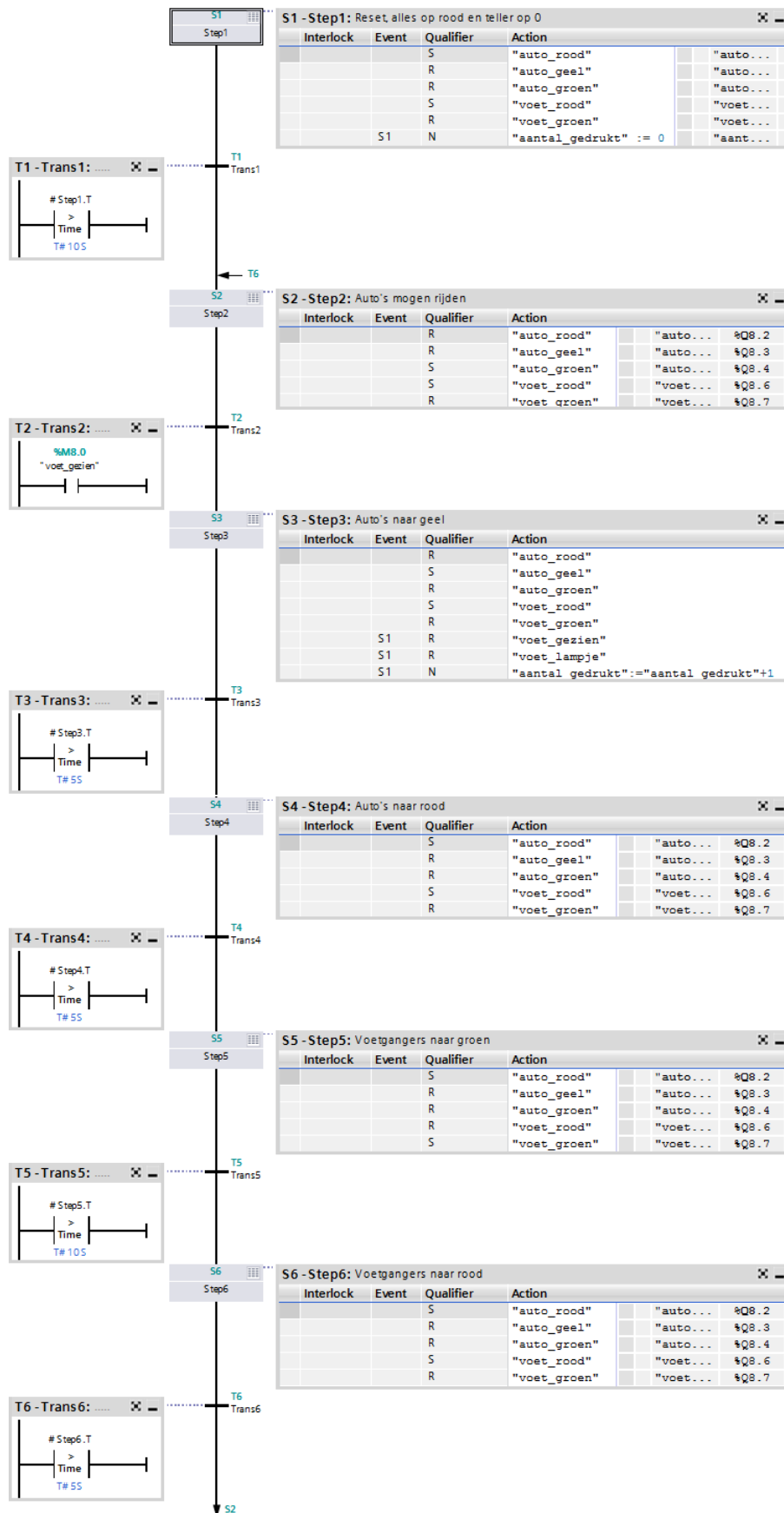
Nu moet de sequencer nog gecompileerd worden. Selecteer in het rechter paneel onder het menuonderdeel Program Block de FB met het Graph-programma. Klik daarna op de rechter muisknop en selecteer **Compile** → **Software (changes only)**. Als er geen fouten worden geconstateerd slaagt de compilatie, zie figuur 5.11.

## 5.9 Invoeren OB1

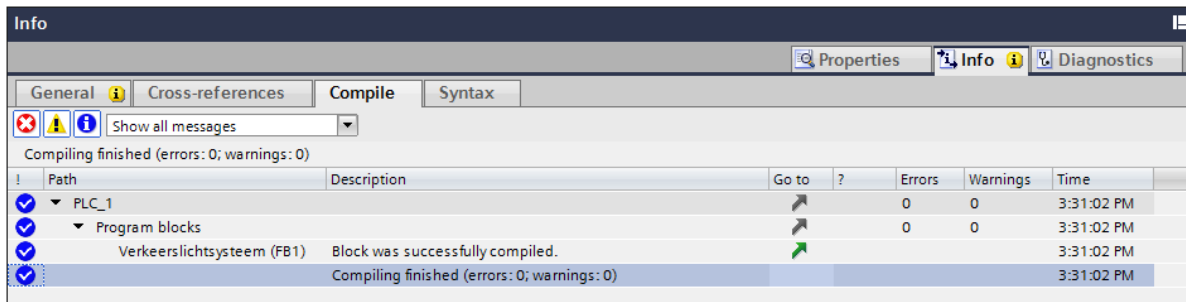
Nu moet nog OB1 ingevoerd worden, die FC1 en FB1 moet aanroepen. Zonder OB1 kan de PLC geen programma draaien. OB1 wordt hier geprogrammeerd in LAD. Open OB1 door erop te dubbelklikken. In het middenpaneel zijn de rungs van OB1 te zien. Sleep nu FC1 naar de eerste rung. Sleep vervolgens FB1 naar de tweede rung. Zie de rode pijlen in figuur 5.12.

TIA Portal zal nu vragen om een *Instance Data Block* aan te maken. Zoals bekend is, moet

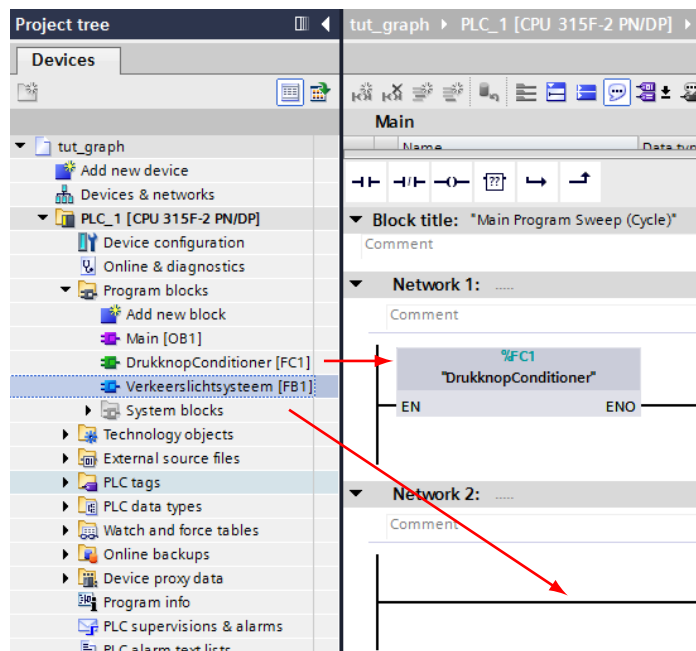




Figuur 5.10: Volledige Graph-sequencer.

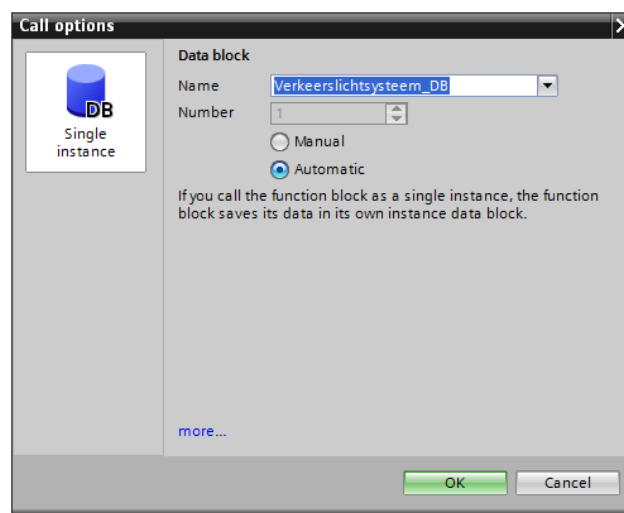


Figuur 5.11: Compilatie is gelukt.



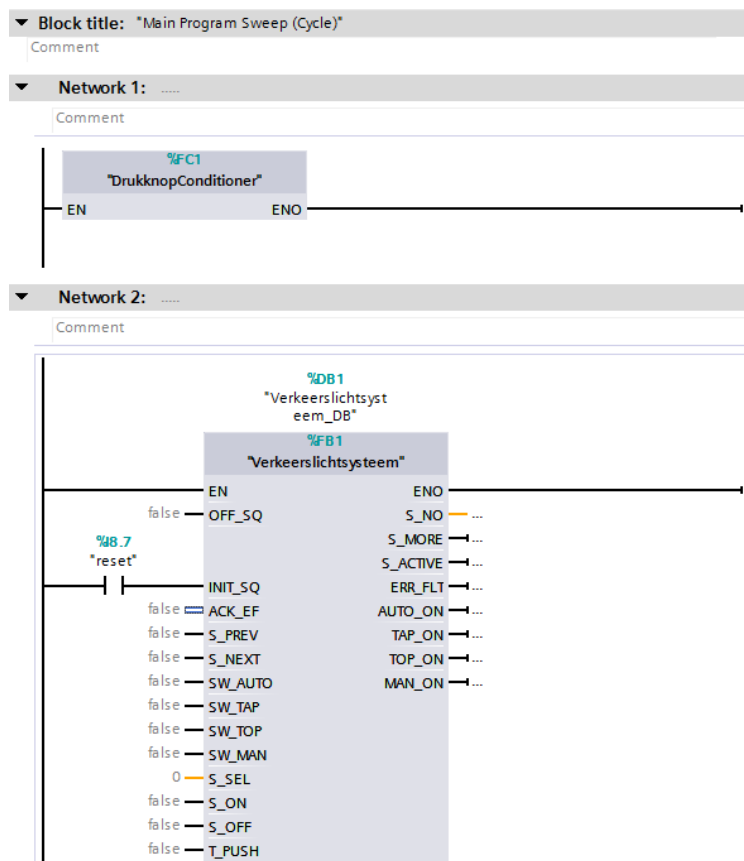
Figuur 5.12: Invoeren OB1.

aan elke FB een DB gekoppeld worden. In figuur 5.13 is te zien dat TIA Portal een DB aanmaakt met nummer 1. Klik op de knop **OK**.



Figuur 5.13: Aanmaken Data Block voor FB1.

Het programma van OB1 bevat twee rungs, één voor het aanroepen van FC1 en één voor het aanroepen van FB1, die DB1 gebruikt. Let heel goed op de aanroep van FB1. Zodra dit wordt ingevoerd, zal de LAD-editor gegevens toevoegen. Dit zijn de formele parameters van FB1. Alleen INIT\_SQ moet gekoppeld worden aan de tag reset, de rest wordt niet gebruikt. De reset kan dan gebruikt worden om de sequencer te herstarten. Zie figuur 5.14 voor de inhoud van OB1. Sla de gegevens op en sluit de editor.



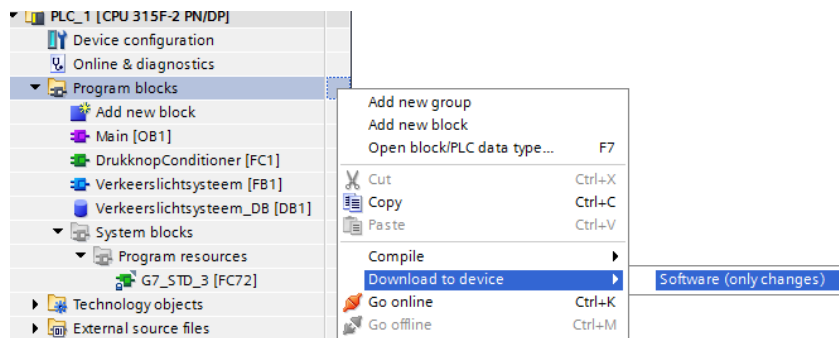
Figuur 5.14: Volledige OB1 voor aanroepen FC1 en FB1.

## 5.10 Laden PLC met programma

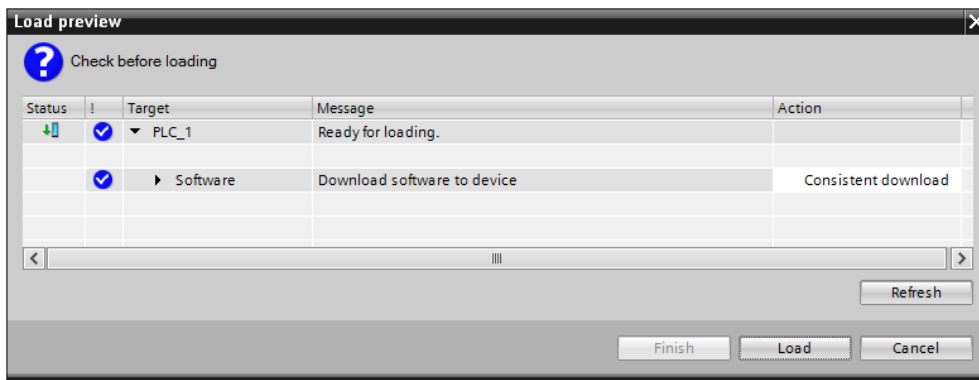
Het laden van het programma gaat zoals is beschreven in hoofdstuk 4. Zorg ervoor dat de PLC in STOP staat. Selecteer onder PLC\_1 het menu-onderdeel Program Blocks. Klik op de rechter muisknop en selecteer **Download to device** → **Software (only changes)**. Zie figuur 5.15.

Merk op dat onder het menu-onderdeel System Blocks → Program resources functie FC72 is aangemaakt. Dit blok is door de Graph-compiler aangemaakt. FC72 is een blok waarin generieke (voor ieder hetzelfde) code wordt geplaatst voor S7-Graph ingevoerde blokken. Dit blok moet ook in de PLC geladen worden anders werken de Graph-programma's niet.

TIA Portal komt met een dialoog waarin bevestiging wordt gevraagd voor de download. Zie figuur 5.16. Klik op de knop **Load**. Na download kan het programma getest worden.



Figuur 5.15: Download alle blokken.



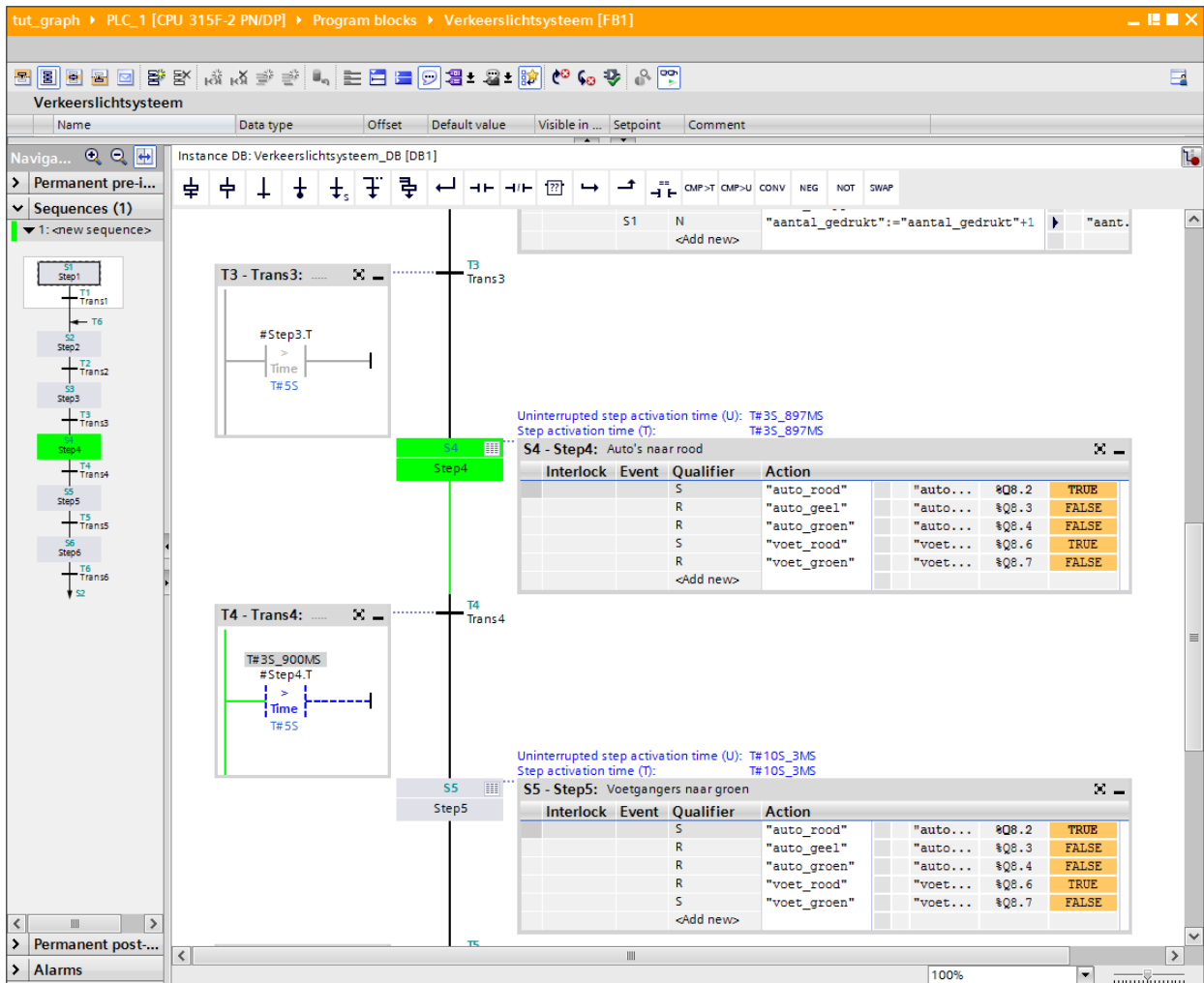
Figuur 5.16: Download preview.

## 5.11 Monitoren van het programma en variabelen

Het S7-Graph-programma kan gemonitord worden, d.w.z. gevolgd worden. De S7-Graph-editor bevat hiervoor een speciale modus. Open de editor door op FB1 te dubbelklikken. Aan de bovenrand is een pictogram met een brilletje te vinden. Klik hier één keer op. De editor zal nu communiceren met de PLC en weergeven in welke stap de sequencer zich bevindt (groen gekleurde stap). Ook de conditie van de transitie wordt weergegeven. Zie figuur 5.17.

De variabelen kunnen op eenzelfde wijze gemonitord worden. Open de PLC tags-lijst en klik op het brilletje. De variabelen worden dan continu bijgewerkt met de meest actuele waarden. Zie figuur 5.18.

De tutorial is hiermee ten einde.



Figuur 5.17: Download preview.

The screenshot shows the 'Default tag table' in TIA Portal. The table contains the following data:

Name	Data type	Address	Retain	Acces...	Visibl...	Monitor value	Comment
drukknop1	Bool	%I8.0		✓	✓	FALSE	Drukknop 1
drukknop2	Bool	%I8.1		✓	✓	FALSE	Drukknop 2
knipknop	Bool	%I8.2		✓	✓	FALSE	Drukknop voor knippen geel licht
reset	Bool	%I8.7		✓	✓	FALSE	Reset systeem
voet_gezien	Bool	%M8.0		✓	✓	FALSE	Voetganger gezien
knip_gezien	Bool	%M8.1		✓	✓	TRUE	Knipperschakelaar gezien
voet_gezien_flank	Bool	%M8.2		✓	✓	FALSE	Flankdetector voet_gezien
knip_gezien_flank	Bool	%M8.3		✓	✓	FALSE	Flankdetector knip_gezien
aantal_gedrukt	Int	%MW20		✓	✓	6	Aantal keer dat er op een drukknoop is gedr.
voet_lampje	Bool	%Q8.0		✓	✓	FALSE	Voetganger gezien feedback lamp
auto_road	Bool	%Q8.2		✓	✓	FALSE	Verkeerslicht auto rood
auto_geel	Bool	%Q8.3		✓	✓	FALSE	Verkeerslicht auto geel
auto_groen	Bool	%Q8.4		✓	✓	TRUE	Verkeerslicht auto groen
voet_road	Bool	%Q8.6		✓	✓	TRUE	Verkeerslicht voetganger rood
voet_groen	Bool	%Q8.7		✓	✓	FALSE	Verkeerslicht voetganger groen

Figuur 5.18: PLC tags online bekijken.

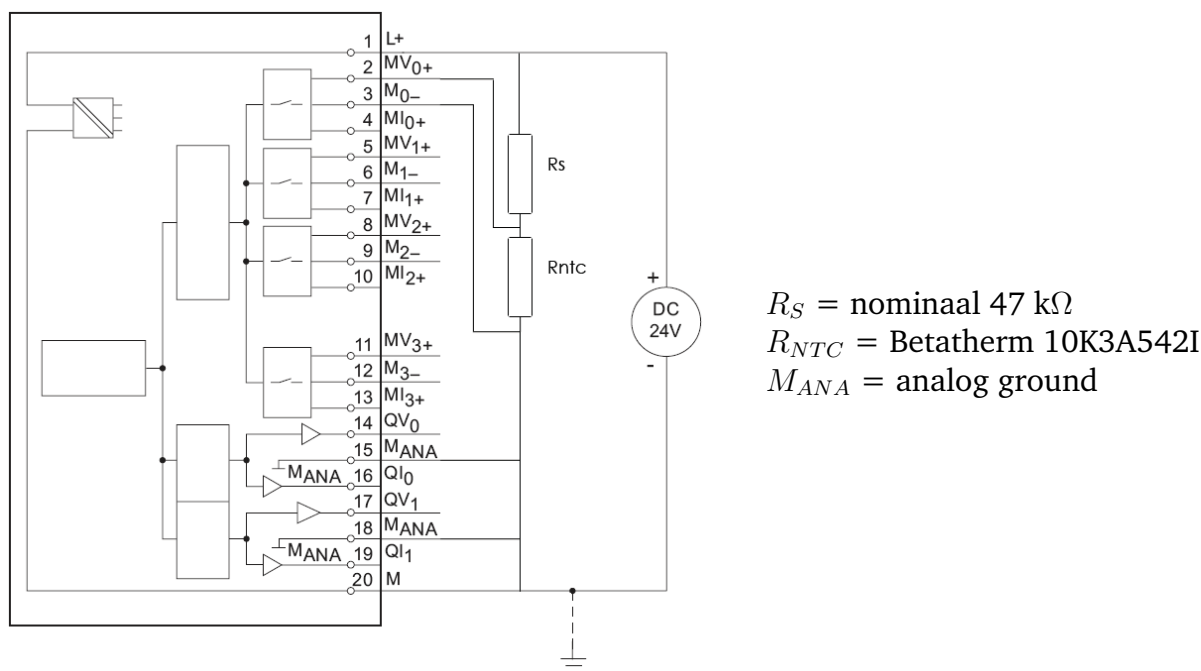
## 6. TUTORIAL SCL

In deze tutorial wordt de programmeertaal SCL (Structured Command Language) gebruikt. SCL is een variant op de IEC-taal ST. De taal lijkt erg veel op Pascal. Het is een gestructureerde taal waarin mogelijkheden als lussen en beslissingen eenvoudig kunnen worden ingevoerd. Het werken met complexe datastructuren als arrays en structures en met complexe rekenkundige functies is veel eenvoudiger dan in STL of LAD.

Voor programmeurs van andere talen zoals C zal het werken met SCL eenvoudig uitpakken. Deze tutorial laat slechts een klein deel van SCL zien. Er wordt niet ingegaan op de syntax en mogelijkheden. Zie hiervoor de SCL-handleiding.

### 6.1 Analoge ingang en temperatuur meten

In deze tutorial wordt gebruik gemaakt van een analoge ingang. Er wordt een thermometer gebouwd met behulp van een NTC-weerstand van Betatherm [5] en een serieweerstand. De spanning over de NTC is een maat voor de temperatuur. Het aansluitschema is te zien in figuur 6.1. Tevens wordt er een alarmering ingebouwd.



Figuur 6.1: Aansluitschema van de NTC- en serieweerstand.

Opmerking: op de analoge ingang mag maximaal 10 V aangeboden worden, zie hiervoor paragraaf 3.1.4. Zie bijlage H voor de afleiding van de formule voor de temperatuur. Als alternatief voor de NTC-weerstand kan een potentiometer gebruikt worden.

Er worden in totaal vijf blokken aangemaakt die allemaal geschreven worden in SCL. Het

eerste blok bevat de code om de ruwe ADC-waarde om te rekenen naar een temperatuur. Hier is vooral de kracht van SCL te zien voor wat betreft complexe rekenkundige functies. Het tweede blok dat wordt aangemaakt test of de berekende temperatuurwaarde binnen een bepaald bereik ligt. Alarmen worden geactiveerd om aan te geven of de temperatuur binnen of buiten het bereik ligt. Dit blok laat zien hoe beslissingen kunnen worden genomen. Het derde blok laat zien hoe een geheugenelement, in dit geval een RS-flipflop, kan worden beschreven. Het vierde blok laat zien hoe in SCL gemakkelijk met arrays en structures kan worden omgegaan. Bij dit blok wordt een *Instance Data Block* (IDB) gebruikt waarmee het mogelijk is om een blok van geheugen te voorzien dat over aanroepen van het blok heen behouden blijft. Het vijfde en laatste blok betreft een interruptroutine die ervoor zorgt dat de temperatuur eens in de 10 seconden wordt geregistreerd. Ten slotte wordt blok OB1 aangemaakt in LAD. Deze roept de andere vier blokken aan.

## 6.2 Aanmaken nieuw project, PLC configuratie & download

Maak een nieuw project aan met de naam `tut_scl`, stel de configuratie van de PLC op en download de configuratie naar de PLC.

Het aanmaken van een nieuw project, het opstellen van de PLC-configuratie en het downloaden van de configuratie gaat op identieke wijze zoals is beschreven in de paragrafen 4.1 t/m 4.3.

Opmerking: in de oudere versies van STEP7, voor het TIA Portal-tijdperk, was het mogelijk om de configuratie uit de PLC op te halen. In TIA portal is dat voor de S7-300 en S7-400 niet meer mogelijk.

## 6.3 PLC tags invoeren

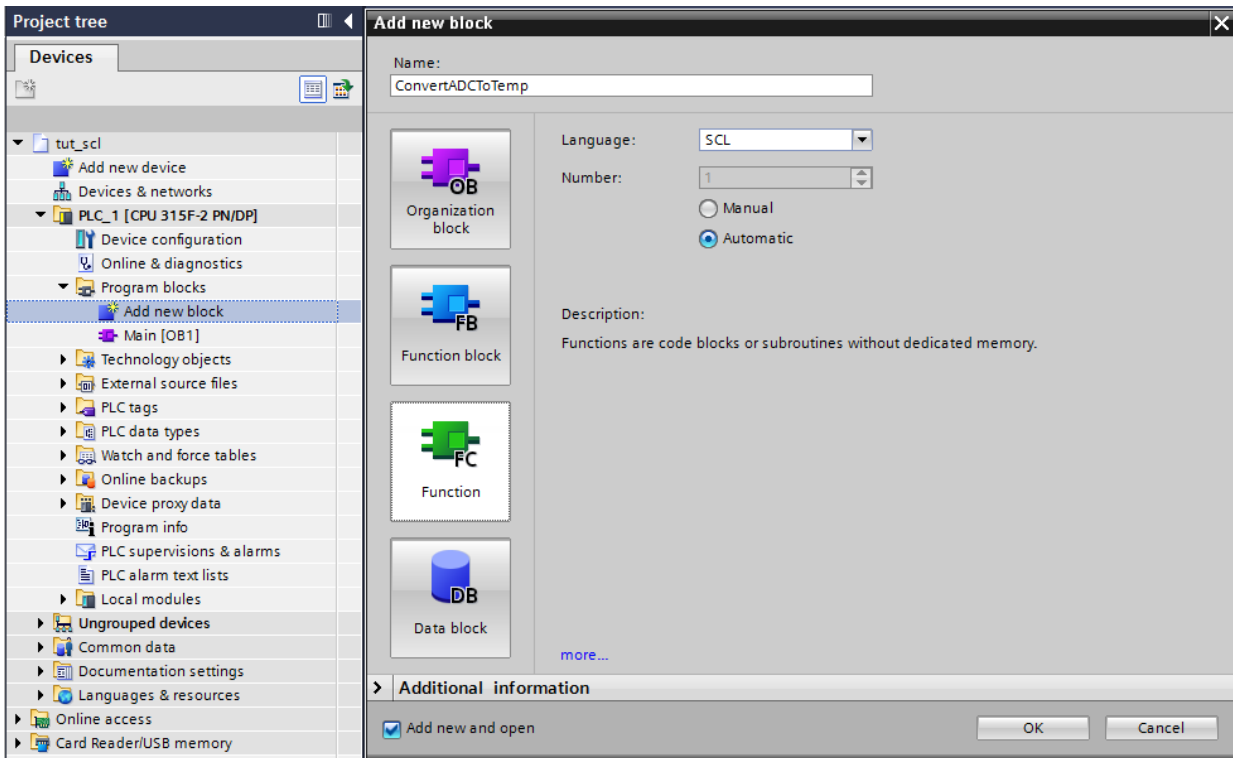
Natuurlijk moeten ook voor deze tutorial PLC tags ingevoerd worden. In figuur 6.2 zijn de PLC tags vermeld. Merk ook het grote aantal Reals.

	Name	Data type	Address	Retain	Acces...	Visibl...	Comment
1	ANALOG_IN	Word	%IW272		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	The raw ADC value
2	VMEAS	Real	%MD20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Measured voltage across the NTC
3	RMEAS	Real	%MD24		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Calculated resistance of the NTC
4	TDEGREE	Real	%MD28		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Calculated temperature
5	IMEAS	Real	%MD32		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Calculated current through the NTC
6	PMEAS	Real	%MD36		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Calculated power dissipated by the NTC
7	TEMPTOOHIGH	Bool	%M18.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	True = Temperature too high (output)
8	TEMPTOLOW	Bool	%M18.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	True = Temperature too low (output)
9	RESETALARMS	Bool	%M18.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	True = Alarms will reset (input)
10	ALARMSHOLDING	Bool	%M18.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Holding relay for alarms
11	RECORD_TEMP	Bool	%M18.4		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Should we record a temperature?
12	AVERAGE_TEMP	Real	%MD40		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Average temperature
13	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figuur 6.2: PLC tags.

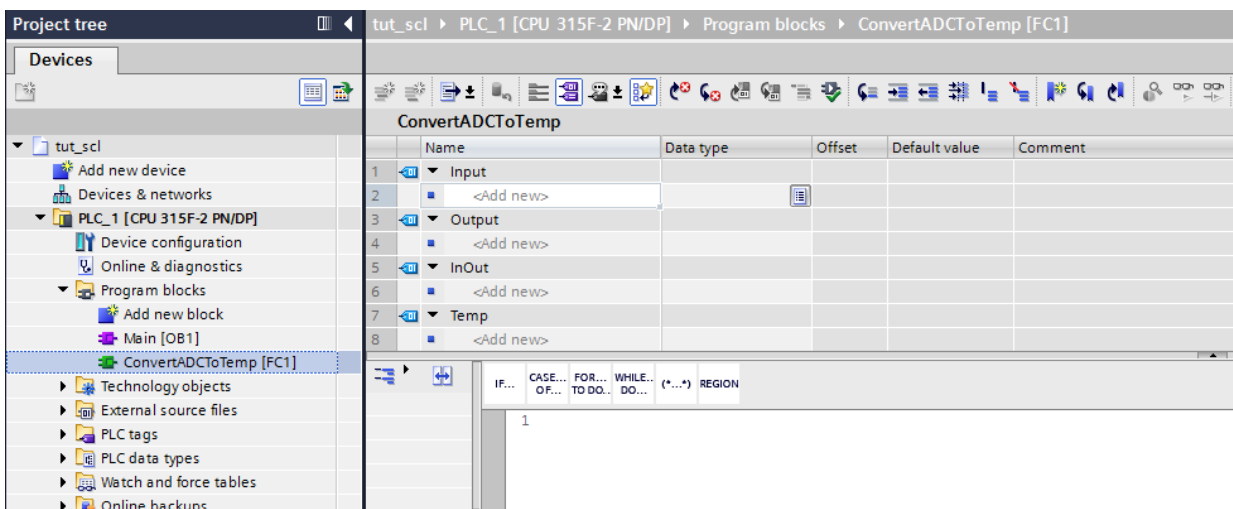
## 6.4 De functie ConvertADCToTemp

Dubbelklik in de project tree onder het onderdeel Program blocks op het onderdeel Add new block. Er wordt een dialoog geopend waarin een nieuw blok kan worden aangemaakt. Klik op de pictogram Function. Geef het blok de naam ConvertADCToTemp en selecteer als programmeertaal SCL. Klik daarna onderin op de knop **OK**. Zie figuur 6.3.



Figuur 6.3: Aanmaken nieuwe functie ConvertADCToTemp (FC1).

Het blok wordt nu aangemaakt en daarna wordt de SCL-editor gestart. Dit is te zien in figuur 6.4.



Figuur 6.4: Functie ConvertADCToTemp aangemaakt en de SCL-editor is gestart.

Nu moeten eerst de formele parameters worden ingevoerd. In TIA Portal wordt dit de



block interface van het blok genoemd, te zien in figuur 6.5. Er is slechts één ingangspaarparameter, de ruwe data van de ADC. Verder zijn er vier uitgangspaarparameters, twee tijdelijke variabelen, negen constanten en één teruggaveparameter te zien.

	Name	Data type	Offset	Default value	Comment
1	Input				
2	ADC_VALUE	Word			ADC raw data
3	Output				
4	VNTC	Real			Measured voltage across the NTC
5	INTC	Real			Calculated current through the NTC
6	RNTC	Real			Calculated value of the NTC
7	PNTC	Real			Calculated power of the NTC
8	InOut				
9	<Add new>				
10	Temp				
11	Z	Real	0.0		Transfer function of the voltage divider
12	ZI	Int	4.0		For conversion from Word to Int
13	Constant				
14	VPOWER	Real		24.0	Power supply voltage
15	VANALOG_MAX	Real		10.0	Maximum voltage on the ADC input
16	ADCVALUE_AT_VANALO...	Int		27648	ADC value at maximum ADC voltage
17	A	Real		0.02010974	A parameter of the NTC
18	B	Real		3899.387	B (beta) parameter of the NTC
19	RS	Real		47000.0	Series resistor
20	RIN	Real		100000.0	Input resistance of the ADC input
21	K_TO_C	Real		273.15	Conversion from Kelvin to Celsius
22	DISSIPATION_CONSTANT	Real		0.002	Dissipation constant
23	Return				
24	ConvertADCToTemp	Real			Return value is temperature in degree Celsius

Figuur 6.5: De interface van functie ConvertADCToTemp.

Nadat de interface is ingevoerd wordt overgegaan tot het invoeren van de code. De code is te zien in figuur 6.6. Neem de code over zoals in de figuur te zien is.

```

IF... CASE... FOR... WHILE... (*..*) REGION
OF... TO DO... DO...

1 // ADC value to integer
2 #ZI := WORD_TO_INT(#ADC_VALUE);
3
4 // Voltage measured
5 #VNTC := (1.0 * #ZI) / #ADCVALUE_AT_VANALOG_MAX * #VANALOG_MAX;
6
7 // Calculate Z as function of ...
8 #Z := #VNTC / #VPOWER;
9
10 // Rntc in Ohms. Function found with the aid of Maple 10.
11 #RNTC := - (#VNTC * #RS * #RIN / (- #VPOWER * #RIN + #VNTC * #RIN + #VNTC * #RS));
12
13 // Calculate current through the NTC
14 #INIC := #VNTC / #RNTC;
15
16 // Calculate power dissipated by the NTC
17 #PNIC := #VNTC * #INIC;
18
19 // Temperature in degree C. Function found with the aid of Maple 10. This is the
20 // oreturn value of the function. There is compensation for the self-heating effect
21 #ConvertADCToTemp := #B / LN(#Z * #RS * #RIN / (#A * (#RIN - #Z * #RS - #Z * #RIN))) - #K_TO_C - #PNIC / #DISSIPATION_CONSTANT;

```

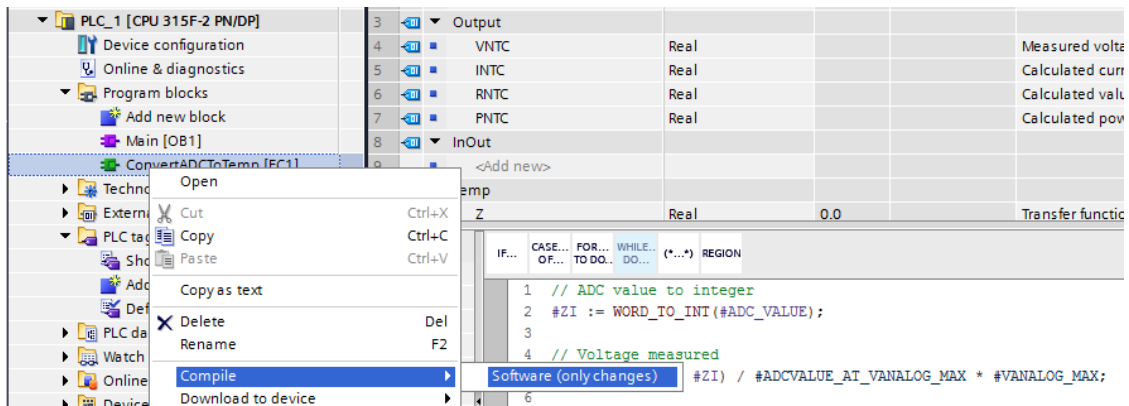
Figuur 6.6: De SCL-code van functie ConvertADCToTemp.

Opmerkingen: de berekening van #ConvertADCToTemp kan eenvoudiger maar in dit voorbeeld wordt gedemonstreerd dat complexe rekenkundige bewerkingen in SCL gemakkelijk geprogrammeerd kunnen worden, zie voor een afleiding van de formules bijlage H. De volledige code van de functie, inclusief de interface, is te vinden in bijlage B.

## 6.5 Compileren SCL-code

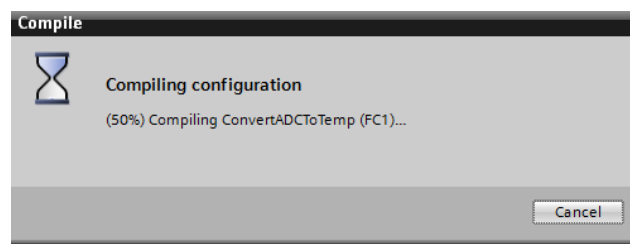
Nu het programma ingevoerd is, moet het gecompileerd worden. De *compiler* vertaalt de SCL-code naar STL, de assemblertaal voor de PLC.

Selecteer in de project tree de functie ConvertADCToTemp en klik op de rechter muisknop. Er wordt een contextmenu geopend. Selecteer daarin Compile→Software (only changes). Zie figuur 6.7. Een alternatief is de sneltoetscombinatie **Ctrl+B**.



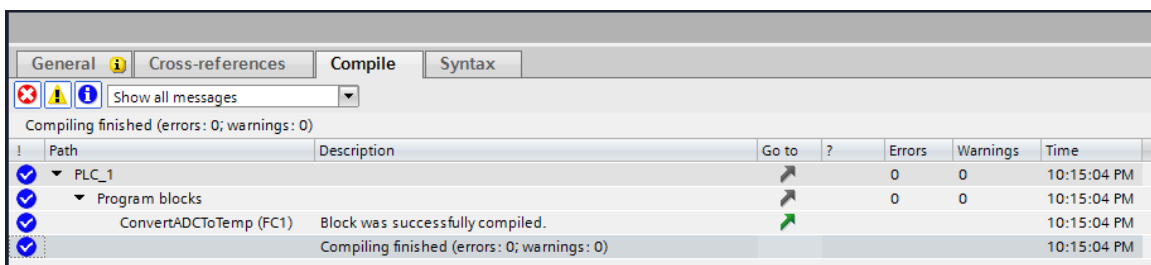
Figuur 6.7: Selecteren compilatie.

TIA Portal zal nu de compilatie starten. Er wordt een voortgangsvenster getoond, zie figuur 6.8.



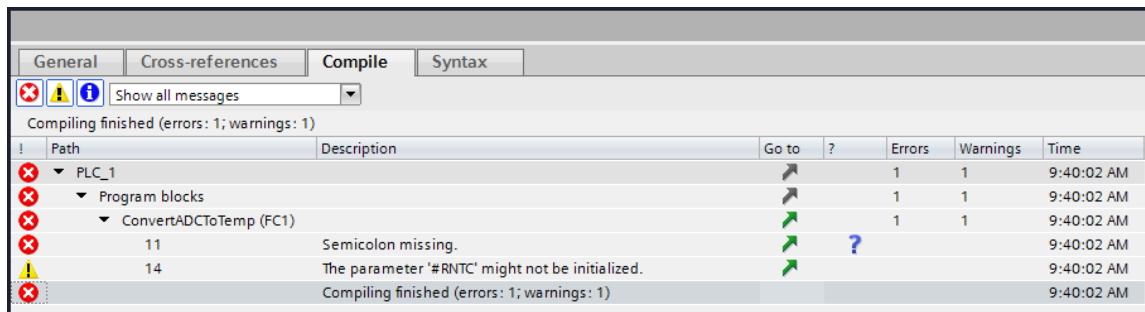
Figuur 6.8: Voortgang compilatie.

In het paneel middenonder is te zien dat de compilatie is geslaagd. Zie figuur 6.9.



Figuur 6.9: Compilatie geslaagd.

Eventuele fouten worden gerapporteerd en moeten verbeterd worden. Zie voor een voorbeeld van een mislukte compilatie figuur 6.10. Merk op dat de waarschuwing in regel 14 volgt uit de syntaxfout in regel 11.

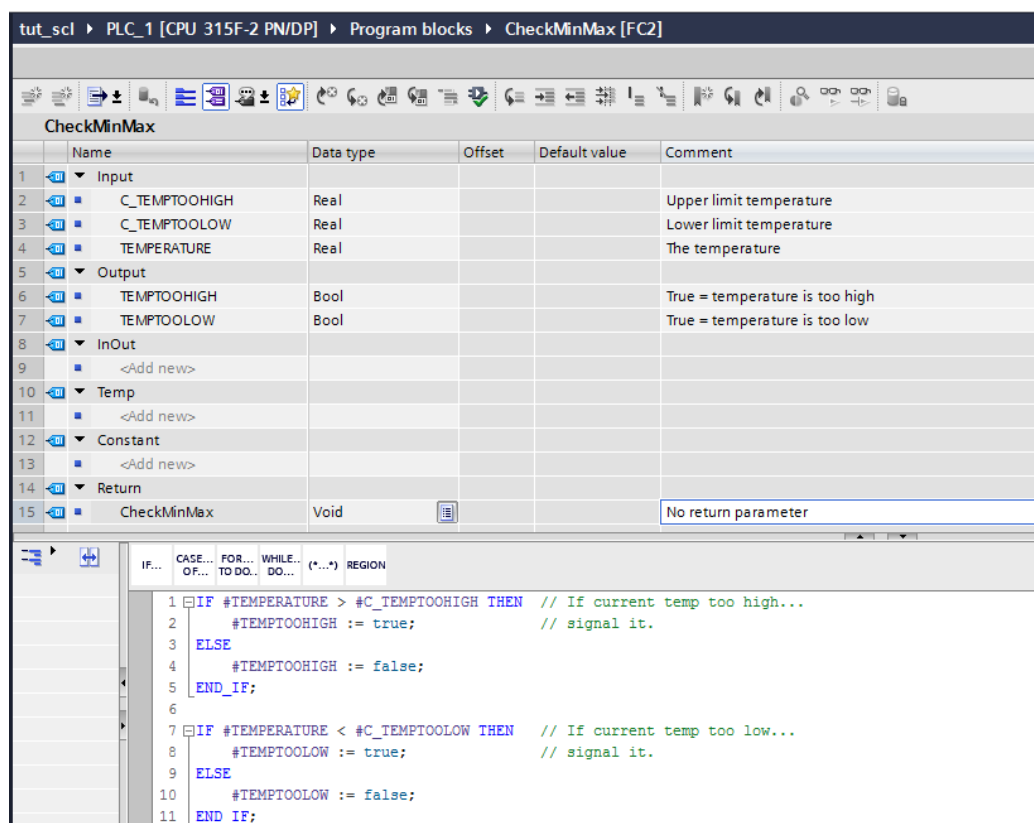


Figuur 6.10: Voorbeeld van een mislukte compilatie.

## 6.6 De functie CheckMinMax

Er wordt nu een tweede SCL-bestand aangemaakt. De code wordt in FC2 geplaatst en zet twee merkerbits indien de temperatuur te hoog of te laag is. Maak een nieuw SCL-bestand aan en noem dit CheckMinMax.

De code, inclusief de interface, is weergegeven in figuur 6.11. Let vooral op de teruggaveparameter van de functie, deze is van het type Void, oftewel er is geen teruggaveparameter. De teruggave wordt geregeld via twee output-parameters.

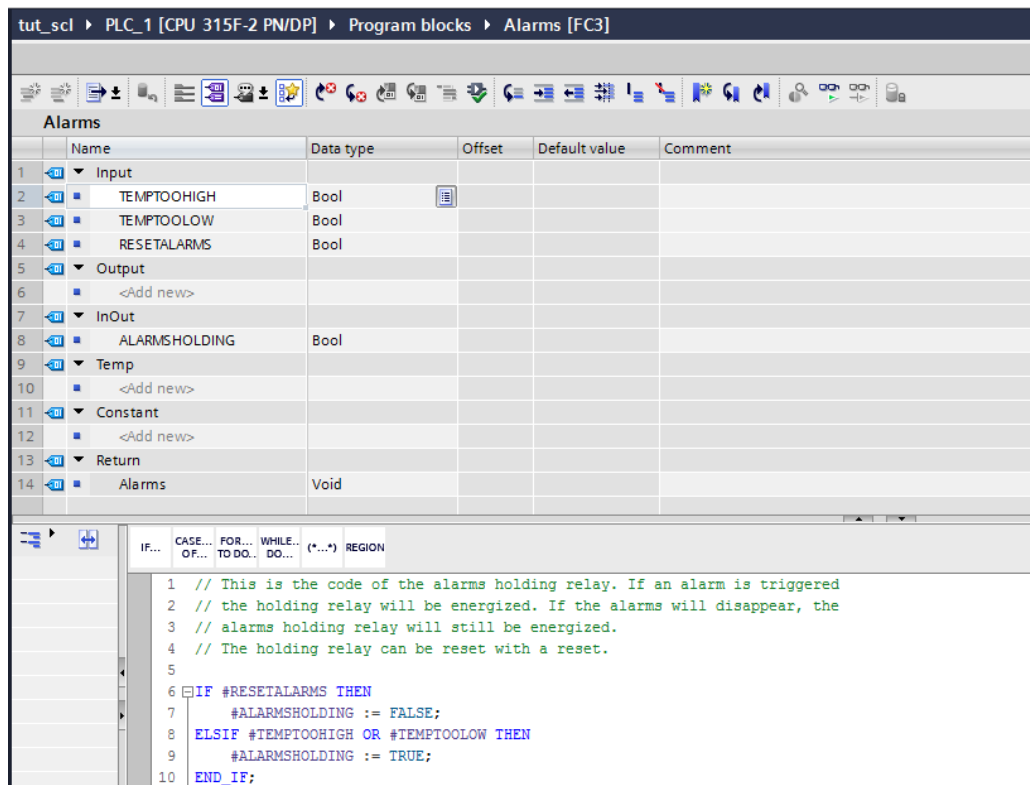


Figuur 6.11: De SCL-code van functie CheckMinMax (FC2).

De code van CheckMinMax is zeer eenvoudig. Het zijn slechts twee simpele IF-statements. Hiermee is ook goed te zien dat SCL gebruikt kan worden voor simpele beslissingen. Natuurlijk had de code veel eenvoudiger gekund en doorgewinterde PLC-programmeurs hadden dit in LAD opgelost. In bijlage C is de code nog eens afgebeeld.

## 6.7 De functie Alarms

Het derde SCL-bestand beschrijft een *holding alarm* middels een eenvoudige RS-flipflop. De code wordt geprogrammeerd in de functie CheckMinMax en is te zien in figuur 6.12. Ook hier is het teruggavetype `Void`. Het geheugen van de RS-flipflop wordt gerealiseerd via een `InOut`-parameter. Bij gebruik van de functie wordt hieraan een merkerbit gekoppeld.



Figuur 6.12: De SCL-code van functie *Alarms* (FC3).

Het holding alarm wordt geactiveerd als één van de twee alarmingangen geactiveerd zijn en wordt gereset als de ingang `RESETALARMS` geactiveerd is. Merk op dat de reset overheersend is ten opzichte van de alarmingangen. De code, inclusief de interface, is te vinden in bijlage D.

## 6.8 Het functieblok AverageTemp

Het vierde en laatste blok dat in SCL geschreven wordt betreft een functieblok met de naam `AverageTemp`. Een functieblok wordt gekoppeld aan een Instance Data Block dat ervoor zorgt dat een functieblok een zogenoemd *statisch geheugen* heeft. Dat is geheugen dat behouden blijft over aanroepen van het functieblok heen. Voor C-programmeurs: dit is te vergelijken met `static` variabelen in een functie.

Dit functieblok maakt gebruik van een complexe datastructuur, namelijk een array van structures. De array wordt gebruikt als circulaire buffer (of ringbuffer) om gegevens op te slaan. Met behulp van een *lus* worden de elementen van de array gemanipuleerd.

Voer de interface-parameters op de gebruikelijke wijze in. Speciale aandacht is nodig voor het invoeren van de array van structures. Maak een variabele `TEMP` aan onder het kopje

Static. Vul als datatype Array[1..10] of Struct in en druk op de enter-toets. Voor de variabele komt nu een klein pijltje te staan. Dit is te zien in figuur 6.13.

	Name	Data type	Offset	Default value	Visible in ...	Setpoint	Comment
1	Input						
2	RECORD_TEMP	Bool	...	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Flag to signal record of temperature
3	CURRENT_TEMP	Real	...	0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperature
4	Output						
5	AVERAGE_TEMP	Real	...	0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The average temperature
6	InOut						
7	<Add new>						
8	Static						
9	RECORD_TEMP_EDGE	Bool	...	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Edge flag for RECORD_TEMP
10	INDEX	Int	...	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The current index in the array
11	TEMP	Array[1..10] of Struct	...		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
12	<Add new>						
13	Temp						
14	<Add new>						
15	Constant						
16	<Add new>						

Figuur 6.13: Aanmaken van een array van structures.

Klik op het pijltje om de array te openen. TIA Portal laat nu tien elementen zien die nog voorzien moeten worden van variabelen. Onder de variabele TEMP[1] is het mogelijk om de variabelen in te voeren.

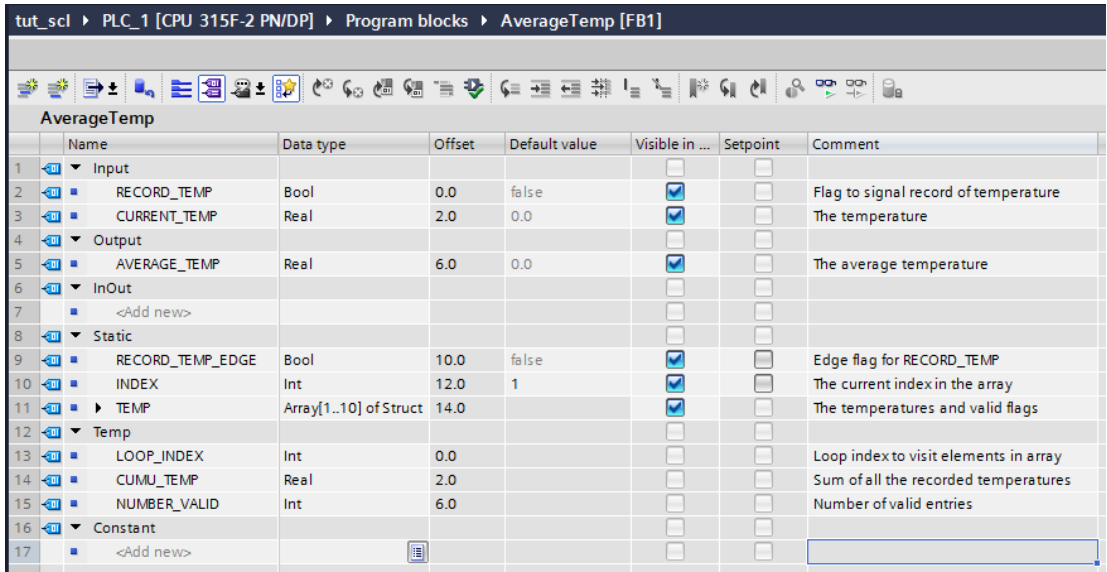
In figuur 6.14 is te zien hoe de structures ingevuld moeten worden. Er worden een Real en een Bool aangemaakt. Bij het invullen van de structure onder TEMP[1] worden de elementen van de structure naar de andere array-elementen gekopieerd. Die hoeven dus niet apart te worden ingevoerd.

8	Static						
9	RECORD_TEMP_EDGE	Bool	...	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Edge flag for RECORD_TEMP
10	INDEX	Int	...	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The current index in the array
11	TEMP	Array[1..10] of Struct	...		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
12	TEMP[1]	Struct	...		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
13	temperature	Real	...	0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
14	valid	Bool	...	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
15	<Add new>						
16	TEMP[2]	Struct	...		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
17	TEMP[3]	Struct	...		<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Figuur 6.14: Invullen van de structure.

Nu de array van structures is ingevuld kunnen de overige variabelen worden ingevuld. Dit is te zien in figuur 6.15. De array is weer “dichtgeklapt”.

Nu wordt de SCL-code ingevoerd, te zien in figuur 6.16. Een korte uitleg over de code is hier wel op zijn plaats. Er wordt gebruik gemaakt van diverse taalconstructies zoals beslissingen, lussen en rekenkundige operaties. Het eerste deel zorgt voor het opslaan van een nieuwe temperatuurwaarde in de array. Daarbij wordt aangegeven dat de opgeslagen waarde geldig is. Dit alles gebeurt onder het toezien van de variabele RECORD\_TEMP. Samen met de variabele RECORD\_TEMP\_EDGE vormen ze een *flankdetector*. Hierdoor wordt bij het actief worden van RECORD\_TEMP slechts eenmaal een nieuwe temperatuur toegevoegd. Om een nieuwe waarde toe te voegen moet RECORD\_TEMP eerst weer gedeactiveerd worden.



Figuur 6.15: De complete interface van functieblok AverageTemp (FB1).

```

IF... CASE... FOR... WHILE... (*...*) REGION
OF... TO DO... DO...

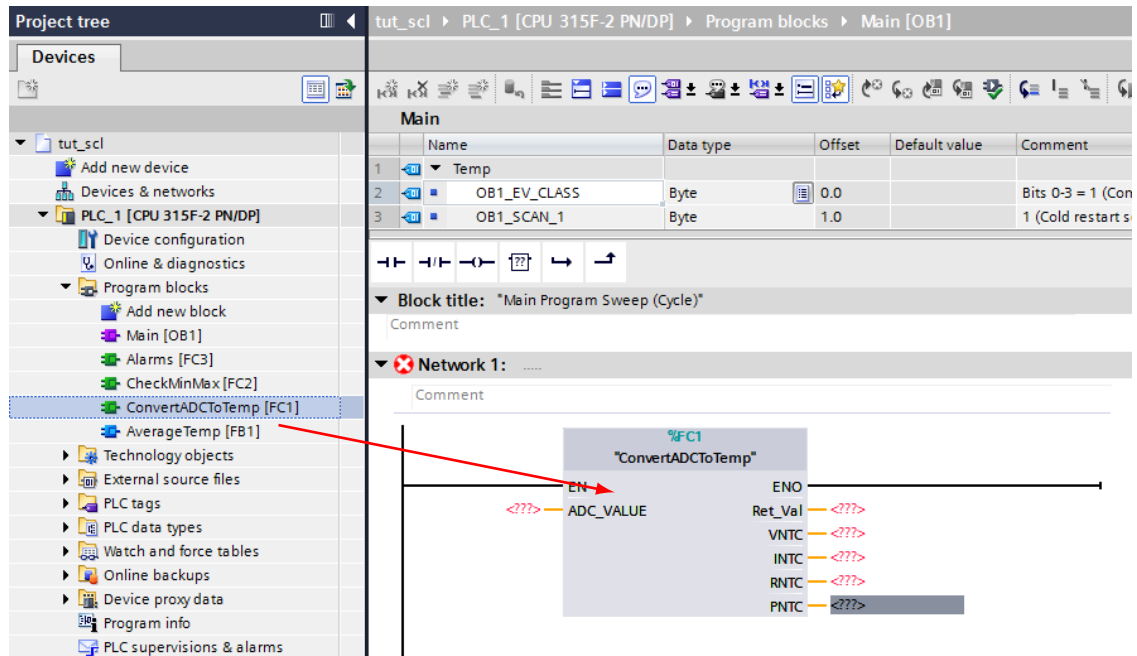
1 // This routine records temperatures in an array and calculates the average
2 // temperature over all valid entries. For recording a temperature, an edge
3 // is detected on input RECORD_TEMP. The entries are recorded in an array
4 // used as a circular buffer.
5
6 // Should we record once?
7 IF #RECORD_TEMP AND NOT #RECORD_TEMP_EDGE THEN
8     // Do record it
9     #TEMP[#INDEX].temperature := #CURRENT_TEMP;
10    #TEMP[#INDEX].valid := TRUE;
11    // Advance the index
12    #INDEX := #INDEX + 1;
13    // But if it is past the end, flip it to the begin
14    IF #INDEX > 10 THEN
15        #INDEX := 1;
16    END_IF;
17 END_IF;
18 // Update edge relay
19 #RECORD_TEMP_EDGE := #RECORD_TEMP;
20
21 // Make sure we start at 0.0
22 #CUMU_TEMP := 0.0;
23 // Count the number of valid entries
24 #NUMBER_VALID := 0;
25 // Visit all array items and sum them
26 FOR #LOOP_INDEX := 1 TO 10 BY 1 DO
27     IF #TEMP[#LOOP_INDEX].valid THEN
28         #CUMU_TEMP := #CUMU_TEMP + #TEMP[#LOOP_INDEX].temperature;
29         #NUMBER_VALID := #NUMBER_VALID + 1;
30     END_IF;
31 END_FOR;
32
33 // Calculate average, but note that if there are no valid entries,
34 // we would get and division by zero error. So give back an
35 // unrealistic average temperature
36 IF #NUMBER_VALID = 0 THEN
37     #AVERAGE_TEMP := -273.15;
38 ELSE
39     #AVERAGE_TEMP := #CUMU_TEMP / #NUMBER_VALID;
40 END_IF;
    
```

Figuur 6.16: De SCL-code van functieblok AverageTemp (FB1).

Het tweede deel zorgt voor het berekenen van de gemiddelde temperatuur. Hierbij worden alle geregistreerde temperatuurwaarden die geldig zijn gesommeerd en gedeeld door het aantal geldige waarden. De gemiddelde temperatuur wordt via een output-parameter teruggegeven. De complete code, inclusief de interface, is te vinden in bijlage E.

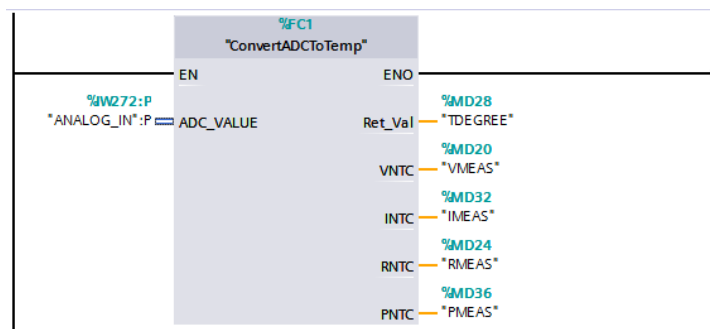
## 6.9 OB1 invoeren

De functies en het functieblok zijn nu ingevoerd. Nu moeten de aanroepen in OB1 geprogrammeerd worden. De aanroepen zijn gemakkelijk te programmeren. Open OB1 door erop te dubbelklikken. Als eerste wordt de functie ConvertADCToTemp aangeroepen. Sleep vanuit de project tree de functie naar de eerste rung van OB1. Zie figuur 6.17.



Figuur 6.17: Functie ConvertADCToTemp toevoegen aan eerste rung OB1.

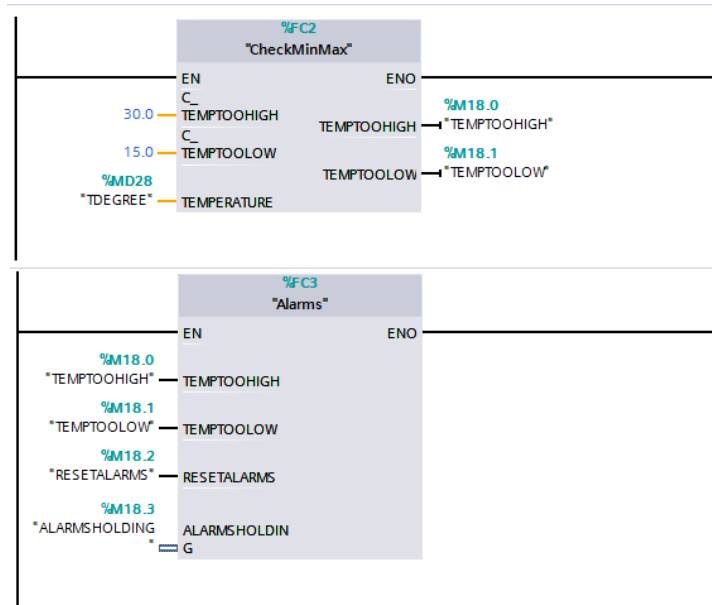
Nu moeten alle argumenten nog ingevuld worden. Aan de ingang van het blok wordt de variabele ANALOG\_IN geplaatst. Merk op dat hier gebruik wordt gemaakt van zogenoemde *peripheral access* door achter de variabelenaam de qualifier :P te plaatsen (zie paragraaf 8.6). Aan de teruggaveparameter en de uitgangen worden achtereenvolgens de argumenten TDEGREE, VMEAS, IMEAS, RMEAS en PMEAS geplaatst. Zie figuur 6.18.



Figuur 6.18: Functie ConvertADCToTemp geplaatst in eerste rung van OB1.

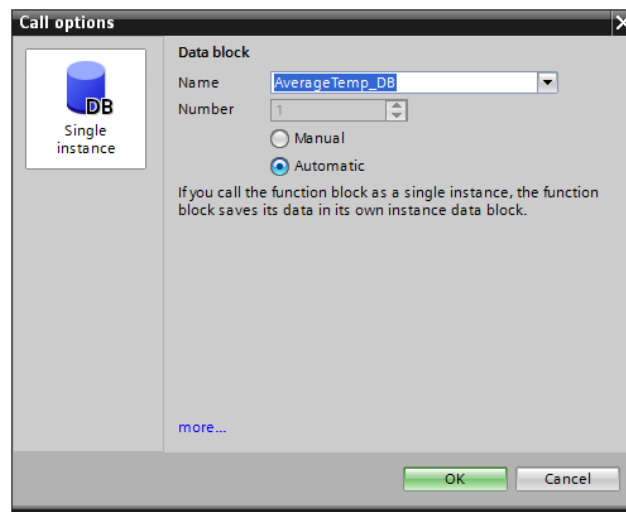
De eerste rung is nu correct ingevoerd. Voer op vergelijkbare wijze de functies CheckMinMax en Alarms in. Dit is te zien in figuur 6.19.

Als laatste moet functieblok AverageTemp worden ingevoegd. Sleep het functieblok naar de vierde rung van OB1. Merk op dat aan een functieblok een Instance Data Block is gekop-



**Figuur 6.19:** Functies *CheckMinMax* en *Alarms* geplaatst in OB1.

peld. TIA Portal zal eerst vragen om een Instance Data Block aan te maken. De standaard naam is de naam van het functieblok aangevuld met `_DB`. Dit is te zien in figuur 6.20. Klik op **OK** om het datablok aan te maken.



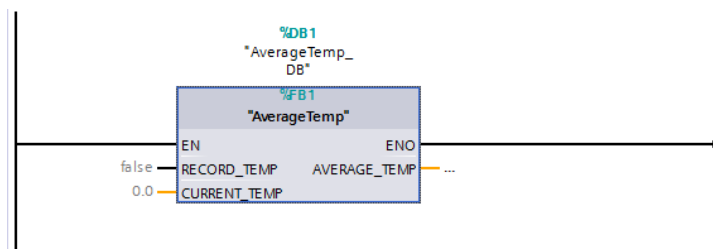
**Figuur 6.20:** Aanmaken datablok voor functieblok *AverageTemp*.

Het datablok is nu aangemaakt en verschijnt in de project tree onder het kopje Program blocks. Tevens wordt in de vierde rung het blok *AverareTemp* zichtbaar. Hierbij zijn de argumenten voor de variabelen `RECORD_TEMP` en `CURRENT_TEMP` al ingevuld. Dat heeft te maken met de wijze waarop parameteroverdracht plaatsvindt met een functieblok. Alle parameters zijn in het datablok geplaatst en hebben een standaard beginwaarde.

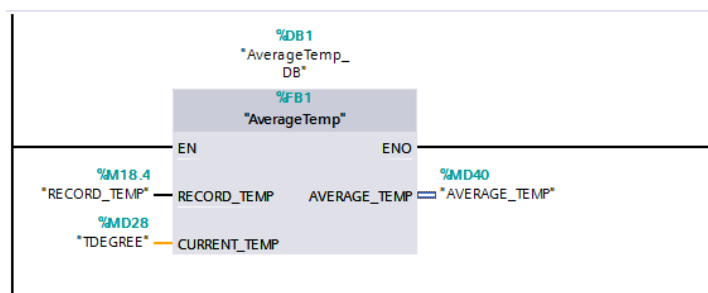
De argumenten moeten worden ingevuld zoals te zien is in figuur 6.22.

Alle rungs zijn nu ingevoerd. Nu kan OB1 gecompileerd worden. Selecteer onder het kopje Program blocks blok OB1 en klik op de rechter muisknop. Selecteer in het contextmenu `Compile`→`Software (only changes)`. OB1 wordt nu gecompileerd.





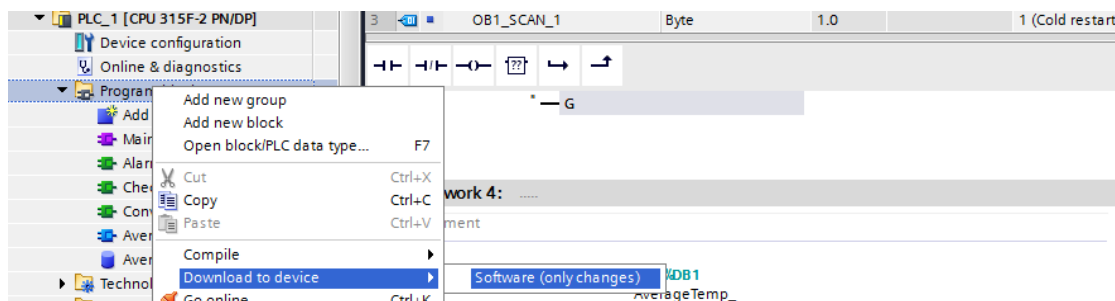
Figuur 6.21: Functieblok AverageTemp is in de vierde rung geplaatst.



Figuur 6.22: De vierde rung compleet.

## 6.10 Laden PLC met programma

Het laden van het programma gaat zoals is beschreven in hoofdstuk 4. Zorg ervoor dat de PLC in STOP staat. Selecteer onder PLC\_1 het menu-onderdeel Program Blocks. Klik op de rechter muisknop en selecteer Download to device→Software (only changes). Zie figuur 6.23.



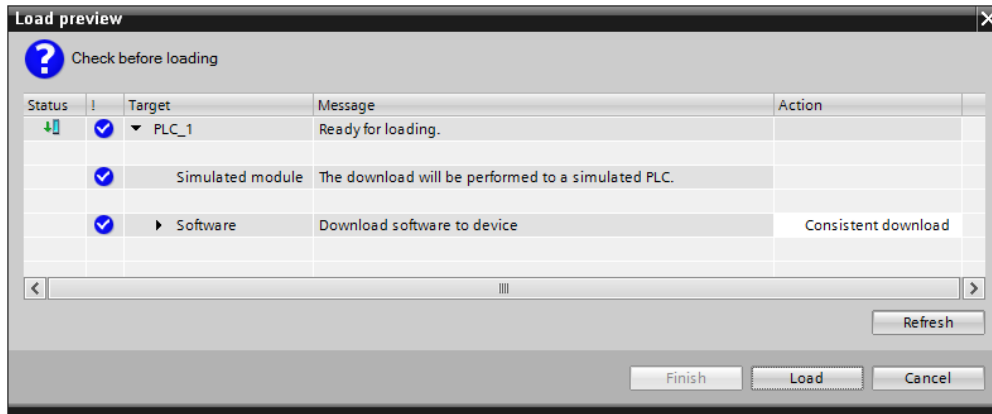
Figuur 6.23: Downloaden van de blokken naar de PLC.

TIA Portal komt nu met een dialoog waarin wordt gevraagd of de blokken in de PLC moeten worden geladen. Zie figuur 6.24. Klik op **Load** om de blokken te laden.

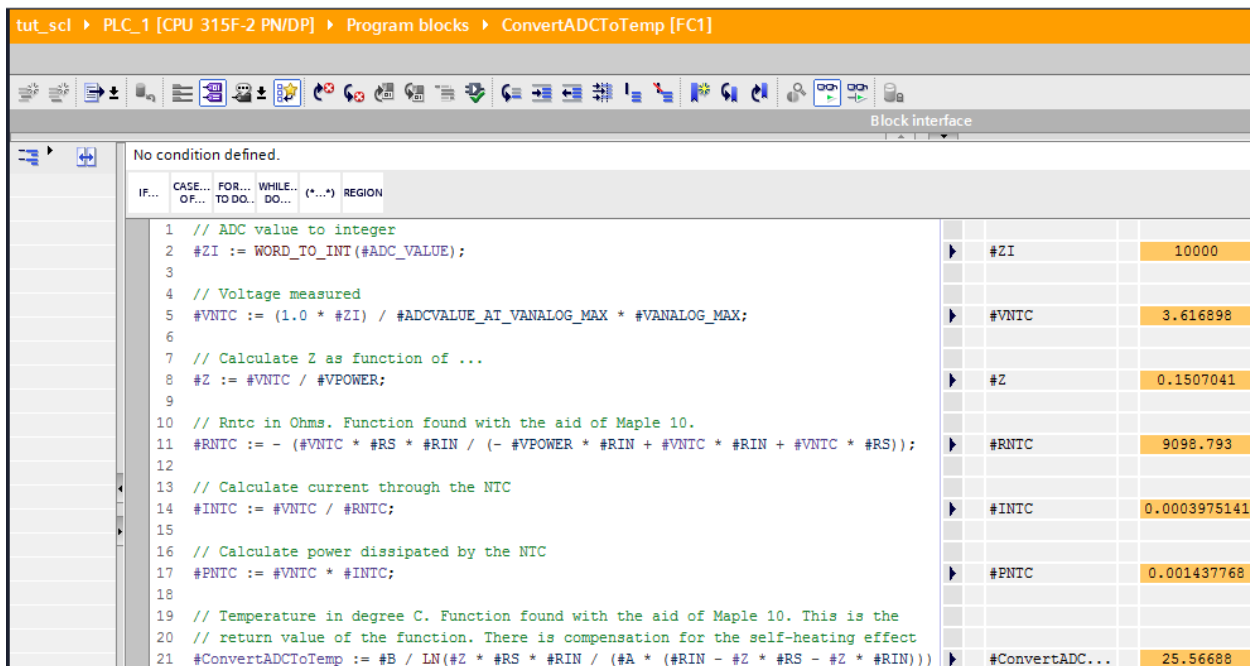
De blokken zijn nu geladen en de PLC kan weer in RUN gezet worden.

## 6.11 Monitoren van programma en variabelen

De SCL-code kan gemonitord worden, d.w.z. gevolgd worden. De SCL-editor bevat hiervoor een speciale modus. Open de editor door op ConvertADCToTEMP te dubbelklikken. Aan de bovenrand is een pictogram met een brilletje te vinden. Klik hier één keer op. De editor zal nu communiceren met de PLC en de waarden van de variabelen weergeven. Zie figuur 6.25. Merk op dat constanten en tijdelijke variabelen niet gemonitord kunnen worden.



Figuur 6.24: Download preview.



Figuur 6.25: Monitoren van functie ConvertADCToTemp.

Ook de PLC tags kunnen gemonitord worden. De PLC tags omvatten de ingangen, uitgangen en het merkergeheugen. Open de PLC tags-tabel in de project tree onder het kopje PLC tags op Default tag table te dubbelklikken. De PLC tags-tabel wordt nu geopend. Klik daarna linksboven op het pictogram met het brilletje. In de tabel worden nu de actuele waarden van de PLC tags weergegeven. Zie figuur 6.26.

Het is ook mogelijk om de variabelen in een datablok te monitoren. Open het datablok AverageTemp\_DB door in de project tree onder het kopje Program blocks op de naam van het datablok te dubbelklikken. Het datablok wordt nu geopend. Klik daarna linksboven op het pictogram met het brilletje. In de tabel worden de actuele waarden van de variabelen weergegeven. Zie figuur 6.27.

tut\_scl ▶ PLC\_1 [CPU 315F-2 PN/DP] ▶ PLC tags ▶ Default tag table [12]

Default tag table

	Name	Data type	Address	Retain	Acces...	Visibl...	Monitor value	Comment
1	ANALOG_IN	Word	%IW272		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	16#1F40	The raw ADC value
2	VMEAS	Real	%MD20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2.893518	Measured voltage across the NTC
3	RMEAS	Real	%MD24		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	6887.052	Calculated resistance of the NTC
4	TDEGREE	Real	%MD28		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	32.22163	Calculated temperature
5	IMEAS	Real	%MD32		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0.0004201389	Calculated current through the NTC
6	PMEAS	Real	%MD36		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0.00121568	Calculated power dissipated by the NTC
7	TEMPTOOHIGH	Bool	%M18.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> TRUE	True = Temperature too high (output)
8	TEMPTOLOW	Bool	%M18.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> FALSE	True = Temperature too low (output)
9	RESETALARMS	Bool	%M18.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> FALSE	True = Alarms will reset (input)
10	ALARMISHOLDING	Bool	%M18.3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> TRUE	Holding relay for alarms
11	RECORD_TEMP	Bool	%M18.4		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> FALSE	Should we record a temperature?
12	AVERAGE_TEMP	Real	%MD40		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	23.37466	Average temperature
13	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

Figuur 6.26: Monitoren van de PLC tags.

tut\_scl ▶ PLC\_1 [CPU 315F-2 PN/DP] ▶ Program blocks ▶ AverageTemp\_DB [DB1]

AverageTemp\_DB

	Name	Data type	Offset	Start value	Monitor value	Retain	Visible in ...	Setpoint	Comment
1	Input					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2	RECORD_TEMP	Bool	0.0	false	FALSE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Flag to signal reco
3	CURRENT_TEMP	Real	2.0	0.0	32.22163	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperature
4	Output					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5	AVERAGE_TEMP	Real	6.0	0.0	32.22163	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The average temp
6	InOut					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7	Static					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
8	RECORD_TEMP_EDGE	Bool	10.0	false	FALSE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Edge flag for RECO
9	INDEX	Int	12.0	1	3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The current indexi
10	TEMP	Array[1..10] of Struct	14.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
11	TEMP[1]	Struct	14.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
12	temperature	Real	14.0	0.0	32.22163	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
13	valid	Bool	18.0	false	TRUE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
14	TEMP[2]	Struct	20.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
15	temperature	Real	20.0	0.0	32.22163	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
16	valid	Bool	24.0	false	TRUE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
17	TEMP[3]	Struct	26.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
18	temperature	Real	26.0	0.0	0.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
19	valid	Bool	30.0	false	FALSE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
20	TEMP[4]	Struct	32.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
21	TEMP[5]	Struct	38.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
22	TEMP[6]	Struct	44.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
23	TEMP[7]	Struct	50.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
24	TEMP[8]	Struct	56.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
25	TEMP[9]	Struct	62.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures
26	TEMP[10]	Struct	68.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The temperatures

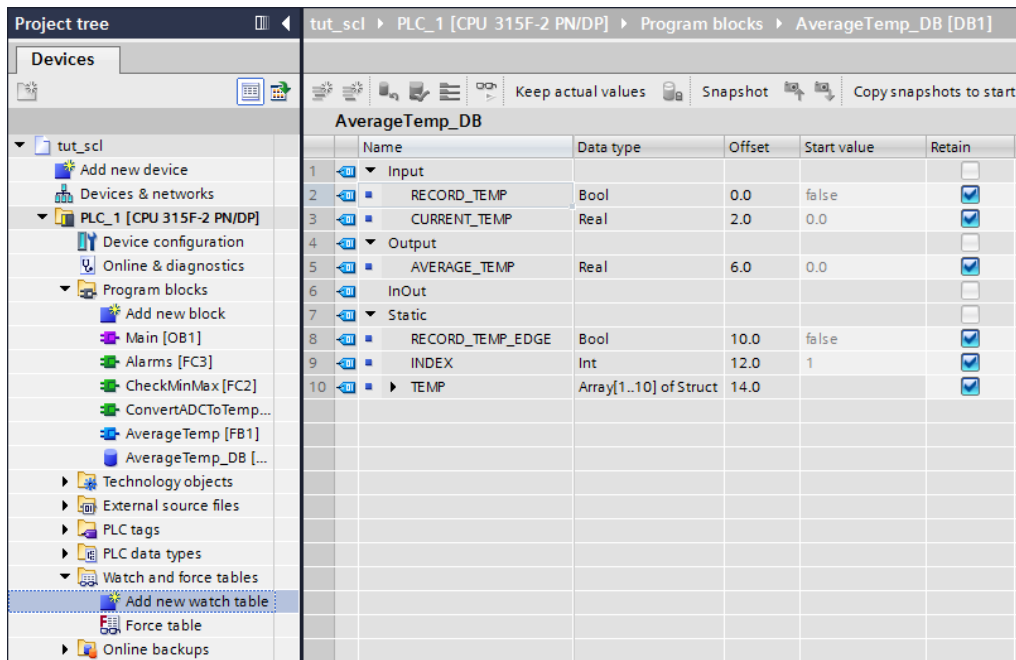
Figuur 6.27: Monitoren van datablok AverageTemp\_DB.

## 6.12 Variabelen manipuleren met watch table

In de vorige paragraaf is uitgelegd hoe de variabelen gemonitord kunnen worden. De variabelen zijn op deze manier echter niet aan te passen. Dat kan wel met een *watch table*. Met een watch table kunnen variabelen selectief gemonitord en gemanipuleerd worden.

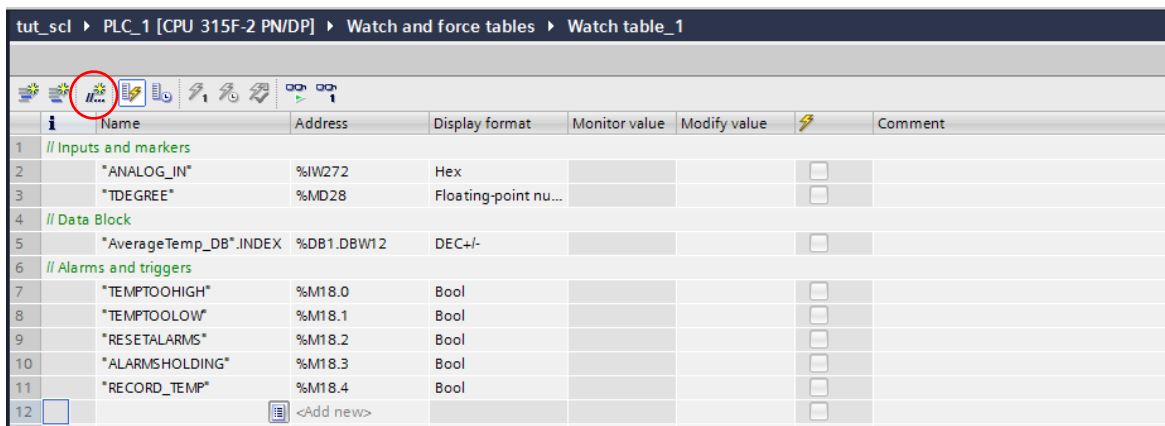
Voeg een nieuwe watch table toe door onder het kopje Watch and force tables te dubbelklikken op Add new watch table. Zie figuur 6.28.

Er wordt een nieuwe watch table aangemaakt met de standaard naam Watch table\_1.



Figuur 6.28: Toevoegen nieuwe watch table.

Vul de tabel in volgens figuur 6.29. Invullen kan op twee manieren: via de naam van de variabelen of via het adres. Een commentaarregel kan ingevoegd worden door op het pictogram met de dubbele slash te klikken. Zie de rode cirkel.



Figuur 6.29: De ingevulde watch table.

De watch table geeft de actuele waarden van de variabelen weer na een klik op het pictogram met het brilletje en het groene driehoekje dat op het “play”-symbool lijkt.

Het wijzigen van een variabele gaat eenvoudig. Vul bij de variabele die gewijzigd moet worden in de kolom Modify value de nieuwe waarde in. Let hierbij goed op het datatype. Bij een Bool kunnen alleen de waarden TRUE en FALSE worden ingevuld.

In figuur 6.31 is te zien dat variabele RESETALARMS op TRUE gezet wordt. Klik daarna op het pictogram met de bliksemschicht met een 1 (zie de rode cirkel). De waarde van RESETALARMS wordt dan op TRUE gezet. De variabele ALARMSHOLDING wordt als resultaat op FALSE gezet.

	Name	Address	Display format	Monitor value	Modify value	Comment
1	// Inputs and markers					
2	*ANALOG_IN*	%IW272	Hex	16#1B58	<input type="checkbox"/>	
3	*TDEGREE*	%MD28	Floating-point nu...	36.17617	<input type="checkbox"/>	
4	// Data Block					
5	*AverageTemp_DB*.INDEX	%DB1.DBW12	DEC+/-	6	<input type="checkbox"/>	
6	// Alarms and triggers					
7	*TEMPTOOHIGH*	%M18.0	Bool	<input checked="" type="checkbox"/> TRUE	<input type="checkbox"/>	
8	*TEMPTOOLOW*	%M18.1	Bool	<input type="checkbox"/> FALSE	<input type="checkbox"/>	
9	*RESETALARMS*	%M18.2	Bool	<input type="checkbox"/> FALSE	<input type="checkbox"/>	
10	*ALARMSHOLDING*	%M18.3	Bool	<input checked="" type="checkbox"/> TRUE	<input type="checkbox"/>	
11	*RECORD_TEMP*	%M18.4	Bool	<input type="checkbox"/> FALSE	<input type="checkbox"/>	
12	<Add new>					

Figuur 6.30: Watch table monitort de variabelen.

	Name	Address	Display format	Monitor value	Modify value	Comment
1	// Inputs and markers					
2	*ANALOG_IN*	%IW272	Hex	16#2710	<input type="checkbox"/>	
3	*TDEGREE*	%MD28	Floating-point nu...	25.56688	<input type="checkbox"/>	
4	// Data Block					
5	*AverageTemp_DB*.INDEX	%DB1.DBW12	DEC+/-	6	<input type="checkbox"/>	
6	// Alarms and triggers					
7	*TEMPTOOHIGH*	%M18.0	Bool	<input type="checkbox"/> FALSE	<input type="checkbox"/>	
8	*TEMPTOOLOW*	%M18.1	Bool	<input type="checkbox"/> FALSE	<input type="checkbox"/>	
9	*RESETALARMS*	%M18.2	Bool	<input checked="" type="checkbox"/> TRUE	TRUE <input checked="" type="checkbox"/>	
10	*ALARMSHOLDING*	%M18.3	Bool	<input type="checkbox"/> FALSE	<input type="checkbox"/>	
11	*RECORD_TEMP*	%M18.4	Bool	<input type="checkbox"/> FALSE	<input type="checkbox"/>	
12	<Add new>					

Figuur 6.31: Wijzigen van de waarde van een variabele.

Opmerking: bitvariabelen kunnen ook direct op TRUE of FALSE gezet worden. Druk op de sneltoetscombinatie **Ctrl+F2** om een bitvariabele op TRUE te zetten en druk op de sneltoetscombinatie **Ctrl+F3** om een bitvariabele op FALSE te zetten.

### 6.13 Automatische temperatuurregistratie

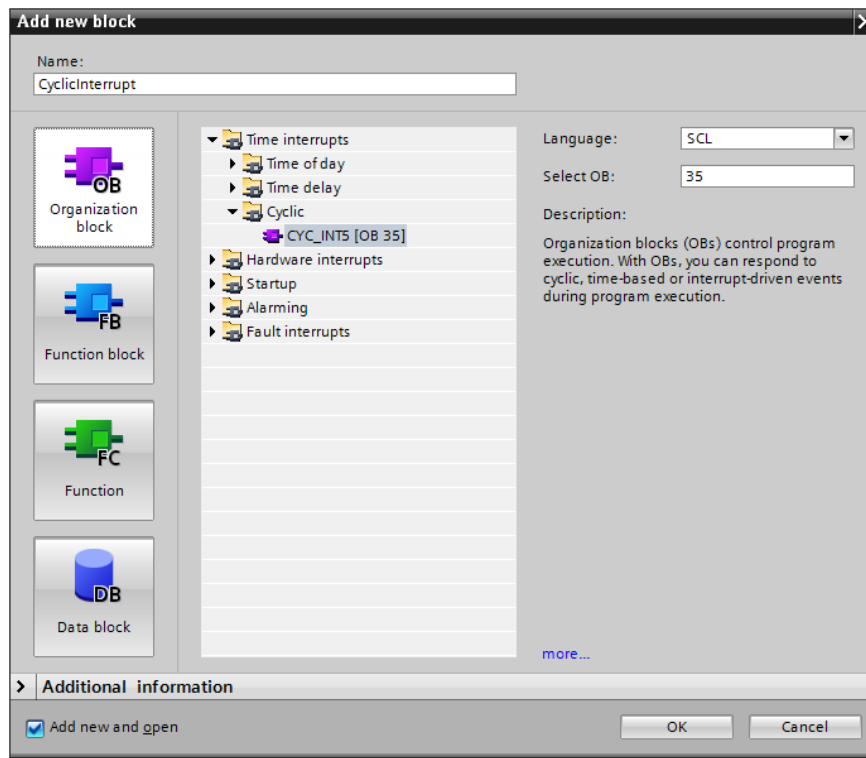
Het registreren van een temperatuur in één van de TEMP-variabelen in datablok DB1 moet met de hand gebeuren. Hiervoor moet variabele RECORD\_TEMP even op TRUE gezet worden en daarna weer op FALSE. De registratie is echter te automatiseren om bijvoorbeeld elke minuut een nieuwe waarde te registreren. Dat kan opgelost worden middels een listige timer-constructie die zichzelf opnieuw start. Maar het kan ook anders namelijk door gebruik te maken van een *cyclische interrupt*<sup>6</sup>. De programmatuur van de interruptroutine wordt geplaatst in OB35. Deze bouwsteen wordt, als het in de PLC bestaat, om de 100 milliseconden aangeroepen (deze tijd is instelbaar). Door nu het aantal interrupts te tellen met behulp van een variabele kan een bepaalde tijd worden afgemeten. Stel dat eens in de minuut een nieuwe registratie van de temperatuur moet plaatsvinden dan moet van 0 t/m 599 geteld worden om deze minuut af te tellen. De variabele RECORD\_TEMP wordt dan even op TRUE gezet zodat de registratie kan plaatsvinden. De scan cycle-tijd is in de regel korter dan 100 milliseconden. RECORD\_TEMP blijft zodoende meerdere scan cycles

<sup>6</sup> Een interrupt is een onderbreking van het lopende programma.

TRUE. Door de flankdetectie in functieblok AverageTemp is dat niet erg.

Maak om te beginnen een nieuwe tag aan in de tag-lijst. De naam van de nieuwe variabele is CYCLIC\_INTERRUPT\_COUNTER, het type is Int en de opslagruimte is MW44.

Maak vervolgens een nieuw blok aan door onder het kopje Program blocks op Add new block te dubbelklikken. Klik op het pictogram Organization Block en selecteer OB35. De naam van OB35 moet veranderd worden in CyclicInterrupt. De taal moet ingesteld worden op SCL. Klik daarna op **OK**. Zie figuur 6.32. De volledige code, inclusief de interface, is opgenomen in bijlage F.



Figuur 6.32: Toevoegen cyclische interrupt OB35.

OB35 wordt nu aangemaakt en is te vinden onder het kopje Program blocks.

De code van OB35 is te vinden in figuur 6.33. Zoals bekend heeft een OB geen ingangs- en uitgangsparameters, alleen maar tijdelijke variabelen die door het operating system van de PLC worden aangemaakt. Bewerkingen kunnen alleen maar gedaan worden op variabelen die in de PLC tags-lijst of datablokken voorkomen.

De code is eenvoudig van opbouw. Er wordt getest of de teller op de waarde 599 staat. Als dat het geval is wordt de RECORD\_TEMP op TRUE gezet en de teller op 0. Zo niet, dat wordt RECORD\_TEMP op FALSE gezet en de teller wordt met één verhoogd. Zodoende wordt eens in de 600 keer een puls gegeven op RECORD\_TEMP.

## 6.14 Foutdetectie NTC-weerstand

Bij normaal gebruik zal de NTC-weerstand een waarde hebben binnen het gespecificeerde bereik. Maar er zijn twee mogelijke bronnen van fouten. De eerste fout betreft kortsluiting van de NTC-weerstand. De spanning over de NTC-weerstand is dan 0 V. De ADC zal een

```

1 // Check if counter is expired (1 minute)
2 IF "CYCLIC_INTERRUPT_COUNTER" = 599 THEN
3   // If so, set RECORD_TEMP and start counting from 0
4   "RECORD_TEMP" := TRUE;
5   "CYCLIC_INTERRUPT_COUNTER" := 0;
6 ELSE
7   // If not, reset RECORD_TEMP and update counter
8   "RECORD_TEMP" := FALSE;
9   "CYCLIC_INTERRUPT_COUNTER" := "CYCLIC_INTERRUPT_COUNTER" + 1;
10 END_IF;
11

```

IF...	CASE... OF...	FOR... TO DO...	WHILE... DO...	(*...*)	REGION
					"CYCLIC_INTERRUPT_COUNTER" %MW44
					"RECORD_TEMP" %M18.4
					"CYCLIC_INTERRUPT_COUNTER" %MW44
					"RECORD_TEMP" %M18.4
					"CYCLIC_INTERRUPT_COUNTER" %MW44

Figuur 6.33: SCL-code OB35.

waarde van 0 teruggeven. Diverse berekeningen in functie ConvertADCToTemp leveren dan onbruikbare waarden op omdat er gedeeld wordt door 0. Dit is eenvoudig af te vangen door in functie ConvertADCToTemp te testen of de ADC-waarde 0 is. Wiskundig gezien is een deling door 0 onbepaald. Maar in de specificatie van *floating point* getallen wordt het resultaat plus of min oneindig. De tweede fout betreft het ontbreken of defect zijn van de NTC-weerstand. Vanwege de spanningsdeling van de serieweerstand en de ingangsweerstand van de ADC-ingang zal de spanning boven het “overflow”-bereik uitkomen (maar nog wel onder de maximale spanning die aan de ingang mag worden aangeboden). Dit is eenvoudig af te vangen door in de functie ConvertADCToTemp te testen of de ADC-waarde 32767 is. Deze aanpassingen moeten gedaan worden in functie ConvertADCToTemp. Tevens kan hier dan ook een Output-parameter worden ingebouwd die aangeeft of de temperatuur een geldige waarde heeft. In de functie Alarms kan deze waarde meegenomen worden bij het instellen van de holding alarm.

De lezer wordt uitgedaagd deze aanpassingen in de diverse functies door te voeren.

## 7. SEQUENCERS IN SCL

Een *sequencer* is een programma dat een aantal stappen (steps) doorloopt. Op enig moment is een sequencer in een bepaalde stap. Het is mogelijk om van de ene stap naar een andere stap te gaan aan de hand van *overgangscondities* (transitions). De overgangscondities worden gerealiseerd door de ingangswaarden van de sequencer. Bij elke stap worden *acties* (actions) ondernomen. Hierbij worden uitgangen (of merkers, timers, counters) op een bepaalde waarde gezet. Het is mogelijk om de ingangswaarden mee te nemen in de actie-conditie. Het geheel is te vergelijken met een toestandsmachine in de digitale techniek. De ISA-88-norm spreekt echter over Sequential Function Charts (SFC).

### 7.1 Opzet sequencer

In listing 7.1 is de opzet van een sequencer te zien. De sequencer is uitgerust met een

```

1 // On a reset, we do ...
2 IF _reset_is_active_possibly_with_edge_detect_ THEN
3     STEP := _reset_step_;
4     output := _value_of_output_;
5     ...
6 ELSE
7     // Evaluate which step we're in...
8     CASE STEP OF
9         _reset_step_: // Do something in the reset step
10            ...
11        _some_step_1_: // Do something...
12            ...
13        _some_step_2_: // example of a step evaluation
14            IF input = _some_value_ THEN
15                STEP := _some_new_step_;
16                output := _value_of_output_;
17                ...
18            ELSE
19                STEP := _some_new_step_;
20                output := _value_of_output_;
21                ...
22            END_IF;
23        _some_step_3_: ...
24    ELSE // If something's horribly wrong, ...
25        STEP := _reset_step_;
26        output := _value_of_output_;
27    END_CASE;
28 END_IF;

```

Listing 7.1: Opzet sequencer in SCL.



overheersende reset. Dat houdt in dat als de reset wordt geactiveerd, de sequencer de code direct onder het IF-statement in regel 1 uitvoert. Vaak wordt gebruikgemaakt van *flankdetectie* anders blijft de sequencer hangen in het stukje code zolang de reset actief is. Flankdetectie wordt besproken in paragraaf 8.11. Er is niet altijd een resetfaciliteit nodig, maar een noodstop wordt wel vaak ingebouwd. De noodstop is op eenzelfde wijze te implementeren als de reset.

Als de reset niet actief is, moet de sequencer zijn gewone stappenplan volgen. Dat stappenplan wordt geëvalueerd in de statements CASE ... OF tot en met END\_CASE. Op enig moment is slechts één van de stappen actief. In dit voorbeeld is de stap `_some_step_2_` nader uitgewerkt. Te zien is dat onder besturing van de ingang `input` de uitgang `output` een waarde krijgt toegewezen evenals de variabele `STEP` waarin de huidige stap wordt bijgehouden. Na het doorlopen van de code is de stap dus veranderd en bij een volgende evaluatie van de code is een andere stap actief (dat hoeft niet zoals later te zien is).

Binnen het CASE-statement is nog een ELSE-statement te zien. Deze “stap” wordt uitgevoerd als de stapvariabele een onbekende waarde bevat. Bij normaal gebruik van de sequencer zal dit nooit voorkomen maar er kunnen natuurlijk programmeerfouten gemaakt worden.

## 7.2 Voorbeeld van een sequencer

In deze paragraaf wordt een voorbeeld van een sequencer behandeld. De eerste stap is het aanmaken van de stappen van de sequencer, dat zijn de stapnamen die gebruikt worden in het CASE-statement.

Het CASE-statement kan alleen variabelen van het type `Int` afhandelen. Dat betekent dus dat de stappen als gehele getallen moeten worden opgegeven. SCL ondersteunt geen enumeratietypes (althans niet in TIA Portal V14) zodat we zijn aangewezen op het gebruik van constanten. Een voorbeeld van het gebruik van constanten is te zien in figuur 7.1. Deze worden aangemaakt in de interface van het gebruikte blok.

Line	Symbol	Value	Type	Initial Value
9	Constant			
10	RESET_ST		Int	0
11	STEP1_ST		Int	1
12	STEP2_ST		Int	2

Figuur 7.1: Declaratie van de stappen.

De stapnummers zijn hier opvolgend vanaf 0 toegewezen maar dat hoeft natuurlijk niet. Let erop dat alle constanten een unieke waarde moeten krijgen. Er mogen dus geen dubbelingen in voorkomen.

De code van de sequencer is te zien in listing 7.2. Naast de constanten die de stappen vertegenwoordigen wordt er gebruikgemaakt van globale variabelen, variabelen die voorkomen in de PLC tags-lijst.

De sequencer begint met een overheersende reset. Zolang de reset actief is wordt deze code uitgevoerd. Nadat de reset gedeactiveerd is wordt naar stap `#RESET_ST` gegaan. Dat gebeurt pas in de volgende scan cycle. In deze stap is te zien dat de uitgangen `output1` en `output2` afhankelijk zijn van de ingangen `input1` en `input2`. Merk op dat de statements IF ... ELSIF worden afgesloten met een ELSE. Als geen van de beschreven condities waar

```

1 // If reset is active...
2 IF "reset" = TRUE THEN
3     "STEP" := #RESET_ST;
4     "output1" := FALSE;
5     "output2" := FALSE;
6 ELSE
7     // Evaluate each step
8     CASE "STEP" OF
9         #RESET_ST:
10            // Output depends on inputs and step
11            IF "input1" = TRUE AND "input2" = FALSE THEN
12                "output1" := TRUE;
13                "output2" := FALSE;
14            ELSIF "input1" = TRUE AND "input2" = TRUE THEN
15                "output1" := FALSE;
16                "output2" := TRUE;
17            ELSE
18                "output1" := TRUE;
19                "output2" := TRUE;
20            END_IF;
21            // Always goto next step
22            "STEP" := #STEP1_ST;
23        #STEP1_ST:
24            // Outputs and new step depend on inputs and step
25            IF "input1" = TRUE THEN
26                "STEP" := #STEP2_ST;
27                "output1" := TRUE;
28                "output2" := FALSE;
29            ELSE
30                "STEP" := #STEP1_ST;
31                "output1" := FALSE;
32                "output2" := TRUE;
33            END_IF;
34        #STEP2_ST:
35            // New step depends on inputs
36            IF "input1" = FALSE AND "input2" = TRUE THEN
37                "STEP" := #STEP1_ST;
38            ELSE
39                "STEP" := #STEP2_ST;
40            END_IF;
41            // Outputs only depend on step
42            "output1" := TRUE;
43            "output2" := FALSE;
44        ELSE // if something's wrong...
45            "STEP" := #RESET_ST;
46            "output1" := FALSE;
47            "output2" := FALSE;
48    END_CASE;
49 END_IF;

```

Listing 7.2: Code van de functie Sequencer.

is zal deze code worden uitgevoerd. Ook is te zien dat de nieuwe stap wordt toegekend onafhankelijk van de ingangen. De sequencer blijft hierdoor slechts één scan cycle in stap #RESET\_ST. In stap #STEP1\_ST zijn zowel de nieuwe stap als de uitgangen afhankelijk van ingang input1 maar niet afhankelijk van input2. Als input1 TRUE is wordt de nieuwe stap #STEP\_2 toegekend. Als dat niet het geval is wordt de nieuwe stap #STEP\_1. De sequencer blijft dan in de huidige stap. Op deze manier is het mogelijk om te blijven “hangen” in een stap totdat aan een bepaalde conditie wordt voldaan. In stap #STEP\_2 is alleen de nieuwe stap afhankelijk van de ingangen. Ook hier is het mogelijk om te blijven hangen in de stap. De uitgangen zijn niet afhankelijk van de ingangen (maar wel van de stap natuurlijk).

Het is niet altijd noodzakelijk om een nieuwe stapwaarde toe te kennen aan de stapvariabele. Als de sequencer onder een bepaalde conditie in de huidige stap moet blijven hoeft de stap niet expliciet te worden toegekend. In listing 7.3 zijn twee stukjes code te zien die een identieke werking hebben. In de code links wordt in regel 6 de huidige stapwaarde expliciet toegekend aan de variabele STEP. In de code rechts is die toekenning weggelaten.

<pre> 1      ... 2      #STEP2_ST: 3          IF "input1" = FALSE THEN 4              "STEP" := #STEP1_ST; 5          ELSE 6              "STEP" := #STEP2_ST; 7          END_IF; 8          "output1" := TRUE; 9          "output2" := FALSE; 10         ELSE // if something's wrong... 11         ... </pre>	<pre> 1      ... 2      #STEP2_ST: 3          IF "input1" = FALSE THEN 4              "STEP" := #STEP1_ST; 5          END_IF; 6          "output1" := TRUE; 7          "output2" := FALSE; 8          ELSE // if something's wrong... 9          ... </pre>
---	---

**Listing 7.3:** Voorbeeld van het weglaten van de toekenning van een nieuwe stapwaarde.

### 7.3 Eerste binnenkomst en laatste verlaten van een stap

Soms is het nodig om bij de eerste binnenkomst van een stap een actie uit te voeren. Te denken valt aan het verhogen van een teller of het starten van een timer. Evenzo is goed mogelijk om tijdens de laatste keer verlaten van een stap iets te ondernemen. Beide kunnen eenvoudig gerealiseerd worden door het gebruik van een extra stapvariabele. We hebben het hier trouwens over de eerste binnenkomst vanuit een andere stap. Bij het laatste verlaten spreken we over de laatste keer voordat naar een andere stap wordt gegaan.

In het codevoorbeeld in listing 7.4 is gebruikgemaakt van een tweede stapvariabele OLDSTEP. Zoals de naam suggereert bevat deze variabele de oude waarde van de stap. Nu is dat niet helemaal waar, we kunnen beter spreken van een variabele die het mogelijk maakt om, in samenwerking met de variabele STEP, een verandering van de stapwaarde te detecteren.

Stel dat de sequencer uit een stap komt en de nieuwe stapwaarde `_some_step_` is. Er is voor

```

1      ...
2      _some_step_: // example of a step evaluation
3          IF OLDSTEP <> STEP THEN
4              // Code A
5              OLDSTEP := STEP;
6              ...
7          END_IF;
8
9          // Code B
10         IF _some_condition_is_true_ THEN
11             STEP := _some_new_step_;
12         END_IF;
13
14         IF OLD_STEP <> STEP THEN
15             // Code C
16             ...
17         END_IF;
18         ...

```

**Listing 7.4:** Opzet sequencer met detectie binnenkomst en verlaten.

gezorgd dat STEP en OLDSTEP niet aan elkaar gelijk zijn (we zien zo waarom). Bij de eerste binnenkomst vanuit een andere stap levert het IF-statement in regel 3 TRUE op zodat de code in regel 4 t/m 6 wordt uitgevoerd (Code A). In dit stukje code wordt de stapwaarde STEP gekopieerd naar OLDSTEP om ervoor te zorgen dat bij de volgende scan cycle deze code niet wordt uitgevoerd. Vervolgens wordt Code B uitgevoerd. Verder nemen we voor nu even aan dat de stapwaarde niet wijzigt. Dan is STEP gelijk aan OLDSTEP en wordt de code bij Code C niet uitgevoerd. Bij de volgende keer dat deze stap wordt geëvalueerd zal alleen de code bij Code B worden uitgevoerd omdat STEP en OLDSTEP aan elkaar gelijk zijn. Als nu in Code B een nieuwe stapwaarde wordt toegekend (regel 11), dan zal de code bij Code C ook worden uitgevoerd, want STEP en OLDSTEP zijn nu ongelijk aan elkaar.

In listing 7.5 is het gebruik van de eerste binnenkomst en laatste vertrek te zien. Er zijn zes tellers aangemaakt die bijhouden hoe vaak een stap wordt binnengegaan en verlaten. De tellers worden allemaal op 0 gezet onder besturing van de reset. Zodra de reset inactief wordt, worden de stappen geëvalueerd. Bij de eerste binnenkomst of laatste keer verlaten van een stap wordt de desbetreffende teller met één verhoogd.

Let ook op de toekenning van STEP en OLDSTEP binnen de reset. De volgende stap na de reset is #RESET\_ST. Deze waarde wordt toegekend aan STEP maar mag niet aan OLDSTEP worden toegewezen anders wordt de allereerste keer dat stap #RESET\_ST wordt geëvalueerd de code in de regels 15 t/m 18 niet uitgevoerd. OLDSTEP moet dus een andere waarde krijgen dan STEP.

## 7.4 Het gebruik van timers in een sequencer

Een timer kan in een sequencer worden gebruikt om een bepaalde tijd af te meten. Soms is het nodig om gewoon een bepaalde tijd te wachten voordat naar een andere stap wordt gegaan zonder enige andere conditie. Maar het is ook mogelijk om een timer te gebruiken om een timeout te realiseren. Denk hierbij aan een beweging van een onderdeel van een opstelling. Als het onderdeel niet binnen een bepaalde tijd een eindmelder raakt wordt er

```

1 // If reset is active ...
2 IF "reset" = TRUE THEN
3     "STEP" := #RESET_ST;
4     "OLDSTEP" := #STEP2_ST;
5     "counter_entry_reset" := 0;
6     "counter_exit_reset" := 0;
7     "counter_entry_step1" := 0;
8     "counter_exit_step1" := 0;
9     "counter_entry_step2" := 0;
10    "counter_exit_step2" := 0;
11 ELSE
12     // Evaluate the current step
13     CASE "STEP" OF
14         #RESET_ST:
15             IF "OLDSTEP" <> "STEP" THEN
16                 "OLDSTEP" := "STEP";
17                 "counter_entry_reset" := "counter_entry_reset" + 1;
18             END_IF;
19
20             "STEP" := #STEP1_ST;
21
22             IF "OLDSTEP" <> "STEP" THEN
23                 "counter_exit_reset" := "counter_exit_reset" + 1;
24             END_IF;
25         #STEP1_ST:
26             IF "OLDSTEP" <> "STEP" THEN
27                 "OLDSTEP" := "STEP";
28                 "counter_entry_step1" := "counter_entry_step1" + 1;
29             END_IF;
30
31             IF "input" = TRUE THEN
32                 "STEP" := #STEP2_ST;
33             END_IF;
34
35             IF "OLDSTEP" <> "STEP" THEN
36                 "counter_exit_step1" := "counter_exit_step1" + 1;
37             END_IF;
38         #STEP2_ST:
39             IF "OLDSTEP" <> "STEP" THEN
40                 "OLDSTEP" := "STEP";
41                 "counter_entry_step2" := "counter_entry_step2" + 1;
42             END_IF;
43
44             IF "input" = FALSE THEN
45                 "STEP" := #STEP1_ST;
46             END_IF;
47
48             IF "OLDSTEP" <> "STEP" THEN
49                 "counter_exit_step2" := "counter_exit_step2" + 1;
50             END_IF;
51     END_CASE;
52 END_IF;

```

Listing 7.5: Code van de functie SequencerEntryExit.

gesprongen naar een foutroutine.

We bespreken een voorbeeld van een sequencer met een IEC-timer. Dit keer maken we gebruik van een functieblok om de sequencer in onder te brengen. Aan het functieblok wordt een datablok gekoppeld dat alle parameters van het functieblok herbergt. De declaratie van de parameters is te zien in figuur 7.2. We maken dit keer gebruik van een noodstopvoorziening die op een flank werkt (zie de parameters in regel 2 en 15). De sequencer heeft een ingang en een uitgang (regels 3 en 6). Natuurlijk heeft de sequencer ook een variabele voor STEP en OLDSTEP zodat binnenkomst en verlaten van een stap kan worden geprogrammeerd. Een opvallende regel is regel 13. Hierin wordt een IEC-timer TheTimer van het type TON (on-delay timer) gedeclareerd. Een IEC-timer heeft een eigen data-opslag. Die wordt ondergebracht in het datablok van de sequencer. De timer is dus als een lokale variabele gedeclareerd. Zie voor meer informatie over IEC-timers paragraaf 8.12.2. De variabele preset\_value wordt gebruikt de tijd van de timer op te geven (regel 14). De variabele start\_timer wordt gebruikt om de timer te starten (regel 16). In de regels 21 t/m 23 zijn de stapnummers gedeclareerd.

SequencerIECTimer							
	Name	Data type	Offset	Default value	Visible in ...	Setpoint	Comment
1	▼ Input				<input type="checkbox"/>	<input type="checkbox"/>	
2	▣ emergency_stop	Bool	0.0	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Emergency stop input
3	▣ input	Bool	0.1	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Just an input
4	▣ <Add new>				<input type="checkbox"/>	<input type="checkbox"/>	
5	▼ Output				<input type="checkbox"/>	<input type="checkbox"/>	
6	▣ output	Bool	2.0	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Just an output
7	▣ <Add new>				<input type="checkbox"/>	<input type="checkbox"/>	
8	▼ InOut				<input type="checkbox"/>	<input type="checkbox"/>	
9	▣ <Add new>				<input type="checkbox"/>	<input type="checkbox"/>	
10	▼ Static				<input type="checkbox"/>	<input type="checkbox"/>	
11	▣ STEP	Int	4.0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The current step (or the next one)
12	▣ OLDSTEP	Int	6.0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The previous step
13	▣ TheTimer	TON	8.0		<input checked="" type="checkbox"/>	<input type="checkbox"/>	The IEC timer
14	▣ preset_value	Time	30.0	T#0ms	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Timer value
15	▣ emergency_stop_edge_flag	Bool	34.0	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Edge detect flag for emergency stop
16	▣ start_timer	Bool	34.1	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Flag to start the timer
17	▣ <Add new>				<input type="checkbox"/>	<input type="checkbox"/>	
18	▼ Temp				<input type="checkbox"/>	<input type="checkbox"/>	
19	▣ <Add new>				<input type="checkbox"/>	<input type="checkbox"/>	
20	▼ Constant				<input type="checkbox"/>	<input type="checkbox"/>	
21	▣ EMERGENCY_STOP_ST	Int		0	<input type="checkbox"/>	<input type="checkbox"/>	
22	▣ STEP1_ST	Int		1	<input type="checkbox"/>	<input type="checkbox"/>	
23	▣ STEP2_ST	Int		2	<input type="checkbox"/>	<input type="checkbox"/>	

Figuur 7.2: Declaratie van het datablok van de sequencer met timer.

Het voordeel van een functieblok is dat alle parameters behouden blijven over aanroepen van het functieblok heen.

De code van de sequencer is te zien in listing 7.6. Het programma wordt begonnen met het aanroepen van de timer (regel 2). Dat moet iedere scan cycle gebeuren anders worden de parameters IN en PT niet door de timer geëvalueerd. Dit geldt ook voor de uitgangen Q en ET (De uitgangen hoeven niet opgegeven te worden zoals we later zullen zien). De evaluatie van de noodstop en de stappen is iets anders geprogrammeerd dan de voorbeelden hiervoor. In de regels 5 t/m 8 wordt de noodstop geëvalueerd. Hier wordt gebruik gemaakt van flankdetectie. Bij een opgaande flank op parameter emergency\_stop wordt de code uitgevoerd. Flankdetectie wordt besproken in paragraaf 8.11. De evaluatie van de stappen is los geprogrammeerd van de noodstopdetectie. Dat heeft als voordeel dat na een noodstopdetectie binnen dezelfde scan cycle ook de noodstopstap wordt uitgevoerd.

```

1 //Call timer to update inputs and outputs
2 #TheTimer(IN := #start_timer, PT := #preset_value);
3
4 // If there is a positive edge on emergency stop..
5 IF #emergency_stop = TRUE AND #emergency_stop_edge_flag = FALSE THEN
6     #STEP := #EMERGENCY_STOP_ST;
7     #OLDSTEP := #STEP1_ST;
8 END_IF;
9
10 // Evaluate the steps
11 CASE #STEP OF
12     #EMERGENCY_STOP_ST:
13         #OLDSTEP := #STEP;
14         #preset_value := T#0s;
15         #start_timer := FALSE;
16         #output := FALSE;
17         #STEP := #STEP1_ST;
18     #STEP1_ST:
19         // Entry code: start timer
20         IF #OLDSTEP <> #STEP THEN
21             #OLDSTEP := #STEP;
22             // Set the time
23             #preset_value := T#30s;
24             // Start the timer
25             #start_timer := TRUE;
26         END_IF;
27
28         // Timer expired?
29         IF #TheTimer.Q = TRUE THEN
30             #STEP := #STEP2_ST;
31         END_IF;
32
33         #output := TRUE;
34
35         // Exit code: reset timer
36         IF #OLDSTEP <> #STEP THEN
37             #preset_value := T#0s;
38             #start_timer := FALSE;
39         END_IF;
40     #STEP2_ST:
41         #OLDSTEP := #STEP;
42
43         #output := FALSE;
44
45         IF #input = TRUE THEN
46             #STEP := #STEP1_ST;
47         END_IF;
48 END_CASE;
49
50 // Update the emergency edge flag
51 #emergency_stop_edge_flag := #emergency_stop;

```

Listing 7.6: Code van het functieblok SequencerIECTimer.

In de stap `#EMERGENCY_STOP_ST` wordt de timer gestopt en de tijd op 0 seconden gezet. Daarna wordt direct naar de volgende stap gegaan.

De timer wordt actief gebruikt in stap `#STEP1_ST`. Bij de eerste binnenkomst van de stap wordt de tijd op 30 seconden gezet en de timer gestart (regels 23 en 25). Natuurlijk moet er gewacht worden tot de volgende scan cycle voordat de timer daadwerkelijk wordt gestart. Omdat de variabelen zijn opgeslagen in een datablok blijven ze ongewijzigd totdat er een andere waarde aan wordt toegekend. We hoeven de variabele `#start_timer` dus maar één keer op `TRUE` te zetten. In de regels 29 t/m 31 wordt gekeken of de timer niet verlopen is. Dat doen we door de Q-uitgang van de timer te testen. De Q-uitgang blijft namelijk `FALSE` zolang de tijd niet verstreken is. Nu is ook te zien waarom bij de aanroep van de timer in regel 2 de uitgangen niet hoeven te worden opgegeven. Het is namelijk mogelijk om de uitgangen van de timer direct te gebruiken. Als de tijd verstreken is wordt stap `#STEP2_ST` als nieuwe stap aangemerkt. Bij de laatste keer verlaten van de stap wordt de tijd weer op 0 seconden gezet en de timer uitgeschakeld. Merk op dat de IEC-timers gestart worden door een opgaande flank op de IN-parameter (zie regel 2). De variabele `#start_timer` moet dus eerst op `FALSE` gezet worden voordat het weer op `TRUE` gezet kan worden. Dit mag niet binnen dezelfde scan cycle gebeuren anders ziet de timer geen opgaande flank.

In stap `#STEP2_ST` wordt gewacht totdat ingang `#input` `TRUE` is. Daarna wordt naar stap `#STEP1_ST` gegaan.

In regel 51 wordt de variabele `#emergency_stop_edge_flag` bijgewerkt met de huidige waarde van de noodstopingang. Dit is nodig om in de volgende scan cycle een verandering op de noodstopingang te detecteren.

## 7.5 Parallele verwerking van sequencers

In veel productiesystemen is enige mate van parallellisme te ontdekken. Delen van de verwerking in zo'n systeem kunnen dan gelijktijdig uitgevoerd te worden. Het ligt dan voor de hand om een besturingsprogramma te ontwikkelen dat aansluit op de parallele productieprocessen.

Let wel: de PLC laat zich naar de gebruiker en de programmeur zien als een processor met één kern. Er is dus geen sprake van echt parallellisme. Het operating system van de PLC ondersteunt geen parallele taken op één processor. De enige uitzondering hierop zijn de Organization Blocks. Die kunnen het lopende programma op willekeurige momenten onderbreken (en werken dan nog steeds niet parallel).

In listing 7.7 is de opzet van twee parallele sequencers te zien. In feite zijn het twee sequencers die na elkaar doorlopen worden. Eerst wordt sequencer A doorlopen en daarna sequencer B. Sequencer A bestuurt de onderdelen van het ene deelproces en sequencer B de andere. Als de scan cycle-tijd maar kort genoeg is, lijkt het voor de gebruiker van het systeem of de twee sequencers parallel opereren.

In de code is te zien dat sequencer A synchroniseert op sequencer B. Daartoe wordt een variabele `reached_synchronize_point` gebruikt die tijdens de initialisatiestap in sequencer B op `FALSE` wordt gezet. In sequencer A wordt in een stap getest of de variabele op `TRUE` is gezet. Dan wordt naar een andere stap gegaan. Als de variabele `FALSE` is, blijft de sequencer in de huidige stap. In sequencer B wordt op een bepaald moment de synchro-



```

1  CASE STEP_SEQUENCER_A OF
2      ...
3      _some_step_A :
4          ...
5          IF reached_synchronize_point = TRUE THEN
6              STEP_SEQUENCER_A := _some_other_step_A;
7          END_IF;
8          ...
9      _some_other_step_A:
10         ...
11  END_CASE;
12
13  CASE STEP_SEQUENCER_B OF
14      _init_step_B :
15          reached_synchronize_point := FALSE;
16          ...
17      _some_step_B :
18          ...
19          reached_synchronize_point := TRUE;
20          ...
21  END_CASE;

```

Listing 7.7: Opzet parallele sequencers.

nisatievariabele op TRUE gezet waardoor sequencer A verder kan. Op dat moment zijn sequencer A en B gesynchroniseerd.

## 7.6 Parallele verwerking binnen een sequencer

Het is ook mogelijk om binnen een sequencer parallellisme te realiseren. We spreken dan liever van taken om het verschil aan te geven met parallel opererende sequencers. De taken worden uitgevoerd binnen één stap. De stap zelf doet vrij weinig. Het zorgt ervoor dat de taken gestart worden en wacht in de stap totdat beide taken zijn afgelopen. Dat moet ook wel want er kan niet zomaar naar een andere stap gegaan worden anders stoppen de taken met uitvoering.

We bespreken een voorbeeld van een sequencer met twee taken. Binnen de taken maken we gebruik van drie timers (type TON). Taak A gebruikt twee timers die na elkaar gestart worden (de eerste is afgelopen en de tweede begint). Taak B gebruikt één timer. De tijden zijn zo ingesteld dat taak A eerder klaar is dan taak B.

In figuur 7.3 is de interface van de sequencer te zien. We ontwikkelen de sequencer weer in een functieblok met bijbehorend datablok. De sequencer heeft een noodstop, een ingang en een uitgang (de uitgang gebruiken we verder niet). Verder zijn er variabelen om de stappen bij te houden, twee voor de sequencer en voor de taken elk één. Binnen de taken maken we geen gebruik van binnenkomst en verlaten. Dit is gedaan om het geheel overzichtelijk te houden. Verder zijn er variabelen om de tijden van de timers in te stellen en om de timers te starten en te stoppen. Er zijn twee variabelen waarmee de taken kunnen aangeven dat ze zijn afgerond.

De code van de sequencer met twee taken is te zien in de listings 7.8 en 7.9. Aan het begin

SequencerSimultaneous							
	Name	Data type	Offset	Default value	Visible in ...	Setpoint	Comment
1	Input						
2	emergency_stop	Bool	0.0	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Emergency stop input
3	input	Bool	0.1	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Just an input
4	Output						
5	output	Bool	2.0	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Just an output
6	InOut						
7	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	
8	Static						
9	STEP	Int	4.0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The current step (or the next one)
10	OLDSTEP	Int	6.0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The previous step
11	TASKA_STEP	Int	8.0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The current step of task A
12	TASKB_STEP	Int	10.0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The current step of task B
13	TimerA1	TON	12.0		<input checked="" type="checkbox"/>	<input type="checkbox"/>	IEC timer for task A
14	TimerA2	TON	34.0		<input checked="" type="checkbox"/>	<input type="checkbox"/>	IEC timer for task A
15	TimerB	TON	56.0		<input checked="" type="checkbox"/>	<input type="checkbox"/>	IEC timer for task B
16	timer1_pv	Time	78.0	T#0ms	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Timer value
17	timer2_pv	Time	82.0	T#0ms	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Timer value
18	timerb_pv	Time	86.0	T#0ms	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Timer value
19	emergency_stop_edg...	Bool	90.0	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Edge detect flag for emergency stop
20	timer1_start	Bool	90.1	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Flag to start the timer
21	timer2_start	Bool	90.2	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Flag to start the timer
22	timerb_start	Bool	90.3	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Flag to start the timer
23	taska_ready	Bool	90.4	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Task A ready flag
24	taskb_ready	Bool	90.5	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Task B ready flag
25	Temp						
26	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	
27	Constant						
28	EMERGENCY_STOP_ST	Int		0	<input type="checkbox"/>	<input type="checkbox"/>	
29	STEP1_ST	Int		1	<input type="checkbox"/>	<input type="checkbox"/>	
30	STEP2_ST	Int		2	<input type="checkbox"/>	<input type="checkbox"/>	
31	TASKA_INIT_ST	Int		0	<input type="checkbox"/>	<input type="checkbox"/>	
32	TASKA_STEP1_ST	Int		1	<input type="checkbox"/>	<input type="checkbox"/>	
33	TASKA_STEP2_ST	Int		2	<input type="checkbox"/>	<input type="checkbox"/>	
34	TASKA_STEP3_ST	Int		3	<input type="checkbox"/>	<input type="checkbox"/>	
35	TASKB_INIT_ST	Int		0	<input type="checkbox"/>	<input type="checkbox"/>	
36	TASKB_STEP1_ST	Int		1	<input type="checkbox"/>	<input type="checkbox"/>	
37	TASKB_STEP2_ST	Int		2	<input type="checkbox"/>	<input type="checkbox"/>	
38	TASKB_STEP3_ST	Int		3	<input type="checkbox"/>	<input type="checkbox"/>	

Figuur 7.3: Declaratie van het datablok van de sequencer met twee taken.

worden de timers aangeroepen om de ingangen en uitgangen te actualiseren. Daarna wordt getest of de noodstop is gebruikt (opgaande flank).

De taken worden uitgevoerd in stap #STEP1\_ST van de sequencer. Bij de eerste binnenkomst in de stap worden de taken geïnitieerd. De stapwaarde van de taken worden toegekend en de variabelen die aangeven dat een taak klaar is worden op FALSE gezet. Daarna wordt taak A doorlopen. Die start de eerste timer (TimerA1) en stopt de tweede timer (TimerA2). Daarna wordt gewacht tot de eerste timer is verlopen. Als dat zo is, wordt de tweede timer gestart. Dan wordt gewacht tot de tweede timer is afgelopen. In de laatste stap (#TASKA\_STEP3\_ST) wordt de ready-vlag op true gezet om aan te geven dat de taak klaar is. Taak B is iets eenvoudiger van opzet. Er wordt maar één timer gebruikt (TimerB). Als de timer is verlopen wordt deze weer gestopt. Daarna wordt aangegeven dat de taak klaar is.

Zoals eerder al is besproken wacht de sequencer in stap #STEP1\_ST totdat beide taken klaar zijn. Dit gebeurt na het doorlopen van taak B. Er is geen code voor het verlaten van de stap.

```

1 //Call timers to update inputs and outputs
2 #TimerA1(IN := #timera1_start, PT := #timera1_pv);
3 #TimerA2(IN := #timera2_start, PT := #timera2_pv);
4 #TimerB(IN := #timerb_start, PT := #timerb_pv);
5
6 // If there is a positive edge on emergency stop..
7 IF #emergency_stop = TRUE AND #emergency_stop_edge_flag = FALSE THEN
8     #STEP := #EMERGENCY_STOP_ST;
9     #OLDSTEP := #STEP1_ST;
10 END_IF;
11
12 // Evaluate the steps
13 CASE #STEP OF
14     #EMERGENCY_STOP_ST:
15         #OLDSTEP := #STEP;
16         #output := FALSE;
17         #STEP := #STEP1_ST;
18     #STEP1_ST:
19         // Entry code: start tasks
20         IF #OLDSTEP <> #STEP THEN
21             #OLDSTEP := #STEP;
22             // Set up start of tasks
23             #TASKA_STEP := #TASKA_INIT_ST;
24             #TASKB_STEP := #TASKB_INIT_ST;
25             // Reset task ready flags
26             #taska_ready := FALSE;
27             #taskb_ready := FALSE;
28         END_IF;
29
30         // Task A
31         CASE #TASKA_STEP OF
32             #TASKA_INIT_ST:
33                 // Start timer A1 and stop timer A2
34                 #timera1_pv := T#15s;
35                 #timera1_start := TRUE;
36                 #timera2_pv := T#0s;
37                 #timera2_start := FALSE;
38                 #TASKA_STEP := #TASKA_STEP1_ST;
39             #TASKA_STEP1_ST:
40                 // If timer ready...
41                 IF #TimerA1.Q = TRUE THEN
42                     #timera1_pv := T#0s;
43                     #timera1_start := FALSE;
44                     #timera2_pv := T#10s;
45                     #timera2_start := TRUE;
46                     #TASKA_STEP := #TASKA_STEP2_ST;
47                 END_IF;
48             #TASKA_STEP2_ST:
49                 // If timer ready...
50                 IF #TimerA2.Q = TRUE THEN
51                     #timera2_pv := T#0s;

```

Listing 7.8: Code van het functieblok SequencerSimultaneous (deel 1).

```

1         #timera2_start := FALSE;
2         #TASKA_STEP := #TASKA_STEP3_ST;
3         END_IF;
4         #TASKA_STEP3_ST:
5             // Signal task ready
6             #taska_ready := TRUE;
7         END_CASE;
8
9         // Task B
10        CASE #TASKB_STEP OF
11            #TASKB_INIT_ST:
12                // Start timer B
13                #timerb_pv := T#30s;
14                #timerb_start := TRUE;
15                #TASKB_STEP := #TASKB_STEP1_ST;
16            #TASKB_STEP1_ST:
17                // If timer ready
18                IF #TimerB.Q = TRUE THEN
19                    #TASKB_STEP := #TASKB_STEP2_ST;
20                END_IF;
21            #TASKB_STEP2_ST:
22                // Stop timer
23                #timerb_pv := T#0s;
24                #timerb_start := FALSE;
25                #TASKB_STEP := #TASKB_STEP3_ST;
26            #TASKB_STEP3_ST:
27                // Signal task ready
28                #taskb_ready := true;
29        END_CASE;
30
31        // Both tasks complete?
32        IF #taska_ready AND #taskb_ready THEN
33            #STEP := #STEP2_ST;
34        END_IF;
35
36        // No exit code
37        #STEP2_ST:
38            #OLDSTEP := #STEP;
39
40            #output := FALSE;
41
42            IF #input = TRUE THEN
43                #STEP := #STEP1_ST;
44            END_IF;
45        END_CASE;
46
47        // Update the emergency stop edge flag
48        #emergency_stop_edge_flag := #emergency_stop;

```

Listing 7.9: Code van het functieblok *SequencerSimultaneous* (deel 2).

## 8. TIPS, TRICKS & TROUBLESHOOT

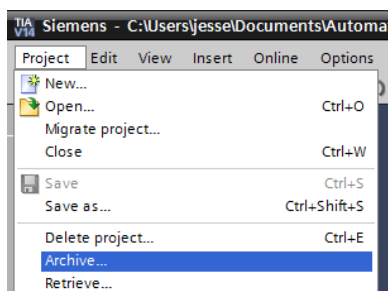
In dit hoofdstuk worden wat zaken behandeld die niet direct in een tutorial thuishoren maar wel handig zijn om te weten. Zo wordt uitgelegd hoe je projecten kan archiveren en gearchiveerde projecten kan inlezen. De PLC kan in STOP gaan na een fout, maar wat is er nu gebeurd? Dat is te zien via diagnostiek.

### 8.1 Projecten archiveren

Via TIA Portal is het mogelijk om projecten te archiveren. Alle bestanden van het project worden dan door middel van compressie in één bestand geplaatst. Dit archief, dat in feite een gewoon ZIP-bestand is, kan dan worden opgeslagen op bijvoorbeeld een USB-stick.

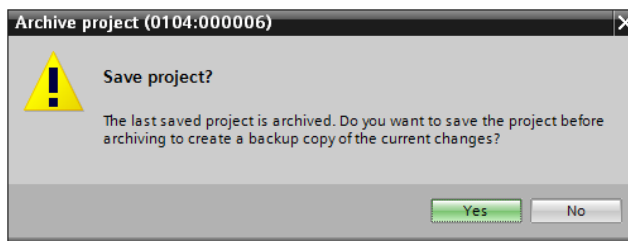
Opmerking: maak nooit zelf een archief aan, altijd via TIA Portal.

Selecteer in Tia Portal **Project**→**Archive**, zie figuur 8.1.



Figuur 8.1: Archiveren project.

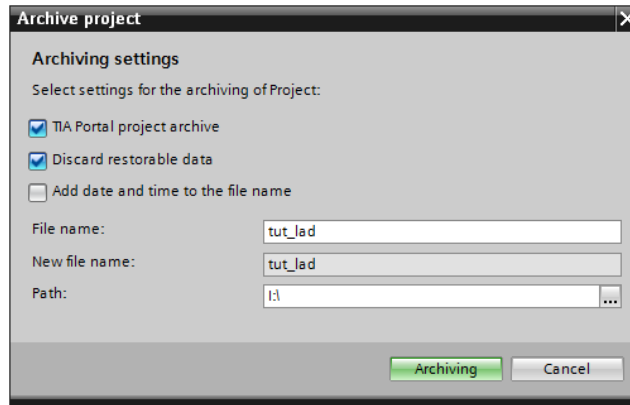
Vervolgens wordt gevraagd of het project opgeslagen moet worden, zie figuur 8.2. Klik op knop **Yes**.



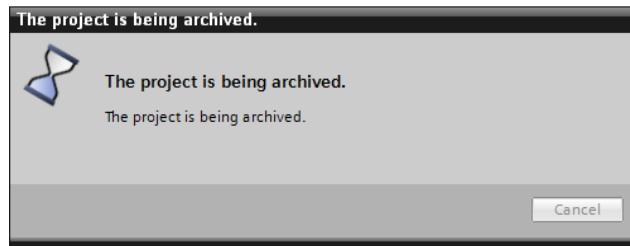
Figuur 8.2: Project opslaan.

Er wordt een dialoogvenster geopend waarin de naam van het archiefbestand en de plaats waar het moet worden opgeslagen moeten worden ingevuld. Zie figuur 8.3. In de figuur wordt het archief met de naam tut\_lad op schijf I: (USB-stick) opgeslagen.

Er volgt een venster waarin de voortgang wordt weergegeven, zie figuur 8.4.



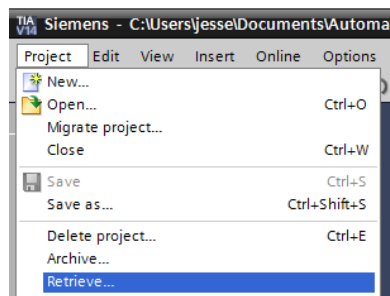
**Figuur 8.3:** Bestandsnaam en doelmap opgeven.



**Figuur 8.4:** Project wordt gearchiveerd.

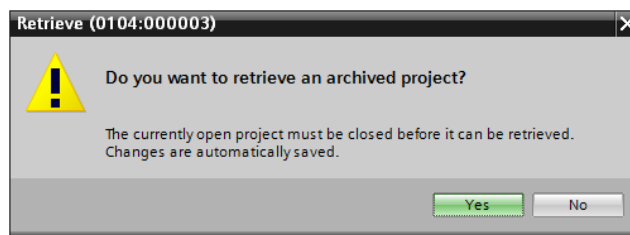
## 8.2 Projecten inlezen

Gearchiveerde projecten kunnen weer worden ingelezen. Daarna zijn ze te openen via TIA Portal. Selecteer in TIA Portal de menu-optie **Project** → **Retrieve** (figuur 8.5).

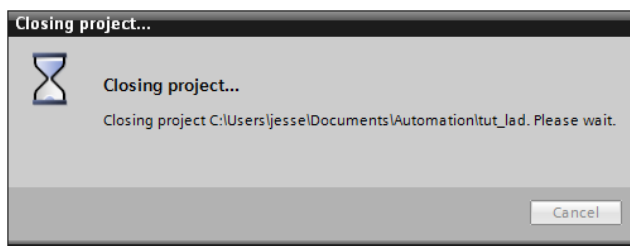


**Figuur 8.5:** Inlezen project.

TIA Portal vraagt of er een project moet worden ingelezen. Het huidige project wordt afgesloten. Zie de figuren 8.6 en 8.7.

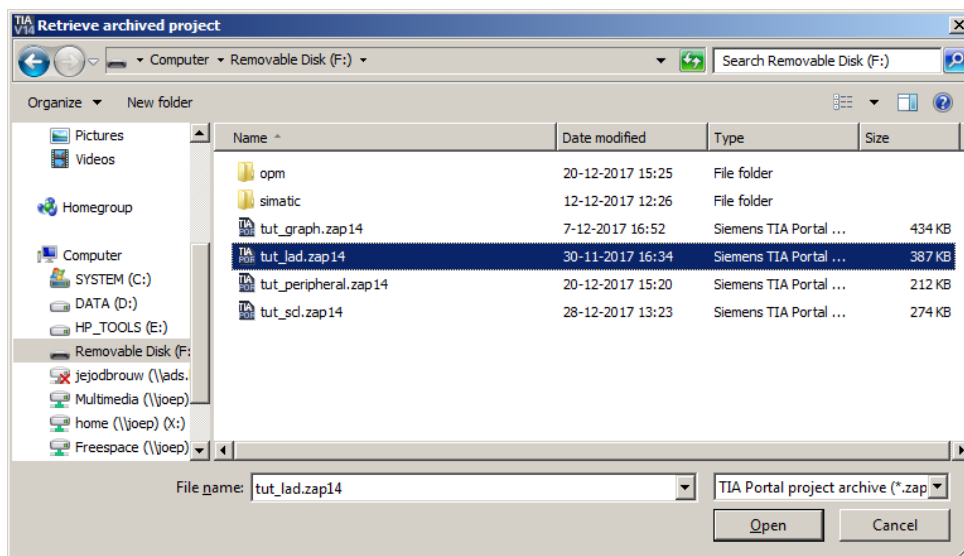


**Figuur 8.6:** Inlezen project.



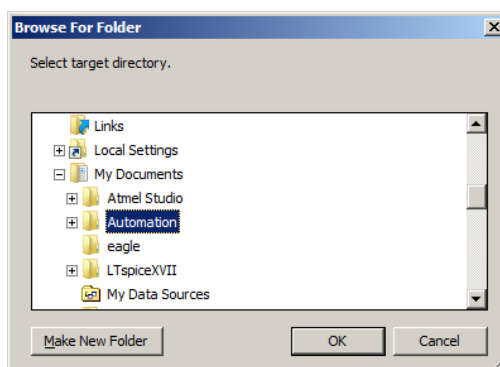
**Figuur 8.7:** Huidige project wordt afgesloten.

Selecteer het gearchiveerde project (figuur 8.8) en klik op de knop **Open**.



**Figuur 8.8:** Selecteer het opgeslagen project.

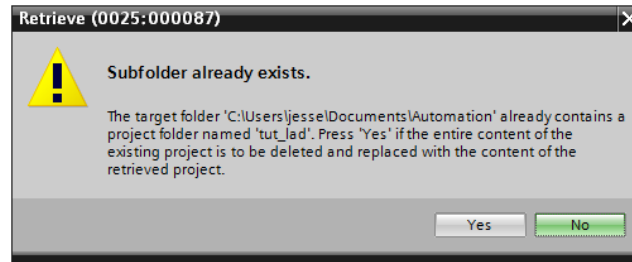
Hierna moet de map worden opgegeven waarin het project moet worden opgeslagen. Zoek een mooie plek uit (figuur 8.9).



**Figuur 8.9:** Selecteer de doelmap.

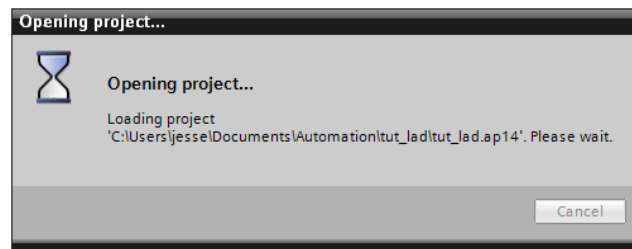
Het kan zijn dat de doelmap al bestaat. TIA Portal vraagt dan of de doelmap mag worden overschreven. Klik op de knop **Yes** als het bestaande project mag worden overschreven. Zie figuur 8.10.

**Let op! Het bestaande project wordt verwijderd!**



Figuur 8.10: Overschrijf de doelmap.

Nadat het project is ingelezen wordt het direct geopend. Zie figuur 8.11.



Figuur 8.11: Het project wordt geopend.

### 8.3 PLC diagnostiek

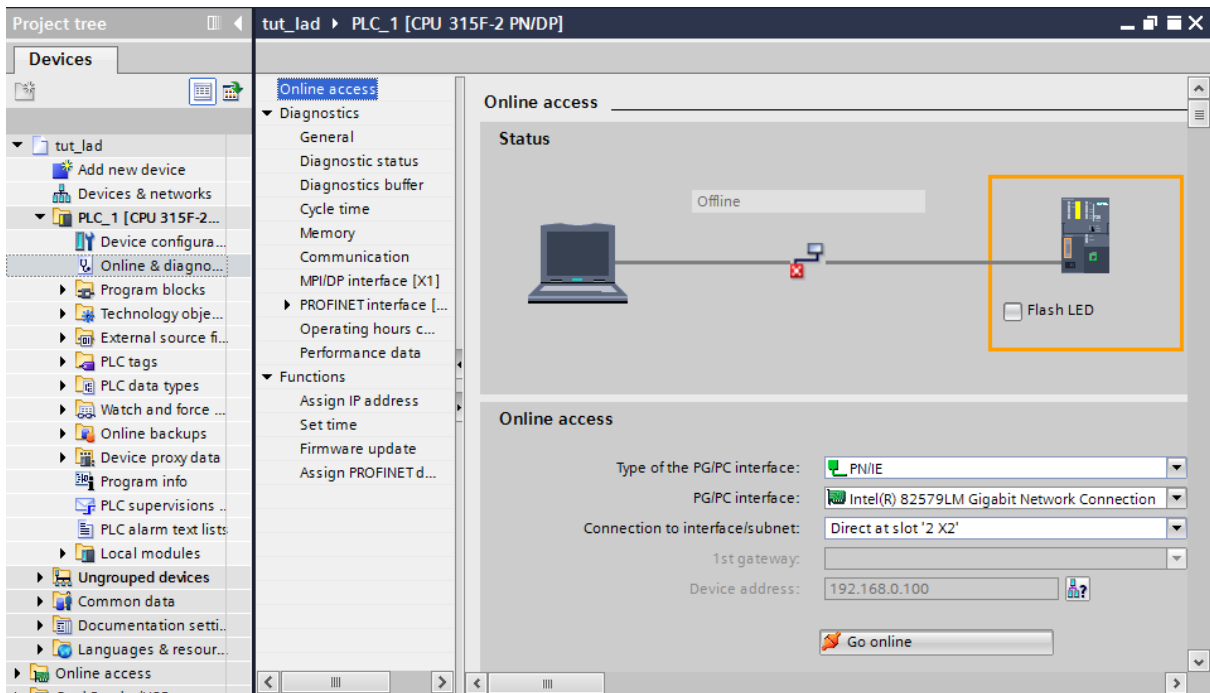
Als de PLC onverwacht in STOP gaat, is er meestal een serieus probleem. Maar waarom is de PLC in STOP gegaan? Er zijn veel verschillende problemen mogelijk. Een BCD-conversie kan zijn mislukt, een *I/O-slave* kan offline gegaan zijn, er kan een programmeerfout zijn gemaakt.

De PLC registreert al deze gebeurtenissen. Ze kunnen worden opgevraagd via het diagnostische menu. Dubbelklik in het linker paneel onder PLC\_1 op het onderdeel **Online & Diagnostics**. Zie figuur 8.12.

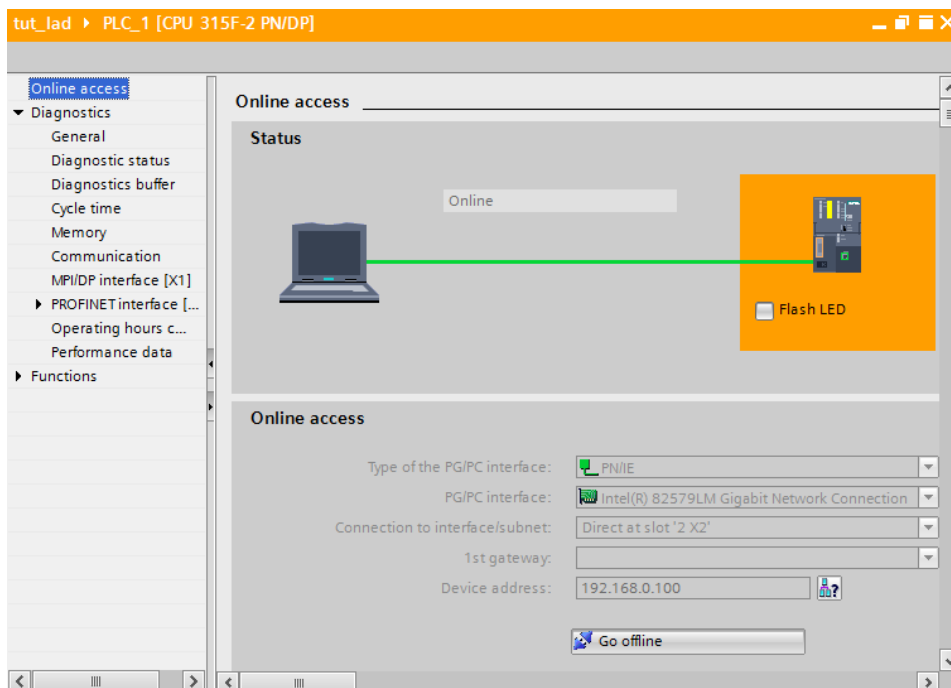
Er wordt een paneel in het midden van het scherm geopend waarin de diverse onderdelen geselecteerd kunnen worden. Vul de onderdelen in zoals het wordt getoond in de figuur. Klik daarna op de knop **Go Online**. Nu gaat de PC online met de PLC. Dat is te zien aan de balk aan de bovenkant; de kleur verandert naar oranje. Zie figuur 8.13.

Klik links in het middenpaneel op het menu-onderdeel **Diagnostics Buffer**. In het middenpaneel is nu een lijst met meldingen van de PLC te zien. Merk op dat de datum en de tijd van de PLC nogal verkeerd zijn ingesteld. Zie figuur 8.14.

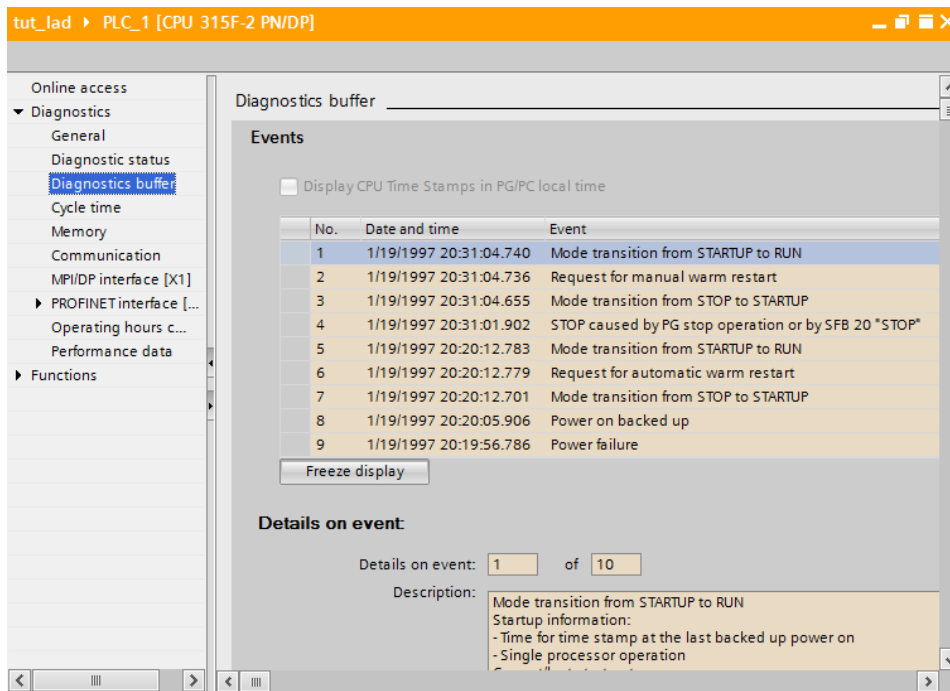




Figuur 8.12: Verbinding opzetten voor diagnostiek.



Figuur 8.13: De verbinding is opgezet.

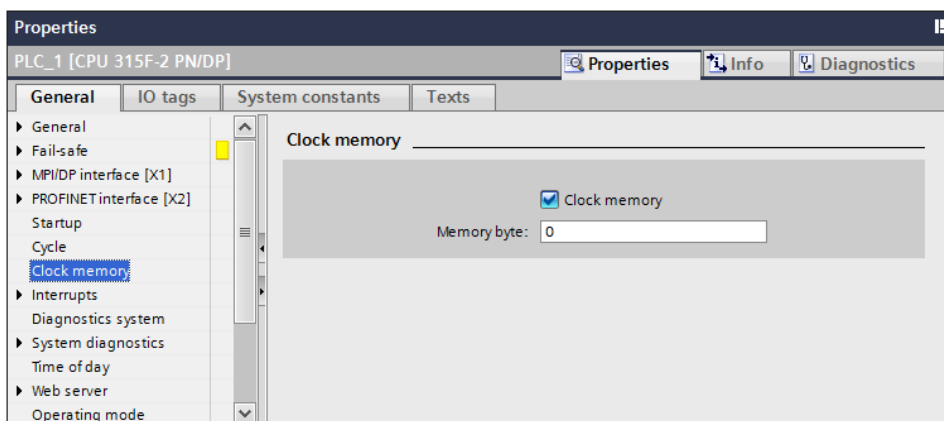


Figuur 8.14: Inhoud diagnostische buffer.

## 8.4 Clock Memory Byte

In de PLC is het mogelijk om één byte te reserveren als *clock memory*. De PLC genereert in elke bit een kloksignaal van verschillende frequenties. Alle kloksignalen hebben een duty cycle van 50%. Het gebruik van de clock memory wordt tijdens de configuratie opgegeven.

Dubbelklik in het linker paneel op het PLC menu-onderdeel **Device configuration**. Dubbelklik in het midden paneel op de CPU-module. Klik daarna in het paneel middenonder op Clock memory. Vink het onderdeel Clock memory aan en vul het byte-adres van de clock memory in. Zie figuur 8.15. In de figuur is byte M0.5 gereserveerd voor de clock memory.



Figuur 8.15: Instellen clock memory byte.

In tabel 8.1 staan alle alle frequenties en periodetijden vermeld. Om bijvoorbeeld een knipperlamp voor alarmering te maken met een frequentie van 1 Hz moet bit M0.5 ge-

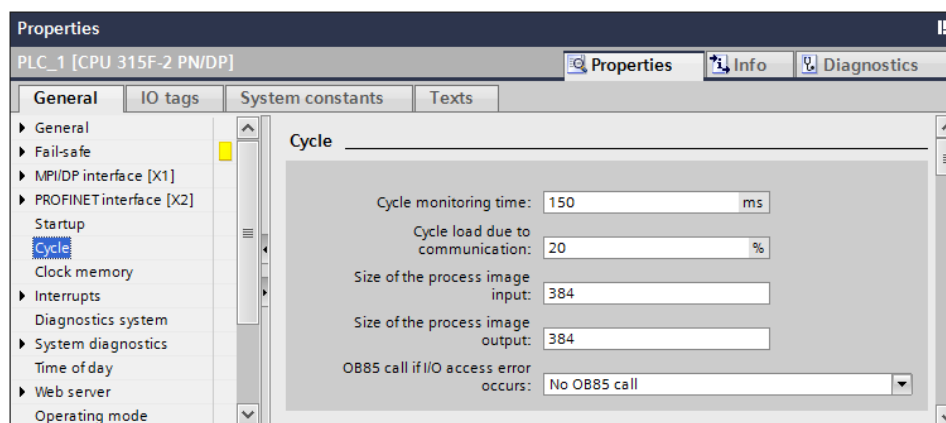
bruikt worden als MBO het clock memory byte is.

Tabel 8.1: Frequenties en periodetijden van de clock memory byte.

bit	7	6	5	4	3	2	1	0
Frequentie (Hz)	0,5	0,625	1,0	1,25	2,0	2,5	5,0	10,0
Periode (s)	2,0	1,6	1,0	0,8	0,5	0,4	0,2	0,1

## 8.5 Instellen maximum scan cycle tijd

Het is mogelijk om de maximale scan cycle tijd in te stellen. Dubbelklik in het linker paneel op het PLC menu-onderdeel **Device configuration**. Dubbelklik in het midden paneel op de CPU-module. Klik daarna in het paneel middenonder op **Cycle**. Vul in het onderdeel **Cycle monitoring time** de gewenste tijd in. Zie figuur 8.16.



Figuur 8.16: Instellen maximum scan cycle tijd.

## 8.6 Peripheral access

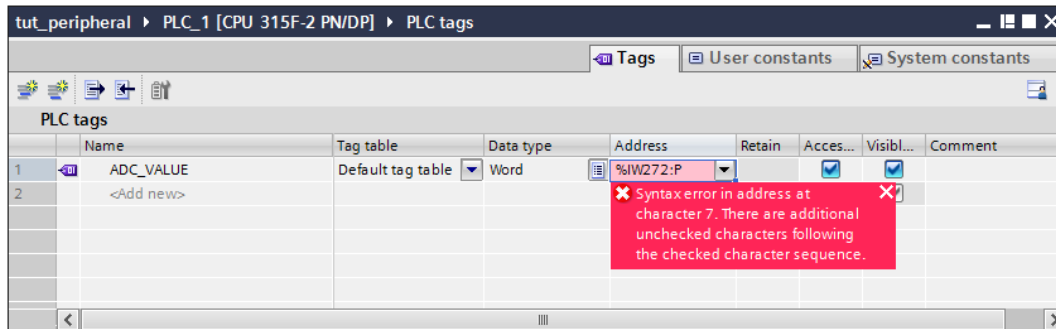
Zoals bekend maakt de PLC voor het starten van een scan cycle eerste een kopie van (een deel van) alle ingangen. Deze waarden worden dan in de *process image table* geplaatst. Het afvragen van ingangen vindt dan plaats met waarden uit deze tabel. Uitgangen worden eerst in de process image table geplaatst en pas aan het eind van de scan cycle naar buiten gestuurd (Bij Siemens is dat eigenlijk aan het begin van de scan cycle voordat de ingangen worden ingelezen.) De reden hiervoor is dat (in ieder geval voor de ingangen) de ingangswaarden consistent blijven tijdens het uitvoeren van een scan cycle.

Soms is het echter wenselijk om de momentane waarden van ingangen op te vragen. Een voorbeeld hiervan is bijvoorbeeld dat de ingangen buiten het bereik van de process image table liggen. Evenzo kan het nodig zijn om de uitgangen direct aan te sturen niet pas aan het eind van een scan cycle.

Het is mogelijk om ingangen op te vragen en uitgangen direct aan te sturen buiten de process image table om. Dat wordt *peripheral access* genoemd. Merk op dat peripheral access alleen met eenheden van byte, word en long word mogelijk zijn. Bits kunnen niet los worden opgevraagd of geschreven.

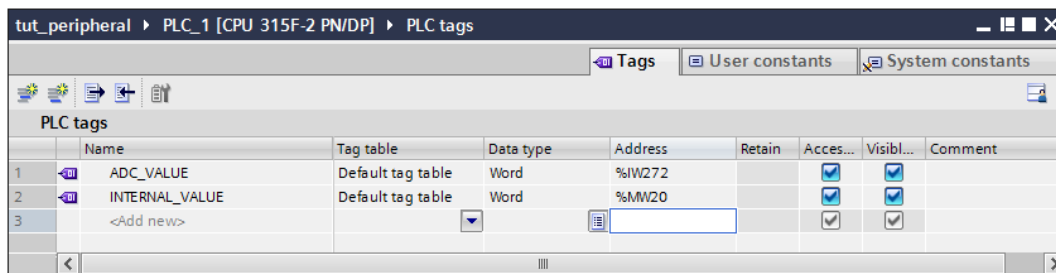
Peripheral access wordt tot stand gebracht door achter het adres de *qualifier* :P te plaatsen. Stel dat word-ingang IW272 moet worden ingelezen dan wordt dat met peripheral access dus %IW272:P.

Het is echter niet mogelijk om de qualifier :P in de PLC tags-tabel op te nemen zoals te zien is in figuur 8.17. Dat moet gebeuren in de bouwstenen als de adressen worden ingevoerd.

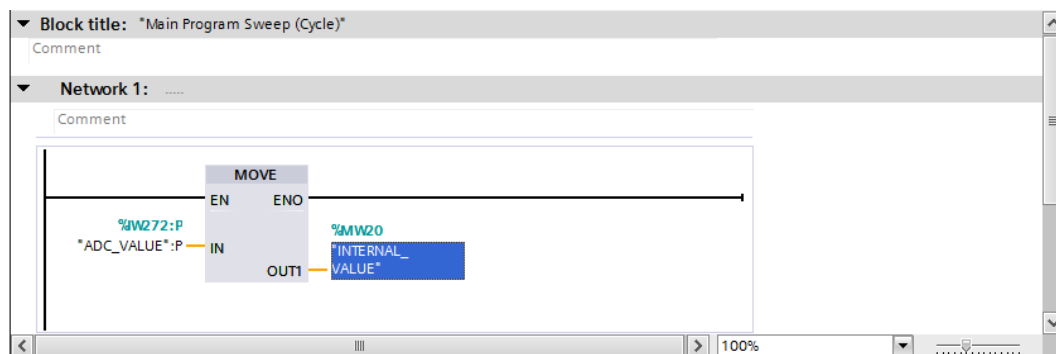


Figuur 8.17: Foutieve invoer PLC tag met peripheral access.

In figuur 8.18 is zijn twee tags te zien. De qualifier :P is niet ingevoerd. In figuur 8.19 is een rung uit OB1 te zien. Hierin een een move-blok te zien. Aan de ingangskant is nu de tag ADC\_VALUE met de qualifier :P te zien.



Figuur 8.18: Correcte invoer in de tags-tabel.

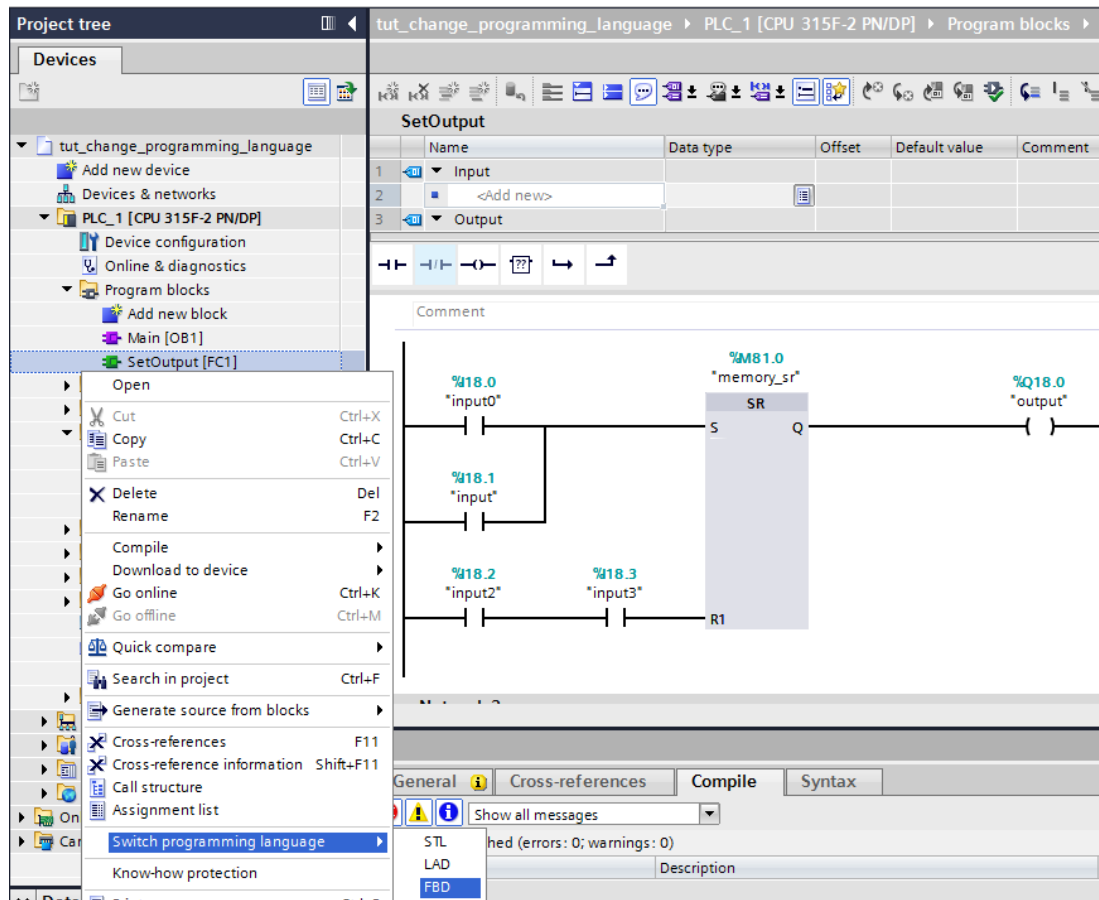


Figuur 8.19: Aanroep van peripheral access in OB1.

## 8.7 Schakelen tussen programmeertalen

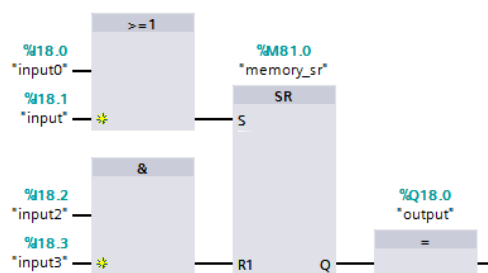
Het is mogelijk om te schakelen tussen de programmeertalen LAD, FBD en STL. In figuur 8.20 is een rung te zien in LAD. Deze rung wordt omgezet naar FBD. Selecteer het

blok dat omgezet moet worden en klik daarna op de rechter muisknop. Selecteer in het contextmenu Switch programming language en selecteer daarna FBD.



Figuur 8.20: Omzetten van een blok in LAD naar FBD.

In figuur 8.21 is het resultaat van de omzetting te zien.



Figuur 8.21: Het blok in FBD.

Het is op deze manier niet mogelijk om in Graph de programmeertaal van de overgangscondities te wijzigen. Dat moet op een andere manier gebeuren.

## 8.8 Importeren extern bronbestand

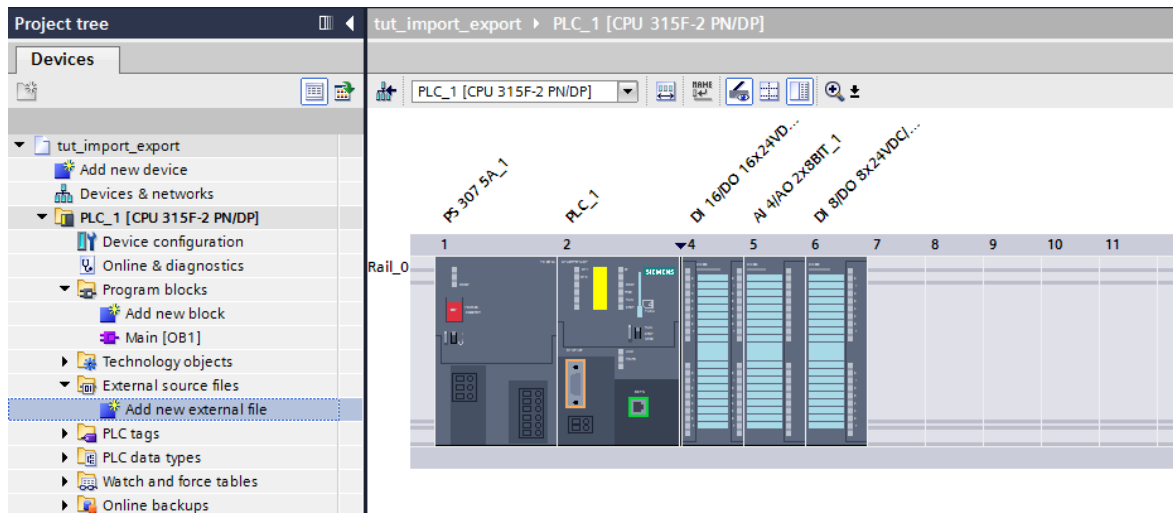
In TIA Portal is het mogelijk om externe bronbestanden te importeren. De volgende typen bronnen kunnen worden geïmporteerd: Statement List (STL)<sup>7</sup>, Structured Control Language

<sup>7</sup> Het bestand moet de extensie .awl hebben wat staat voor Anweisungsliste.

## CONCEPT

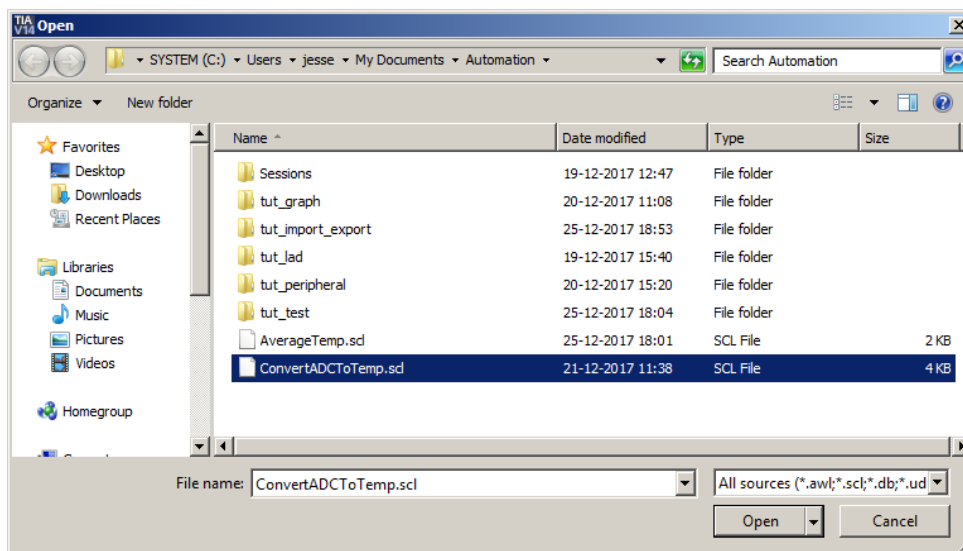
age (SCL), Data Block (DB) en User Data Type (UDT). Deze laatste heet in TIA Portal PLC Data Type. LAD en FDB kunnen niet (direct) geïmporteerd worden.

Dubbelklik om een extern bronbestand in het project te integreren in de Project Tree op *Add new external file*. Zie figuur 8.22.



Figuur 8.22: *Add new external file.*

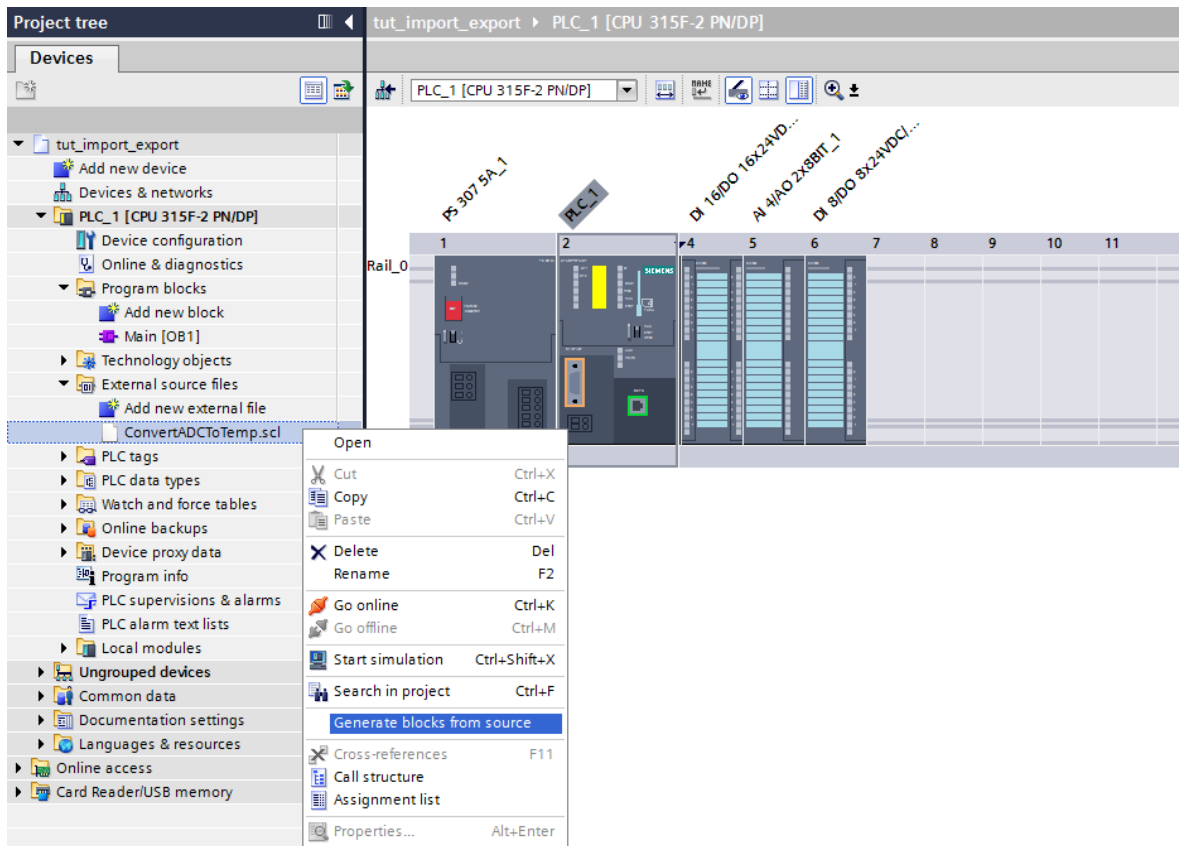
TIA Portal opent een dialoog waarin het externe bronbestand kan worden aangegeven. In dit geval betreft het een SCL-bestand. Zie figuur 8.23. Selecteer het bestand en klik op **Open**.



Figuur 8.23: *Selecteer extern bronbestand.*

Het externe bronbestand wordt ingevoegd onder het onderdeel *External source files*. Het bronbestand kan nu als blok in het project worden toegevoegd door het om te zetten. Klik op de naam van het externe bronbestand met de rechter muisknop. Selecteer in het contextmenu de optie *Generate blocks from source*. Zie figuur 8.24.

Er volgt een waarschuwing dat een bestand blok kan worden overschreven. Zie figuur 8.25. Klik op **OK**.

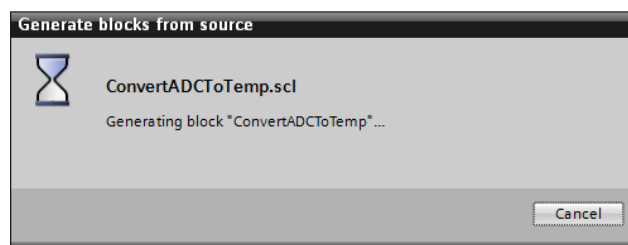


Figuur 8.24: Omzetten van een extern bronbestand in een blok.



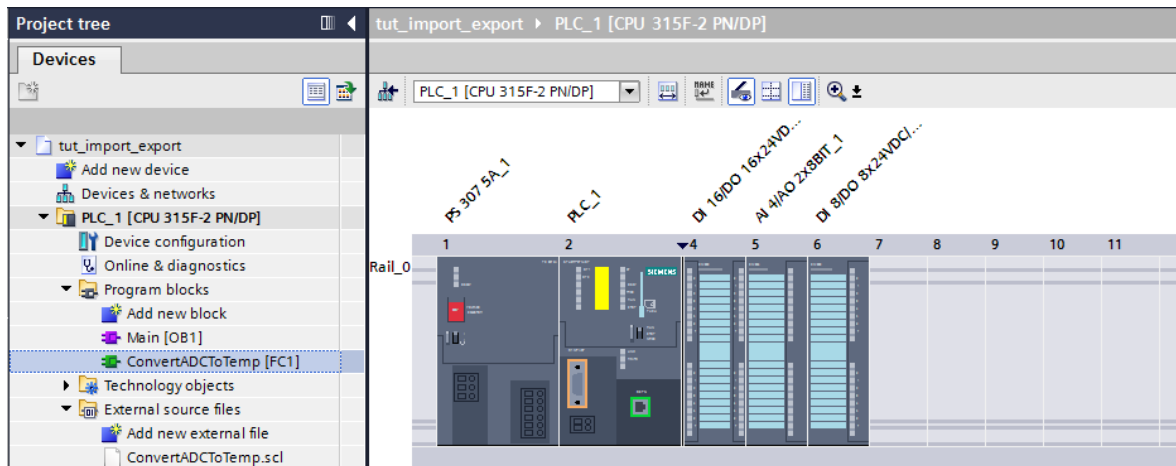
Figuur 8.25: Waarschuwing overschrijven blok.

Daarna volgt een scherm waarop de voortgang is te zien. Zie figuur 8.26.



Figuur 8.26: Voortgang conversie extern bronbestand naar blok.

Na het importeren is het blok toegevoegd aan de lijst met blokken. Zie figuur 8.27.



Figuur 8.27: Het blok is gegenereerd.

## 8.9 Exporteren naar een extern bronbestand.

In TIA Portal is het mogelijk om een blok te exporteren naar een extern bronbestand. De volgende programmeertalen kunnen geëxporteerd worden: Statement List (STL)<sup>8</sup>, Structured Control Language (SCL), Data Block (DB) en User Data Type (UDT). Deze laatste heet in TIA Portal PLC Data Type. LAD en FDB kunnen niet (direct) geïmporteerd worden. Daarvoor moeten ze eerst omgezet worden naar STL. Zie paragraaf 8.7. Graph kan niet geëxporteerd worden.

Selecteer het blok dat geëxporteerd moet worden. Klik daarna op de rechter muisknop en selecteer het menu-onderdeel **Generate source from blocks** → **Selected blocks only**. Zie figuur 8.28.

TIA Portal opent daarna een dialoog waarin wordt gevraagd om de bestandsnaam op te geven. Selecteer de juiste map en vul de bestandsnaam in en klik op **Save**. Zie figuur 8.29.

## 8.10 Dupliceren van een project

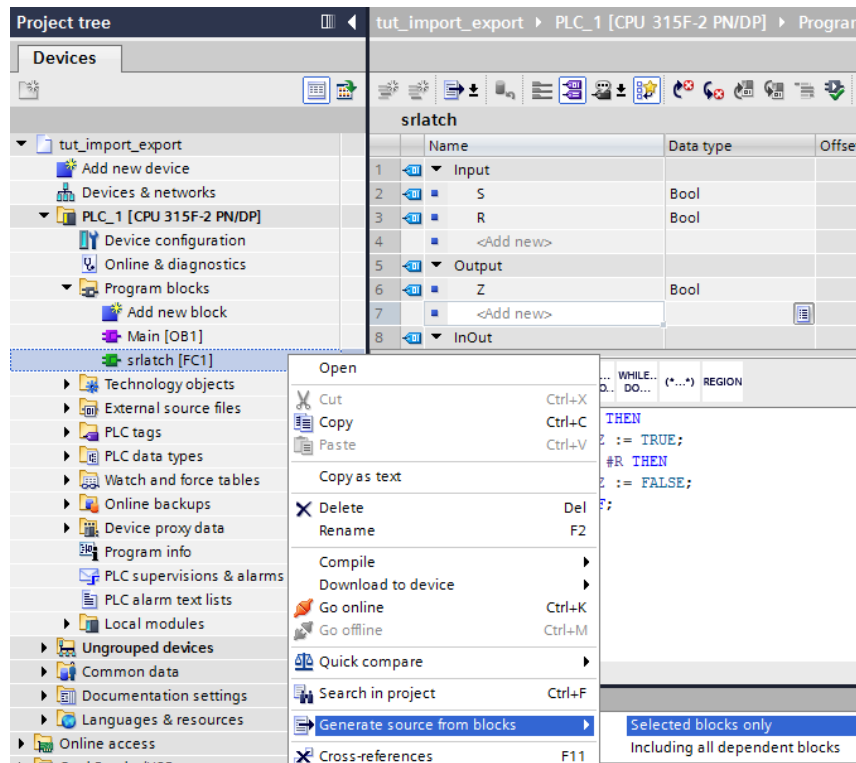
Van een project is eenvoudig een kopie (of duplicaat) te maken. Navigeer in Explorer naar de map waarin het project is opgeslagen en selecteer het project. Maak een kopie aan (rechter muisknop en dan Copy). Hernoem de kopie-map naar de gewenste naam. Navigeer vervolgens in de kopie-map. Daar is een projectbestand te vinden dat eindigt met de extensie ap14. Hernoem het bestand naar een nieuwe naam en laat de extensie intact. Er is nu een volledige kopie gemaakt van het project.

## 8.11 Flankdetectie in SCL

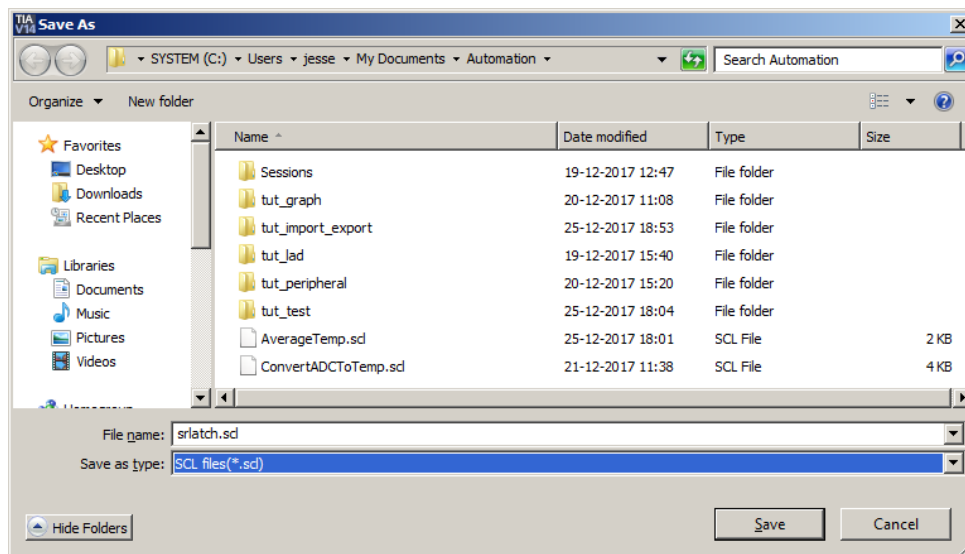
In LAD, FBD en STL is met eenvoudige bouwstenen of instructies mogelijk om een opgaande of neergaande flank te detecteren. In SCL bestaat zo'n functie niet, het moet zelf door de programmeur geprogrammeerd worden. Gelukkig is het eenvoudig te realiseren middels een merkerbit.

<sup>8</sup> Het bestand krijgt de extensie .awl wat staat voor Anweisungsliste.





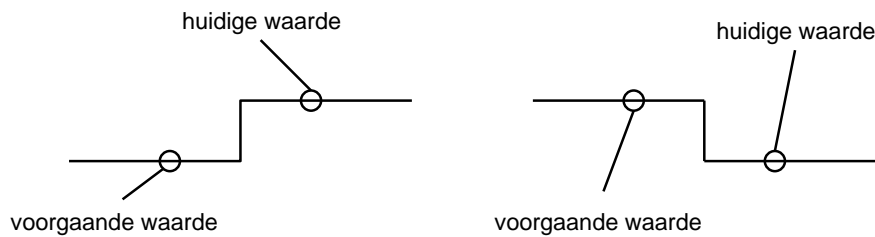
Figuur 8.28: Selectie te exporteren blok.



Figuur 8.29: Specificeer de naam van het te exporteren blok.

In figuur 8.30 te zien hoe een opgaande of neergaande flank gedetecteerd kan worden. Voor het detecteren van een opgaande flank wordt vergeleken of de huidige waarde van een ingang (of merker) logisch 1 is en de vorige waarde een 0. Zo ja, dan heeft de ingang een sprong gemaakt van 0 naar 1. Voor een neergaande flank is het precies andersom; als de huidige waarde een logische 0 is en de vorige waarde een 1 dan heeft de ingang een sprong gemaakt van 1 naar 0.

Het detecteren gebeurt tijdens het doorlopen van een scan cycle. In de volgende scan cycle is de huidige waarde dan natuurlijk de voorgaande waarde. Het merkerbit met de



**Figuur 8.30:** Huidige en voorgaande waarden bij flankdetectie.

voorgaande waarde moet dus aangepast worden.

De code is te zien in figuur 8.31. Het eerste deel zorgt voor detectie van de opgaande flank en het tweede deel zorg voor detectie van de neergaande flank. Let erop dat de voorgaande waarde bijgewerkt wordt, zie regel 20.

IF...	CASE... OF...	FOR... TO DO...	WHILE... DO...	(*...*)	REGION
					1 // Detect rising edge
					2 IF "INPUT" = TRUE AND "INPUT_PREV" = FALSE THEN
					3 // Rising edge detected, do your stuff
					4 "OUTPUT1" := TRUE;
					5 ELSE
					6 // Not a rising edge
					7 "OUTPUT1" := FALSE;
					8 END_IF;
					9
					10 // Detect falling edge
					11 IF "INPUT" = FALSE AND "INPUT_PREV" = TRUE THEN
					12 // Falling edge detected, do your stuff
					13 "OUTPUT1" := TRUE;
					14 ELSE
					15 // Not a falling edge
					16 "OUTPUT1" := FALSE;
					17 END_IF;
					18
					19 // Update previous value;
					20 "INPUT_PREV" := "INPUT";

**Figuur 8.31:** SCL-code voor het detecteren van een opgaande en neergaande flank.

## 8.12 Timers is SCL

In SCL is het mogelijk om timers te gebruiken. Zowel de Simatic als de IEC-timers worden ondersteund. Er worden voorbeelden gegeven van beide type timers.

### 8.12.1 Simatic timers

In dit voorbeeld wordt een Simatic timer gebruikt. Eerst wordt een tags-tabel opgesteld met diverse variabelen. Zie figuur 8.32.

De code voor instantiëring en gebruik van de timer is te zien in figuur 8.33. De timer gedraagt zich in SCL als een functie. Het heeft dan ook een teruggaveparameter (hier timer\_val\_return) van het type S5Time. De parameter T\_NO het nummer van de timer. Dat mag van het type Timer zijn zoals in het voorbeeld te zien is, maar het mag ook een Int zijn. De parameter S is de startingang van de timer. Deze is van het type Bool. De parameter TV is de startwaarde van de timer. Deze is van het type S5Time. In het voorbeeld is een constante gebruikt. De parameter R is de resetingang van de timer van het type Bool. De parameter BI geeft de interne timerwaarde terug en is van het type Word. De laatste parameter, Q, is de Q-uitgang van de timer en is van het type Bool. Deze is (in dit geval)

	Name	Data type	Address	Retain	Acces...	Visibl...	Comment
1	start_timer	Bool	%I8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	timer	Timer	%T0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	reset_timer	Bool	%I8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	value_timer	Word	%MW20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	output_timer	Bool	%Q8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	timer_val_return	S5Time	%MW22		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	q_timer	Bool	%M8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
8	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figuur 8.32: Tags-tabel voor gebruik van een timer.

TRUE als de timer aan het aftellen is.

Het is mogelijk om de Q- en BI-parameters te gebruiken in vergelijkingen. Let er wel op dat BI van het type Word is en eerst omgezet moet worden naar een Int. Dat wordt gerealiseerd met de functie WORD\_TO\_INT. Verder is het mogelijk om de teruggaveparameter te vergelijken. Hier is echter eerst een conversie nodig van het type S5Time naar het type Time. Dit wordt gerealiseerd door de functie S5TIME\_TO\_TIME. Let erop dat de SCL-compiler de conversie niet zelf regelt. Dit wordt gedaan door de functies DT\_TIME (FC6) en S5TI\_TIM (FC33). Deze functies zijn te vinden onder Program blocks→System blocks→Program resources. Ze moeten met de overige blokken in de PLC geladen worden.

### 8.12.2 IEC timers

In dit voorbeeld wordt een IEC timer gebruikt. Eerst wordt een tags-tabel opgesteld met diverse variabelen. Zie figuur 8.34.

```

IF... CASE... FOR... WHILE... (*...*) REGION
OF... TO DO... DO...

1 // Instantiate timer 0
2 "timer_val_return" := S_PEXT(T_NO:="timer", S:="start_timer", TV:=s5t#10s, R:="reset_timer", BI=>"value_timer", Q=>"q_timer");
3
4 // Do something with value of q_timer
5 IF "q_timer" = TRUE THEN
6     // Do your stuff if Q output of timer is true
7     "output_timer" := TRUE;
8 ELSE
9     // Do your stuff if Q output of timer is false
10    "output_timer" := FALSE;
11 END_IF;
12
13 // Do something with the internal timer value
14 IF WORD_TO_INT("value_timer") > 50 THEN
15     // Do stuff if internal timer value > 50 ...
16     "output_timer" := TRUE;
17 ELSE
18     // ... do stuff if not
19     "output_timer" := FALSE;
20 END_IF;
21
22 // Test the return value of the timer function
23 IF S5TIME_TO_TIME("timer_val_return") > T#10s THEN
24     "output_timer" := TRUE;
25 ELSE
26     "output_timer" := FALSE;
27 END_IF;

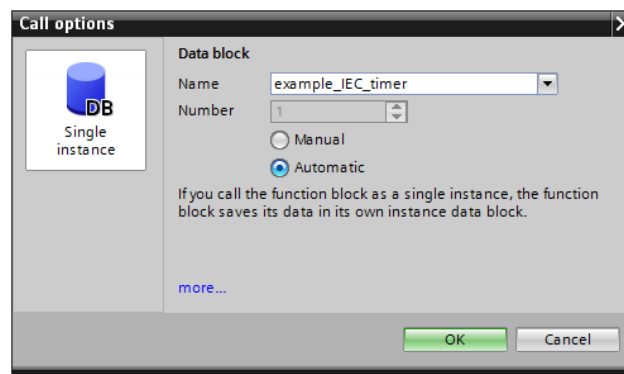
```

Figuur 8.33: SCL-code voor het instantiëren en gebruiken van een Simatic timer.

	Name	Data type	Address	Retain	Acces...	Visibl...	Comment
1	start_timer	Bool	%I8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	q_timer	Bool	%M8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	elapsed_time_timer	Time	%MD20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	output	Bool	%Q8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

**Figuur 8.34:** Tags-tabel voor gebruik van een IEC timer.

Een IEC timer wordt gerealiseerd door middel van een System Function Block (SFB). Bij het instantiëren van de timer wordt daarom ook eerst gevraagd naar het aanmaken van een datablok. Zie figuur 8.35. De naam van het datablok kan eventueel gewijzigd worden zoals in de figuur te zien is. Aangezien het hier niet om een “gebruikers“-blok gaat, wordt het datablok opgeslagen onder Program blocks→System blocks→Program resources.



**Figuur 8.35:** Aanmaken van een datablok voor gebruik van een IEC timer.

Na aanmaken van het datablok wordt in de SCL-editor de naam van het datablok gebruikt, samen met de naam van het type timer (in dit geval TP). De code is te zien in figuur 8.36.

```

1 // Instantiate a TP IEC timer
2 "example_IEC_timer".TP(IN:="start_timer", PT:=T#10s, Q=>"q_timer", ET=>"elapsed_time_timer");
3
4 // Test Q output of timer
5 IF "example_IEC_timer".Q = TRUE THEN
6   "output" := TRUE;
7 ELSE
8   "output" := FALSE;
9 END_IF;
10
11 // Use elapsed time in comparison
12 IF "example_IEC_timer".ET > T#8s THEN
13   "output" := TRUE;
14 ELSIF "example_IEC_timer".ET > T#5s THEN
15   "output" := FALSE;
16 ELSE
17   "output" := TRUE;
18 END_IF;

```

**Figuur 8.36:** SCL-code voor het instantiëren en gebruiken van een IEC timer.

De timer heeft vier parameters. De parameter IN is van het type Bool en wordt gebruikt om de timer te starten. Een opgaande flank start de timer. De parameter PT is van het

type Time (en niet S5Time) en bevat de te tellen tijd. De parameter Q van het type Bool. De werking is afhankelijk van het type timer. Bij het type TP is Q TRUE zolang de tijd niet verstreken is. De parameter ET is van het type Time en geeft de verstreken tijd aan.

Aangezien de parameteroverdracht via een datablok wordt geregeld is het mogelijk om de parameters via het datablok te manipuleren. In het codevoorbeeld is te zien dat de Q- en EV-parameters op deze manier worden gebruikt in vergelijkingen. Hierdoor is mogelijk om de Q- en ET-parameters weg te laten bij de instantiëring. Ook kunnen de IN- en PT-parameters op deze wijze worden toegekend. Ze moeten wel voorkomen in parameterlijst en gekoppeld worden aan bestaande variabelen of constanten.

### 8.13 Counters in SCL

Het is mogelijk om counters in SCL te gebruiken. Zowel de Simatic als de IEC counters worden ondersteund.

#### 8.13.1 Simatic counters

In dit voorbeeld wordt een Simatic counter van het type S\_CU gebruikt. Eerst wordt een tags-tabel opgesteld met diverse variabelen. Zie figuur 8.37.

	Name	Data type	Address	Retain	Acces...	Visibl...	Comment
1	set_counter	Bool	%I8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	reset_counter	Bool	%I8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	counter	Counter	%C0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	upcount_counter	Bool	%I8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	value_counter	Word	%MW20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	q_counter	Bool	%M8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	counter_val_return	Word	%MW24		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
8	output_counter	Bool	%Q8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figuur 8.37: Tags-tabel voor gebruik van een counter.

De code voor instantiëring en gebruik van de counter is te zien in figuur 8.38. De counter gedraagt zich in SCL als een functie. Het heeft dan ook een teruggaveparameter (hier counter\_val\_return) van het type Word. Merk op dat de teruggaveparameter weliswaar een Word is maar de counterwaarde als BCD-getal teruggeeft.

De parameter C\_NO het nummer van de timer. Dat mag van het type Counter zijn zoals in het voorbeeld te zien is, maar het mag ook een Int zijn. De parameter CU is van het type Bool en zorgt ervoor dat de counterwaarde met één verhoogd wordt. De parameter S is ook van het type Bool en zorgt ervoor dat de counter geladen wordt met de preload value, de waarde die wordt gegeven aan de parameter PV. Merk op dat PV van het type Word maar geeft de counterwaarde in BCD-formaat weer. De parameter R is van het type Bool en zorgt ervoor dat de counterwaarde op 0 gezet wordt (reset).

De parameter CV is van het type Word en geeft de counterwaarde terug als een integer-getal (dus niet als BCD-getal). De parameter Q geeft aan of de counterwaarde groter is dan 0. In dat getal is de waarde van Q TRUE.

De Q-uitgang is te gebruiken in vergelijkingen om testen of de counter de waarde 0 (of

```

1 // Instantiate counter 0
2 "counter_val_return":=S_CU(C_NO:="counter", CU:="upcount_counter", S:="set_counter", FV:=16#98,
3   R:="reset_counter", CV=>"value_counter", Q=>"q_counter");
4
5 // Test the Q output of the timer
6 IF "q_counter" = TRUE THEN
7   // Do something
8   "output_counter" := TRUE;
9 ELSE
10  "output_counter" := FALSE;
11 END_IF;
12
13 // Test the counter value, value as integer
14 IF WORD_TO_INT("value_counter") > 100 THEN
15   "output_counter" := TRUE;
16 ELSE
17   "output_counter" := FALSE;
18 END_IF;
19
20 // Test the return value, value as BCD
21 IF BCD16_TO_INT("counter_val_return") > 100 THEN
22   "output_counter" := TRUE;
23 ELSE
24   "output_counter" := FALSE;
25 END_IF;

```

**Figuur 8.38:** SCL-code voor het instantiëren en gebruiken van een counter.

niet) heeft. De CV-uitgang is te gebruiken in vergelijkingen. Merk op dat CV van het type Word is en eerst moet worden omgezet naar een Int. Dat wordt gedaan door de functie WORD\_TO\_INT. De teruggaveparameter kan ook gebruikt worden in vergelijking maar let erop dat deze waarde in BCD-formaat is. Om het te vergelijken met een integer moet de waarde eerst geconverteerd worden. Dat wordt gedaan door de functie BCD16\_TO\_INT.

Merk op dat de andere twee type counters, S\_CD en S\_CUD, ook andere parameters hebben.

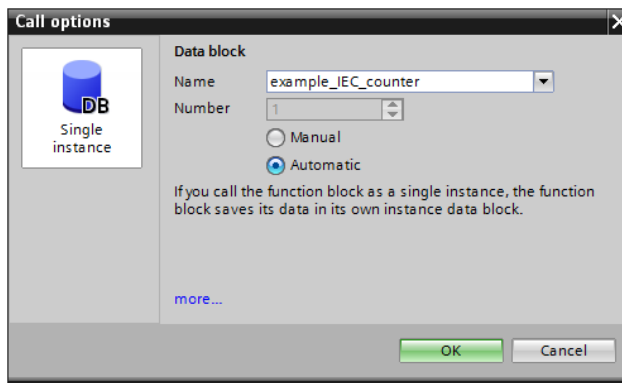
### 8.13.2 IEC counters

In dit voorbeeld wordt een IEC counter gebruikt. IEC counters hebben een groter telbereik dan Simatic counters, van -32768 t/m 32767 (type Int). Eerst wordt een tags-tabel opgesteld met diverse variabelen. Zie figuur 8.39.

	Name	Data type	Address	Retain	Acces...	Visibl...	Comment
1	count_up_counter	Bool	%I8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	reset_counter	Bool	%I8.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	q_counter	Bool	%I8.2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	count_value_counter	Int	%MW20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

**Figuur 8.39:** Tags-tabel voor gebruik van een IEC counter.

Een IEC counter wordt gerealiseerd door middel van een System Function Block (SFB). Bij het instantiëren van de counter wordt daarom ook eerst gevraagd naar het aanmaken van een datablok. Zie figuur 8.40. De naam van het datablok kan eventueel gewijzigd worden zoals in de figuur te zien is. Aangezien het hier niet om een “gebruikers“-blok gaat, wordt het datablok opgeslagen onder Program blocks→System blocks→Program resources.



**Figuur 8.40:** Aanmaken van een datablok voor gebruik van een IEC counter.

Na aanmaken van het datablok wordt in de SCL-editor de naam van het datablok gebruikt, samen met de naam van het type counter (in dit geval CTU). De code is te zien in [figuur 8.41](#).

```

IF... CASE... FOR... WHILE... (*...*) REGION
OF... TO DO... DO...

1 // Instantiate an IEC count-up counter
2 "example_IEC_counter".CTU(CU := "count_up_counter", R := "reset_counter", PV := 10,
3   Q => "q_counter", CV => "count_value_counter");
4
5 // Test de Q-output of the counter
6 IF "example_IEC_counter".Q = TRUE THEN
7   "output" := TRUE;
8 ELSE
9   "output" := FALSE;
10 END_IF;
11
12 // Test de CV-output of the counter
13 IF "example_IEC_counter".CV > 8 THEN
14   "output" := TRUE;
15 ELSIF "example_IEC_counter".CV > 5 THEN
16   "output" := FALSE;
17 ELSE
18   "output" := TRUE;
19 END_IF;

```

**Figuur 8.41:** SCL-code voor het instantiëren en gebruiken van een IEC counter.

De counter heeft vijf parameters. De parameter CU is van het type Bool en wordt gebruikt om de counter met één te verhogen. De parameter R is van het type Bool en zet de counter op 0 (reset). De parameter PV is van het type Int. De parameter Q van het type Bool. De werking is afhankelijk van het type counter. In dit geval wordt Q TRUE als de counter de telwaarde bereikt die is opgegeven bij PV. De parameter CV is van het type Int en geeft de huidige telwaarde aan.

Aangezien de parameteroverdracht via een datablok wordt geregeld is het mogelijk om de parameters via het datablok te manipuleren. In het codevoorbeeld is te zien dat de Q- en CV-parameters op deze manier worden gebruikt in vergelijkingen. Hierdoor is mogelijk om de Q- en CV-parameters weg te laten bij de instantiëring. Ook kunnen de IN-, R en PT-parameters op deze wijze worden toegekend. De R-parameter mag weggelaten worden in de parameterlijst, de andere twee niet. Ze moeten wel voorkomen in parameterlijst en gekoppeld worden aan bestaande variabelen of constanten.

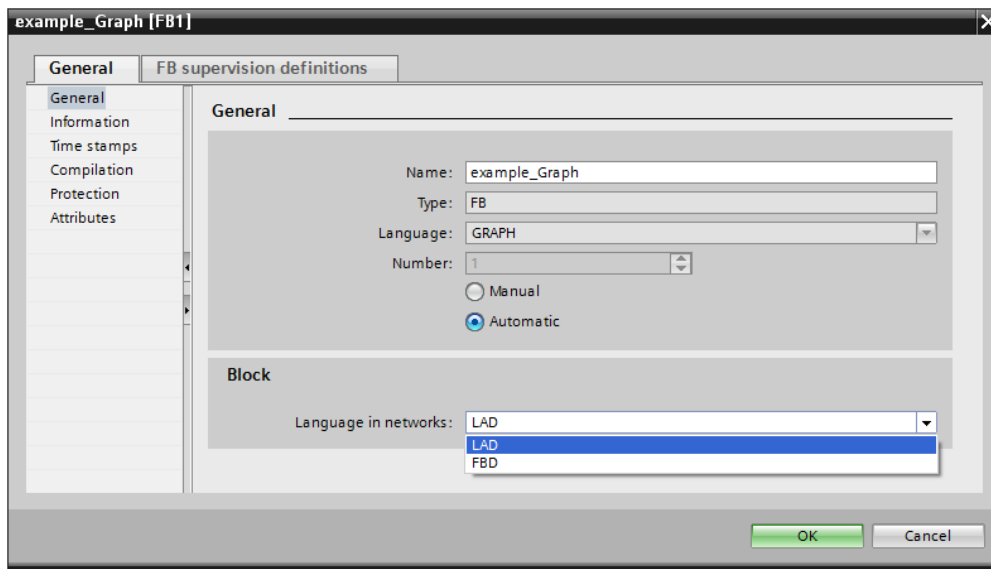
Merk op dat de andere twee counters, naam CTD en CTUD, (ook) andere parameters hebben. Zo heeft de CTD een LD-parameter waarmee de tellerwaarde op een *preset value* kan worden ingesteld. Hier ontbreekt echter de R-parameter.

## 8.14 Wijzigen van de programmeertaal in Graph

Natuurlijk is de taal Graph zelf niet om te zetten naar een andere taal. Het gaat hierbij om de taal die gebruikt wordt bij het programmeren van de overgangscondities. Die kan ingesteld worden op LAD of FBD.

Klik onder het kopje Program blocks met de rechter muisknop op de FB waarin het Graph-programma zich bevindt. Selecteer in het contextmenu de optie Properties. Er wordt nu een dialoog geopend waarin de taal te wijzigen in.

In het tabblad General is de optie General te vinden. Rechtsonder is dan de taal te wijzigen in LAD of FBD. Zie figuur 8.42.



**Figuur 8.42:** Selectie van de taal van de overgangscondities bij een Graph-programma.



# BIBLIOGRAFIE

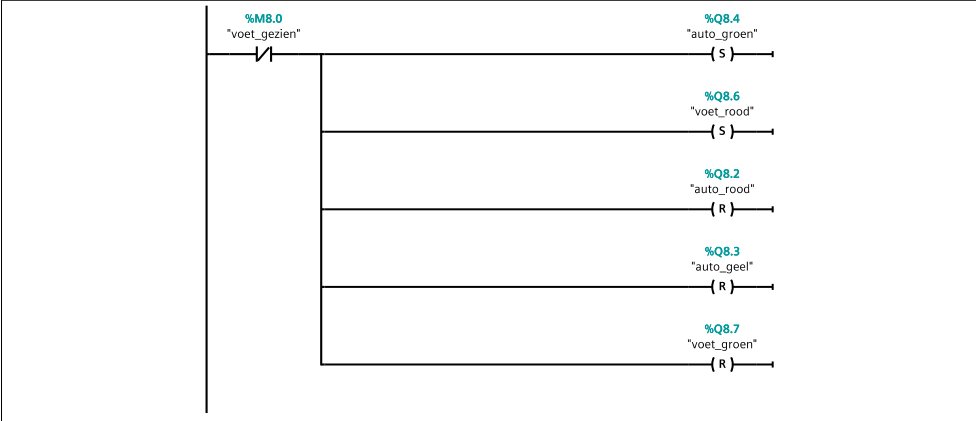
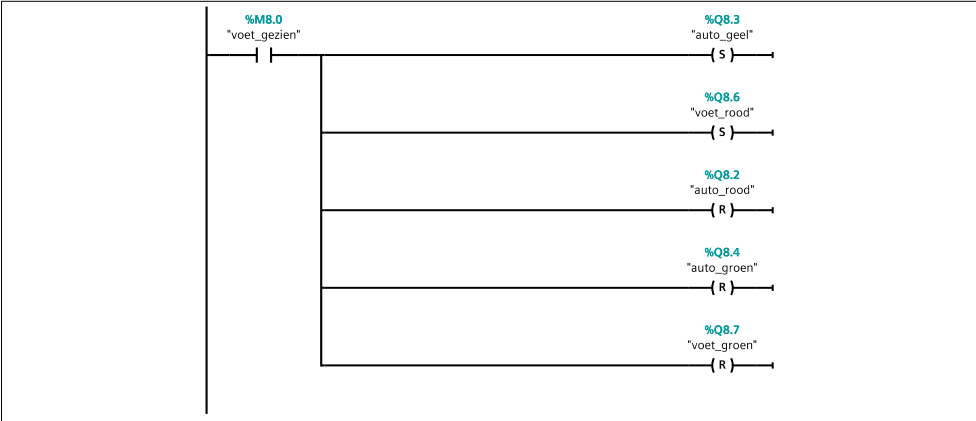
Veel materiaal is tegenwoordig (alleen) via Internet beschikbaar. Voorbeelden hiervan zijn de datasheets van ic's die alleen nog maar via de website van de fabrikant beschikbaar worden gesteld. Dat is veel sneller toegankelijk dan boeken en tijdschriften. De keerzijde is dat websites van tijd tot tijd veranderen of verdwijnen. De geciteerde weblinks werken dan niet meer. Helaas is daar niet veel aan te doen. Er is geen garantie te geven dat een weblink in de toekomst beschikbaar blijft.

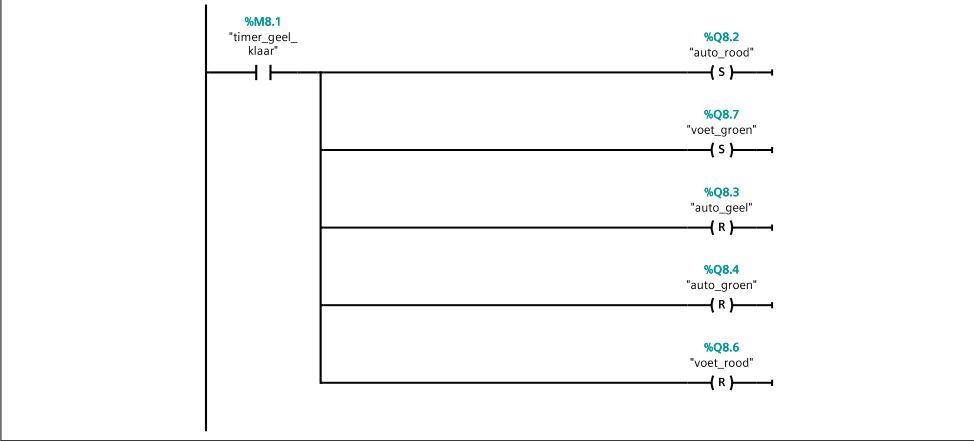
- [1] *Siemens support website*. URL: <https://support.industry.siemens.com/> (bezocht op 02-01-2018) (blz. 11).
- [2] H. Berger. *Automating with SIMATIC S7-300 inside TIA Portal*. 2de ed. Publicis Publishing, 2014. ISBN: 978-3-89578-443-9 (blz. 11).
- [3] Siemens. *S7-300 CPU 31xC and CPU 31x: Technical specifications*. Mrt 2011. URL: [https://cache.industry.siemens.com/dl/files/906/12996906/att\\_70325/v1/s7300\\_cpu\\_31xc\\_and\\_cpu\\_31x\\_manual\\_en-US\\_en-US.pdf](https://cache.industry.siemens.com/dl/files/906/12996906/att_70325/v1/s7300_cpu_31xc_and_cpu_31x_manual_en-US_en-US.pdf) (bezocht op 03-01-2018) (blz. 14).
- [4] Siemens. *SIMATIC S7-300, Module data*. Jun 2017. URL: [https://support.industry.siemens.com/cs/attachments/8859629/s7300\\_module\\_data\\_manual\\_en-US\\_en-US.pdf?download=true](https://support.industry.siemens.com/cs/attachments/8859629/s7300_module_data_manual_en-US_en-US.pdf?download=true) (bezocht op 02-01-2018) (blz. 14).
- [5] Betatherm. *Datasheet 10K3A542i*. 2018. URL: <http://www.farnell.com/datasheets/69441.pdf> (bezocht op 03-01-2018) (blz. 46).

# A. LAD-PROGRAMMA VERKEERSLICHTEN

Totally Integrated Automation Portal					
<b>tut_lad / PLC_1 [CPU 315F-2 PN/DP] / Program blocks</b> <b>verkeerslicht [FC1]</b>					
<b>verkeerslicht Properties</b>					
<b>General</b>					
<b>Name</b>	verkeerslicht	<b>Number</b>	1	<b>Type</b>	FC
<b>Language</b>	LAD	<b>Numbering</b>	Automatic		
<b>Information</b>					
<b>Title</b>	Zeer eenvoudig voetoversteekplaats met verkeerslichten.	<b>Author</b>	JodB	<b>Comment</b>	
<b>Family</b>		<b>Version</b>	0.1	<b>User-defined ID</b>	
<b>verkeerslicht</b>					
<b>Name</b>		<b>Data type</b>		<b>Supervision</b>	<b>Comment</b>
Input					
Output					
InOut					
Temp					
Constant					
▼ Return					
verkeerslicht	Void				
<b>Network 1: Voetganger gezien</b>					
Wil voetganger oversteken? Vang de actie in. Flankdetectie is nodig omdat voet_gezien naar 1 gaat als RESET van 1 naar 0 gaat en de SET-actie nog steeds actief is.					
<b>Symbol</b>	<b>Address</b>	<b>Type</b>	<b>Comment</b>		
"drukknop1"	%I8.0	Bool	Drukknop 1		
"drukknop2"	%I8.1	Bool	Drukknop 2		
"reset"	%I8.7	Bool	Reset systeem		
"timer_groen_klaar"	%M8.2	Bool	Timer voor groen klaar met tijdmeten		
"voet_gezien"	%M8.0	Bool	Voetganger gezien		
"voet_gezien_flank"	%M8.3	Bool	Flankdetector voet_gezien		
"voet_lampje"	%Q8.0	Bool	Voetganger gezien feedback lamp		

Totally Integrated Automation Portal																																														
<p><b>Network 2: Timer voor auto geel</b></p> <p>Eerste tijdmeting. Meet de tijd dat het gele autoverkeerslicht actief is en houdt voetgangerslicht op rood.</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> </div> <table border="1" style="width: 100%; border-collapse: collapse; margin: 10px 0;"> <thead> <tr> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Address</th> <th style="text-align: left;">Type</th> <th style="text-align: left;">Comment</th> </tr> </thead> <tbody> <tr> <td>"reset"</td> <td>%I8.7</td> <td>Bool</td> <td>Reset systeem</td> </tr> <tr> <td>"timer_geel"</td> <td>%T0</td> <td>Timer</td> <td>Timer voor auto geel</td> </tr> <tr> <td>"timer_geel_klaar"</td> <td>%M8.1</td> <td>Bool</td> <td>Timer voor geel klaar met tijdmeten</td> </tr> <tr> <td>"timer_groen_klaar"</td> <td>%M8.2</td> <td>Bool</td> <td>Timer voor groen klaar met tijdmeten</td> </tr> <tr> <td>"voet_gezien"</td> <td>%M8.0</td> <td>Bool</td> <td>Voetganger gezien</td> </tr> </tbody> </table> <p><b>Network 3: Timer voor auto groen</b></p> <p>Tweede tijdmeting. Meet de tijd dat het autoverkeerslicht rood is en het voetgangerslicht op groen. Hier zit een gevaarlijke constructie in. Timer_groen stuurt timer_groen_klaar aan, maar reset zichzelf dan ook na 1 scan cycle. Netter zou zijn om een extra timer te nemen die bv. 100 ms te laten draaien en die de andere timers te laten resetten.</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> </div> <table border="1" style="width: 100%; border-collapse: collapse; margin: 10px 0;"> <thead> <tr> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Address</th> <th style="text-align: left;">Type</th> <th style="text-align: left;">Comment</th> </tr> </thead> <tbody> <tr> <td>"reset"</td> <td>%I8.7</td> <td>Bool</td> <td>Reset systeem</td> </tr> <tr> <td>"timer_geel_klaar"</td> <td>%M8.1</td> <td>Bool</td> <td>Timer voor geel klaar met tijdmeten</td> </tr> <tr> <td>"timer_groen"</td> <td>%T1</td> <td>Timer</td> <td>Timer voor auto groen</td> </tr> <tr> <td>"timer_groen_klaar"</td> <td>%M8.2</td> <td>Bool</td> <td>Timer voor groen klaar met tijdmeten</td> </tr> </tbody> </table> <p><b>Network 4: Verkeerslicht auto groen</b></p> <p>Is voet_gezien gelijk aan 0? Dan hebben we autolicht groen en voetgangerslicht rood. Wederzijds uitgesloten met het volgende netwerk.</p>			Symbol	Address	Type	Comment	"reset"	%I8.7	Bool	Reset systeem	"timer_geel"	%T0	Timer	Timer voor auto geel	"timer_geel_klaar"	%M8.1	Bool	Timer voor geel klaar met tijdmeten	"timer_groen_klaar"	%M8.2	Bool	Timer voor groen klaar met tijdmeten	"voet_gezien"	%M8.0	Bool	Voetganger gezien	Symbol	Address	Type	Comment	"reset"	%I8.7	Bool	Reset systeem	"timer_geel_klaar"	%M8.1	Bool	Timer voor geel klaar met tijdmeten	"timer_groen"	%T1	Timer	Timer voor auto groen	"timer_groen_klaar"	%M8.2	Bool	Timer voor groen klaar met tijdmeten
Symbol	Address	Type	Comment																																											
"reset"	%I8.7	Bool	Reset systeem																																											
"timer_geel"	%T0	Timer	Timer voor auto geel																																											
"timer_geel_klaar"	%M8.1	Bool	Timer voor geel klaar met tijdmeten																																											
"timer_groen_klaar"	%M8.2	Bool	Timer voor groen klaar met tijdmeten																																											
"voet_gezien"	%M8.0	Bool	Voetganger gezien																																											
Symbol	Address	Type	Comment																																											
"reset"	%I8.7	Bool	Reset systeem																																											
"timer_geel_klaar"	%M8.1	Bool	Timer voor geel klaar met tijdmeten																																											
"timer_groen"	%T1	Timer	Timer voor auto groen																																											
"timer_groen_klaar"	%M8.2	Bool	Timer voor groen klaar met tijdmeten																																											

Totally Integrated Automation Portal																														
																														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Symbol</th> <th>Address</th> <th>Type</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td>"auto_geel"</td> <td>%Q8.3</td> <td>Bool</td> <td>Verkeerslicht auto geel</td> </tr> <tr> <td>"auto_groen"</td> <td>%Q8.4</td> <td>Bool</td> <td>Verkeerslicht auto groen</td> </tr> <tr> <td>"auto_rood"</td> <td>%Q8.2</td> <td>Bool</td> <td>Verkeerslicht auto rood</td> </tr> <tr> <td>"voet_gezien"</td> <td>%M8.0</td> <td>Bool</td> <td>Voetganger gezien</td> </tr> <tr> <td>"voet_groen"</td> <td>%Q8.7</td> <td>Bool</td> <td>Verkeerslicht voetganger groen</td> </tr> <tr> <td>"voet_rood"</td> <td>%Q8.6</td> <td>Bool</td> <td>Verkeerslicht voetganger rood</td> </tr> </tbody> </table>	Symbol	Address	Type	Comment	"auto_geel"	%Q8.3	Bool	Verkeerslicht auto geel	"auto_groen"	%Q8.4	Bool	Verkeerslicht auto groen	"auto_rood"	%Q8.2	Bool	Verkeerslicht auto rood	"voet_gezien"	%M8.0	Bool	Voetganger gezien	"voet_groen"	%Q8.7	Bool	Verkeerslicht voetganger groen	"voet_rood"	%Q8.6	Bool	Verkeerslicht voetganger rood		
Symbol	Address	Type	Comment																											
"auto_geel"	%Q8.3	Bool	Verkeerslicht auto geel																											
"auto_groen"	%Q8.4	Bool	Verkeerslicht auto groen																											
"auto_rood"	%Q8.2	Bool	Verkeerslicht auto rood																											
"voet_gezien"	%M8.0	Bool	Voetganger gezien																											
"voet_groen"	%Q8.7	Bool	Verkeerslicht voetganger groen																											
"voet_rood"	%Q8.6	Bool	Verkeerslicht voetganger rood																											
<p><b>Network 5: Verkeerslicht auto geel</b></p> <p>Is voet_gezien gelijk aan 1? Dan hebben we autolicht geel en voetgangerslicht rood. Wederzijds uitgesloten met vorig netwerk.</p>																														
																														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Symbol</th> <th>Address</th> <th>Type</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td>"auto_geel"</td> <td>%Q8.3</td> <td>Bool</td> <td>Verkeerslicht auto geel</td> </tr> <tr> <td>"auto_groen"</td> <td>%Q8.4</td> <td>Bool</td> <td>Verkeerslicht auto groen</td> </tr> <tr> <td>"auto_rood"</td> <td>%Q8.2</td> <td>Bool</td> <td>Verkeerslicht auto rood</td> </tr> <tr> <td>"voet_gezien"</td> <td>%M8.0</td> <td>Bool</td> <td>Voetganger gezien</td> </tr> <tr> <td>"voet_groen"</td> <td>%Q8.7</td> <td>Bool</td> <td>Verkeerslicht voetganger groen</td> </tr> <tr> <td>"voet_rood"</td> <td>%Q8.6</td> <td>Bool</td> <td>Verkeerslicht voetganger rood</td> </tr> </tbody> </table>	Symbol	Address	Type	Comment	"auto_geel"	%Q8.3	Bool	Verkeerslicht auto geel	"auto_groen"	%Q8.4	Bool	Verkeerslicht auto groen	"auto_rood"	%Q8.2	Bool	Verkeerslicht auto rood	"voet_gezien"	%M8.0	Bool	Voetganger gezien	"voet_groen"	%Q8.7	Bool	Verkeerslicht voetganger groen	"voet_rood"	%Q8.6	Bool	Verkeerslicht voetganger rood		
Symbol	Address	Type	Comment																											
"auto_geel"	%Q8.3	Bool	Verkeerslicht auto geel																											
"auto_groen"	%Q8.4	Bool	Verkeerslicht auto groen																											
"auto_rood"	%Q8.2	Bool	Verkeerslicht auto rood																											
"voet_gezien"	%M8.0	Bool	Voetganger gezien																											
"voet_groen"	%Q8.7	Bool	Verkeerslicht voetganger groen																											
"voet_rood"	%Q8.6	Bool	Verkeerslicht voetganger rood																											

Totally Integrated Automation Portal																														
<b>Network 6: Verkeerslicht auto rood</b>																														
Hebben we timer_geel_klaar is 1? Dan hebben we autolicht rood en voetgangerslicht groen. Als timer_geel_klaar is 1, dan worden alle acties van de vorige twee netwerken ongedaan gemaakt. Dat komt omdat de PLC z'n uitgangen pas aan het eind van de huidige scan cycle aanpast (bij Siemens eigenlijk aan het begin, maar goed).																														
																														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e0e0e0;"> <th>Symbol</th> <th>Address</th> <th>Type</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td>"auto_geel"</td> <td>%Q8.3</td> <td>Bool</td> <td>Verkeerslicht auto geel</td> </tr> <tr> <td>"auto_groen"</td> <td>%Q8.4</td> <td>Bool</td> <td>Verkeerslicht auto groen</td> </tr> <tr> <td>"auto_rood"</td> <td>%Q8.2</td> <td>Bool</td> <td>Verkeerslicht auto rood</td> </tr> <tr> <td>"timer_geel_klaar"</td> <td>%M8.1</td> <td>Bool</td> <td>Timer voor geel klaar met tijdmeten</td> </tr> <tr> <td>"voet_groen"</td> <td>%Q8.7</td> <td>Bool</td> <td>Verkeerslicht voetganger groen</td> </tr> <tr> <td>"voet_rood"</td> <td>%Q8.6</td> <td>Bool</td> <td>Verkeerslicht voetganger rood</td> </tr> </tbody> </table>			Symbol	Address	Type	Comment	"auto_geel"	%Q8.3	Bool	Verkeerslicht auto geel	"auto_groen"	%Q8.4	Bool	Verkeerslicht auto groen	"auto_rood"	%Q8.2	Bool	Verkeerslicht auto rood	"timer_geel_klaar"	%M8.1	Bool	Timer voor geel klaar met tijdmeten	"voet_groen"	%Q8.7	Bool	Verkeerslicht voetganger groen	"voet_rood"	%Q8.6	Bool	Verkeerslicht voetganger rood
Symbol	Address	Type	Comment																											
"auto_geel"	%Q8.3	Bool	Verkeerslicht auto geel																											
"auto_groen"	%Q8.4	Bool	Verkeerslicht auto groen																											
"auto_rood"	%Q8.2	Bool	Verkeerslicht auto rood																											
"timer_geel_klaar"	%M8.1	Bool	Timer voor geel klaar met tijdmeten																											
"voet_groen"	%Q8.7	Bool	Verkeerslicht voetganger groen																											
"voet_rood"	%Q8.6	Bool	Verkeerslicht voetganger rood																											

## B. FUNCTIE FC1 "ConvertADCToTemp"

In deze bijlage is de code afgebeeld van de functie ConvertADCToTemp. Maak een tekstbestand aan met een ASCII-editor zoals Notepad. Kopieer de code van deel 1 en deel 2 in het tekstbestand. Verander de extensie van .txt in .scl. De code kan via de importfunctie van TIA Portal in het project geladen worden. Zie daarvoor paragraaf 8.8.

```

1 FUNCTION "ConvertADCToTemp" : Real
2 TITLE = Convert ADC Value to Temperature
3 { S7_Optimized_Access := 'FALSE' }
4 AUTHOR : JodB
5 FAMILY : TEMP
6 NAME : TCONVERT
7 VERSION : 1.0
8 //
9 // This routine converts the raw temperature data from the self
10 // made NTC temp sensor to the temp in degree C.
11 //
12 //      +24 V          The transfer function for Vout is:
13 //      |
14 //      |
15 //      +-+          Vout = Z * 24 V
16 //      | |
17 //      Rs | |       where
18 //      | |          Rntcp
19 //      +-+          Z = -----
20 //      |            Rs + Rntcp
21 //      +-----+----- Vout
22 //      | |          where
23 //      +-+ +-+
24 //      | | | |      Rntcp = {A * exp (B/T)} // Rin
25 // Rin | | | | Rntc
26 //      | | | |
27 //      +-+ +-+     with A and B being parameters of Rntc,
28 //      | |       and Rin being the input resistance of
29 //      +-----+   the analog input.
30 //      |
31 //      ===
32 //
33 // The tranfer function has to be rewritten to a function with
34 // T explicit. Note that T is in Kelvin.
35 // The raw ADC value has a maximum of C600 hex at 10.0 V and
36 // a minimum of 0000 hex at 0.0 V.
37 // Please note that with a high power supply voltage, the self
38 // heating effect of the NTC is considerable.

```

Listing B.1: Code van de functie ConvertADCToTemp (deel 1).

```

1  VAR_INPUT
2      ADC_VALUE : Word;    // ADC raw data
3  END_VAR
4
5  VAR_OUTPUT
6      VNTC : Real;    // Measured voltage across the NTC
7      INTC : Real;    // Calculated current through the NTC
8      RNTC : Real;    // Calculated value of the NTC
9      PNTC : Real;    // Calculated power of the NTC
10 END_VAR
11
12 VAR_TEMP
13     Z : Real;    // Transfer function of the voltage divider
14     ZI : Int;    // For conversion from Word to Int
15 END_VAR
16
17 VAR CONSTANT
18     VPOWER : Real := 24.0;    // Power supply voltage
19     VANALOG_MAX : Real := 10.0;    // Maximum voltage on the ADC input
20     ADCVALUE_AT_VANALOG_MAX : Int := 27648;    // ADC value at maximum ADC voltage
21     A : Real := 0.02010974;    // A parameter of the NTC
22     B : Real := 3899.387;    // B (beta) parameter of the NTC
23     RS : Real := 47000.0;    // Series resistor
24     RIN : Real := 100000.0;    // Input resistance of the ADC input
25     K_TO_C : Real := 273.15;    // Conversion from Kelvin to Celsius
26     DISSIPATION_CONSTANT : Real := 0.002;    // Dissipation constant
27 END_VAR
28
29
30 BEGIN
31     // ADC value to integer
32     #ZI := WORD_TO_INT(#ADC_VALUE);
33
34     // Voltage measured
35     #VNTC := (1.0 * #ZI) / #ADCVALUE_AT_VANALOG_MAX * #VANALOG_MAX;
36
37     // Calculate Z as function of ...
38     #Z := #VNTC / #VPOWER;
39
40     // Rntc in Ohms. Function found with the aid of Maple 10.
41     #RNTC := - (#VNTC * #RS * #RIN / (- #VPOWER * #RIN + #VNTC * #RIN + #VNTC * #RS));
42
43     // Calculate current through the NTC
44     #INTC := #VNTC / #RNTC;
45
46     // Calculate power dissipated by the NTC
47     #PNTC := #VNTC * #INTC;
48
49     // Temperature in degree C. Function found with the aid of Maple 10. This is the
50     // return value of the function. There is compensation for the self-heating effect
51     #ConvertADCToTemp := #B / LN(#Z * #RS * #RIN / (#A * (#RIN - #Z * #RS - #Z * #RIN)))
52     - #K_TO_C - #PNTC / #DISSIPATION_CONSTANT;
53 END_FUNCTION

```

Listing B.2: Code van de functie ConvertADCToTemp (deel 2).

## C. FUNCTIE FC2 "checkMinMax"

In deze bijlage is de code afgebeeld van de functie CheckMinMax. Maak een tekstbestand aan met een ASCII-editor zoals Notepad. Kopieer de code in het tekstbestand. Verander de extensie van .txt in .scl. De code kan via de importfunctie van TIA Portal in het project geladen worden. Zie daarvoor paragraaf 8.8.

```

1 FUNCTION "CheckMinMax" : Void
2 TITLE = Check Minimum and Maximum Temps
3 { S7_Optimized_Access := 'FALSE' }
4 AUTHOR : JodB
5 FAMILY : TEMP
6 NAME : CMM
7 VERSION : 1.0
8   VAR_INPUT
9     C_TEMPTOOHIGH : Real;
10    C_TEMPTOOLOW : Real;
11    TEMPERATURE : Real;
12  END_VAR
13
14  VAR_OUTPUT
15    TEMPTOOHIGH : Bool;
16    TEMPTOOLOW : Bool;
17  END_VAR
18
19
20 BEGIN
21   IF #TEMPERATURE > #C_TEMPTOOHIGH THEN // If current temp too high...
22     #TEMPTOOHIGH := true;           // signal it.
23   ELSE
24     #TEMPTOOHIGH := false;
25   END_IF;
26
27   IF #TEMPERATURE < #C_TEMPTOOLOW THEN // If current temp too low...
28     #TEMPTOOLOW := true;           // signal it.
29   ELSE
30     #TEMPTOOLOW := false;
31   END_IF;
32
33 END_FUNCTION

```

Listing C.1: Code van de functie CheckMinMax.



## D. FUNCTIE FC3 "Alarms"

In deze bijlage is de code afgebeeld van de functie Alarms. Maak een tekstbestand aan met een ASCII-editor zoals Notepad. Kopieer de code in het tekstbestand. Verander de extensie van .txt in .scl. De code kan via de importfunctie van TIA Portal in het project geladen worden. Zie daarvoor paragraaf 8.8.

```
1 FUNCTION "Alarms" : Void
2 TITLE = Alarms holding relay
3 { S7_Optimized_Access := 'FALSE' }
4 AUTHOR : JodB
5 FAMILY : TEMP
6 NAME : HOLD
7 VERSION : 1.0
8   VAR_INPUT
9     TEMPTOOHIGH : Bool;
10    TEMPTOOLOW : Bool;
11    RESEALARMS : Bool;
12  END_VAR
13
14  VAR_IN_OUT
15    ALARMSHOLDING : Bool;
16  END_VAR
17
18
19 BEGIN
20   // This is the code of the alarms holding relay. If an alarm is triggered
21   // the holding relay will be energized. If the alarms will disappear, the
22   // alarms holding relay will still be energized.
23   // The holding relay can be reset with a reset.
24
25   IF #RESEALARMS THEN
26     #ALARMSHOLDING := FALSE;
27   ELSIF #TEMPTOOHIGH OR #TEMPTOOLOW THEN
28     #ALARMSHOLDING := TRUE;
29   END_IF;
30
31
32 END_FUNCTION
```

Listing D.1: Code van de functie Alarms.

## E. FUNCTIEBLOK FB1 "AverageTemp"

In deze bijlage is de code afgebeeld van het functieblok AverageTemp. Maak een tekstbestand aan met een ASCII-editor zoals Notepad. Kopieer de code van deel 1 en deel 2 in het tekstbestand. Verander de extensie van .txt in .scl. De code kan via de importfunctie van TIA Portal in het project geladen worden. Zie daarvoor paragraaf 8.8.

Merk op dat een functieblok gebruikt maakt van een Instance Data Block. Bij compilatie van een functieblok wordt niet gevraagd welk datablok moet worden aangemaakt en gekoppeld. Dat gebeurt pas bij het aanmaken van een aanroep van het functieblok vanuit een ander blok, bijvoorbeeld OB1.

```

1 FUNCTION_BLOCK "AverageTemp"
2 TITLE = Calculate the average temperature over 10 entries
3 { S7_Optimized_Access := 'FALSE' }
4 AUTHOR : JodB
5 FAMILY : TEMP
6 NAME : AVERAGE
7 VERSION : 1.0
8   VAR_INPUT
9     RECORD_TEMP : Bool; // Flag to signal record of temperature
10    CURRENT_TEMP : Real; // The temperature
11  END_VAR
12
13  VAR_OUTPUT
14    AVERAGE_TEMP : Real; // The average temperature
15  END_VAR
16
17  VAR
18    RECORD_TEMP_EDGE : Bool; // Edge flag for RECORD_TEMP
19    INDEX : Int := 1; // The current index in the array
20    TEMP : Array[1..10] of Struct // The temperatures and valid flags
21      temperature : Real;
22      valid : Bool;
23    END_STRUCT;
24  END_VAR
25
26  VAR_TEMP
27    LOOP_INDEX : Int; // Loop index to visit elements in array
28    CUMU_TEMP : Real; // Sum of all the recorded temperatures
29    NUMBER_VALID : Int; // Number of valid entries
30  END_VAR

```

Listing E.1: Code van het functieblok AverageTemp (deel 1).

```

1
2
3 BEGIN
4 // This routine records temperatures in an array and calculates the average
5 // temperature over all valid entries. For recording a temperature, an edge
6 // is detected on input RECORD_TEMP. The entries are recorded in an array
7 // used as a circular buffer.
8
9 // Should we record once?
10 IF #RECORD_TEMP AND NOT #RECORD_TEMP_EDGE THEN
11 // Do record it
12 #TEMP[#INDEX].temperature := #CURRENT_TEMP;
13 #TEMP[#INDEX].valid := TRUE;
14 // Advance the index
15 #INDEX := #INDEX + 1;
16 // But if it is past the end, flip it to the begin
17 IF #INDEX > 10 THEN
18 #INDEX := 1;
19 END_IF;
20 END_IF;
21 // Update edge relay
22 #RECORD_TEMP_EDGE := #RECORD_TEMP;
23
24 // Make sure we start at 0.0
25 #CUMU_TEMP := 0.0;
26 // Count the number of valid entries
27 #NUMBER_VALID := 0;
28 // Visit all array items and sum them
29 FOR #LOOP_INDEX := 1 TO 10 BY 1 DO
30 IF #TEMP[#LOOP_INDEX].valid THEN
31 #CUMU_TEMP := #CUMU_TEMP + #TEMP[#LOOP_INDEX].temperature;
32 #NUMBER_VALID := #NUMBER_VALID + 1;
33 END_IF;
34 END_FOR;
35
36 // Calculate average, but note that if there are no valid entries,
37 // we would get and division by zero error. So give back an
38 // unrealistic average temperature
39 IF #NUMBER_VALID = 0 THEN
40 #AVERAGE_TEMP := -273.15;
41 ELSE
42 #AVERAGE_TEMP := #CUMU_TEMP / #NUMBER_VALID;
43 END_IF;
44
45 END_FUNCTION_BLOCK

```

Listing E.2: Code van het functieblok AverageTemp (deel 2).

## F. OB35 "CyclicInterrupt"

In deze bijlage is de code afgebeeld van de Organization Block CyclicInterrupt. Maak een tekstbestand aan met een ASCII-editor zoals Notepad. Kopieer de code in het tekstbestand. Verander de extensie van .txt in .scl. De code kan via de importfunctie van TIA Portal in het project geladen worden. Zie daarvoor paragraaf 8.8. Let op: TIA Portal importeert dit bestand als OB0. Dit nummer is te wijzigen in de Properties van het blok.

```

1 ORGANIZATION_BLOCK "CyclicInterrupt"
2 TITLE = "Cyclic Interrupt"
3 { S7_Optimized_Access := 'FALSE' }
4 AUTHOR : JodB
5 FAMILY : TEMP
6 NAME : CYCINT
7 VERSION : 1.0
8   VAR_TEMP
9     OB35_EV_CLASS : Byte;    // Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class
10      1)
11     OB35_STRT_INF : Byte;    // 16#36 (OB 35 has started)
12     OB35_PRIORITY : Byte;    // Priority of OB Execution
13     OB35_OB_NUMBR : Byte;    // 35 (Organization block 35, OB35)
14     OB35_RESERVED_1 : Byte;  // Reserved for system
15     OB35_RESERVED_2 : Byte;  // Reserved for system
16     OB35_PHASE_OFFSET : Word; // Phase offset (msec)
17     OB35_RESERVED_3 : Int;    // Reserved for system
18     OB35_EXC_FREQ : Int;     // Frequency of execution (msec)
19     OB35_DATE_TIME : Date_And_Time; // Date and time OB35 started
20   END_VAR
21
22 BEGIN
23   // Check if counter is expired (1 minute)
24   IF "CYCLIC_INTERRUPT_COUNTER" = 599 THEN
25     // If so, set RECORD_TEMP and start counting from 0
26     "RECORD_TEMP" := TRUE;
27     "CYCLIC_INTERRUPT_COUNTER" := 0;
28   ELSE
29     // If not, reset RECORD_TEMP and update counter
30     "RECORD_TEMP" := FALSE;
31     "CYCLIC_INTERRUPT_COUNTER" := "CYCLIC_INTERRUPT_COUNTER" + 1;
32   END_IF;
33
34 END_ORGANIZATION_BLOCK

```

Listing F.1: Code van Organization Block CyclicInterrupt.

# G. PLC BLOCKS, ADRESSEN, MEMORY

De PLC beschikt over een aantal blocks, waarin programma en data gestructureerd kan worden.

## Organization Blocks (OB)

OB's zijn de interfaces tussen het OS en het gebruikersprogramma. De belangrijkste OB is OB1: cyclic operation block, voer het programma "steeds" uit. Daarnaast zijn er tal van andere OB's. Dus: om een programma draaiend te krijgen moet ALTIJD OB1 gebruikt worden, ook al doet deze niets anders dan FB's of FC's aanroepen.

## Function Blocks (FB)

Dit zijn blokken met parameteroverdrachtmogelijkheden en eigen geheugen. Dit geheugen voor dataopslag ligt in een DB en wordt persistent genoemd, dat wil zeggen dat een FB zijn eigen data kan bewaren over aanroepen heen. Voor C-programmeurs is het te vergelijken met static variabelen.

## Functions (FC)

Dit zijn blokken die ook parameteroverdrachtmogelijkheden hebben, maar geen eigen geheugen. Nadat een functie klaar is, wordt alle data gewist. Typische voorbeelden zijn het berekenen van de wortel van een getal, grootste gemene deler. Functions kunnen waarden retourneren.

## Data Blocks (DB)

Dit zijn de blokken waarin de data wordt opgeslagen. Ze kunnen maximaal 16420 bytes groot zijn.

## System Function (SFC), System Function Blocks (SFB)

Deze zijn als FC en FB, maar nu met voorgeprogrammeerde functies, bijvoorbeeld communicatie over de bus. Deze zijn in de firmware van de PLC opgenomen.

## Input- en outputadressen geconfigureerde PLC

De digitale ingangen lopen van I0.0 t/m I1.7. Hierin zijn 0 en 1 het input adresbyte en .0 en .7 het bitnummer binnen een adres. Er zijn in totaal 16 ingangen. De digitale uitgangen lopen van Q0.0 t/m Q1.7. Hierin zijn 0 en 1 het output adresbyte en .0 en .7 het bitnummer binnen een adres. Er zijn in totaal 16 uitgangen. De analoge ingangen zijn PIW272 t/m PIW278. De uitgangen zijn te vinden op PQW272 en PQIW274. Merk op dat het om PI en PQ gaat (buiten de process image table om) en dat het words zijn. Bv. PIW278 bestaat uit byte 278 en 279. De ingangen van de simulatiemodule lopen van I8.0 t/m I8.7. De uitgangen lopen van Q8.0 t/m Q8.7

## Memory

Het geheugen loopt van M0.0 tot M2047.7. Het geheugen is ook byte-, word- en long wordadreseerbaar. Het is een algemeen geheugen dat gedeeld wordt met andere OB's, FB's en FC's.

## H. FORMULES TEMPERATUURMETING

Een NTC-weerstand heeft een weerstandswaarde die aardig benaderd wordt met de formule:

$$R_{NTC} = A \cdot e^{\frac{B}{T}} \quad (\text{H.1})$$

$A$  en  $B$  zijn hierin parameters van de NTC en  $T$  de temperatuur in Kelvin.

De overdracht van de spanningsdeler uit figuur 6.1 is als volgt te berekenen:

$$Z = \frac{V_{NTC}}{V_{POWER}} = \frac{R_{NTC} // R_{IN}}{R_S + R_{NTC} // R_{IN}} \quad (\text{H.2})$$

Hierin is  $V_{POWER}$  de spanning van de voeding,  $V_{NTC}$  de spanning over de NTC,  $R_{NTC}$  de weerstand van de NTC,  $R_{IN}$  de ingangsweerstand van de analoge ingang en  $R_S$  de serieweerstand. Invullen van (H.1) en uitwerken levert de functie:

$$Z = \frac{R_{IN} \cdot A \cdot e^{\frac{B}{T}}}{R_S \cdot R_{IN} + R_S \cdot A \cdot e^{\frac{B}{T}} + R_{IN} \cdot A \cdot e^{\frac{B}{T}}} \quad (\text{H.3})$$

Hieruit is  $T$  (in Kelvin) te herleiden:

$$T = \frac{B}{\ln \left( \frac{Z \cdot R_S \cdot R_{IN}}{A \cdot (R_{IN} - Z \cdot R_S - Z \cdot R_{IN})} \right)} \quad (\text{H.4})$$

$V_{NTC}$  is echter ook gelijk aan:

$$V_{NTC} = \frac{ADC_{VALUE}}{ADC_{VALUE\_MAX}} \cdot V_{ANALOG\_MAX} \quad (\text{H.5})$$

Hierin is  $ADC_{VALUE}$  de huidige waarde van de ADC,  $ADC_{VALUE\_MAX}$  de ADC-waarde bij de maximale analoge ingangsspanning en  $V_{ANALOG\_MAX}$  de maximale analoge ingangsspanning. De overdracht  $Z$  is dan eenvoudig te bepalen uit het eerste deel van (H.2) en in te vullen in (H.4).

Formule H.2 is om te werken zodat  $R_{NTC}$  bepaald kan worden:

$$R_{NTC} = \frac{V_{NTC} \cdot R_S \cdot R_{IN}}{V_{POWER} \cdot R_{IN} - V_{IN} \cdot R_S - V_{NTC} \cdot R_{IN}} \quad (\text{H.6})$$

Dan is ook de stroom  $I_{NTC}$  door de NTC te bepalen:

$$I_{NTC} = \frac{V_{NTC}}{R_{NTC}} \quad (\text{H.7})$$

En het gedissipeerde vermogen is:

$$P_{NTC} = V_{NTC} \cdot I_{NTC} \quad (\text{H.8})$$

De temperatuur  $T$  is ook als volgt te berekenen nu  $R_{NTC}$  bekend is:

$$T = \frac{B}{\ln R_{NTC} - \ln A} \quad (\text{H.9})$$

De waarde van  $A$  is trouwens te *benaderen* door:

$$A = R_{25} \cdot e^{\frac{B_{25}}{T_{25}}} \quad (\text{H.10})$$

Hierin is  $R_{25}$  de weerstandswaarde van de NTC,  $B_{25}$  de betawaarde en  $T_{25}$  de temperatuur in K bij 25 °C. In de praktijk is het beter om  $A$  en  $B$  via *curve fitting* technieken te bepalen.

De NTC is een weerstand en weerstanden dissiperen vermogen. Daardoor warmen ze op. Zo ook dus de NTC. De dissipatieconstante  $DC$  geeft aan hoeveel vermogen wordt gedissipeerd bij een stijging van 1 °C. De temperatuurstijging als gevolg van de zelfopwarming is:

$$T_{self} = \frac{P_{NTC}}{DC} \quad (\text{H.11})$$

Deze temperatuur moet dus afgetrokken worden van de berekende temperatuur in (H.4) en (H.9).

De volgende waarden<sup>9</sup> voor de meetopstelling zijn:

$A = 0,020109745 \Omega$	$V_{POWER} = 24,0 \text{ V}$
$B = 3899,3874 \text{ K}$	$V_{ANALOG\_MAX} = 10,0 \text{ V}$
$R_S = 47.000 \Omega$ (nominaal)	$ADC_{VALUE\_MAX} = 27648$
$R_{IN} = 100.000 \Omega$ (nominaal)	$^{\circ}\text{C} = \text{K} -273,15$
$DC = 2,0 \text{ mW}/^{\circ}\text{C}$	

Bij het ontbreken van de NTC (of defect) zal de spanning op de analoge ingang boven de “overflow range” stijgen. De spanning is dan:

$$V_{ANALOG\_IN} = \frac{R_{IN}}{R_S + R_{IN}} \cdot V_{POWER} = \frac{100.000}{147.000} \cdot 24 = 16,33 \text{ V} \quad (\text{H.12})$$

De maximale spanning op een analoge ingang mag 20 V bedragen.

---

<sup>9</sup>  $A$  en  $B$  bepaald via curve fitting.