

State machines : van functionele beschrijving tot gestructureerd programma in SCL d.m.v. TIA Portal.

Inhoudsopgave

State machines : van functionele beschrijving tot gestructureerd programma in SCL d.m.v. TIA Portal.	1
1 Inleiding	2
2 De functionele beschrijving (probleemstelling)	3
3 State machine.....	3
3.1 Syntax en terminologie van een state machine	4
3.2 Gebruik van tijdfuncties (timers) in een state machine	5
3.3 Vanuit een state meerder andere states gelijktijdig activeren	7
3.4 Gebruik van flankdetectie in een state machine	7
4 Van state machine naar een SCL-programma in TIA Portal: de richtlijnen	8
4.1 Keuze en benaming van de gebruikte PLC-variabelen	9
4.2 Opbouw/volgorde en structuur van het SCL-programma	9
4.3 Voorbeelden van een SCL-programma volgens state machine	10
4.3.1 Teleruptorschakeling (gebruik flankdetectie, state machine van p. 7) ..	10
4.3.2 Gebruik van TON-timers (state machine van p. 5)	11
4.3.3 Programmeren van flankdetecties	12
5 Opgaven	13
6 Tekenen van een state machine op PC	15

1 Inleiding

Het aanvatten van een PLC-project begint bij het goed begrijpen en analyseren van de probleemstelling. Hoe werkt de machine/proces precies ? Wat moet er gebeuren tijdens het proces ? Hoe verloopt het proces ?

Hierbij komt heel wat kijken, omdat bij de bediening van een installatie er meestal ook meerdere modi zijn. Naast het automatisch verlopen van het proces, is er dikwijls ook een manuele modus. Hier kan de operator stap per stap, op zijn eigen initiatief, het procesverloop bepalen. Dikwijls is er ook een servicemodus, waarbij bv. bepaalde bewegingen enkel aan verlaagde snelheid uitgevoerd kunnen worden. Al deze modi kunnen bv. via een keuzeschakelaar of HMI-panel geselecteerd worden.

Naast de werkingsmodi, dienen ook alarmen en/of andere procesgebeurtenissen door het programma verwerkt te worden. Tot slot zijn er ook nog bv. de communicatie met het operator panel en testprocedures die via het programma verwerkt dienen te worden. Dus een PLC-project is in de praktijk een complex gebeuren.

Een zogenaamde functionele beschrijving (FDS of Functional Design Specification) is een document (kan soms een klein boek zijn) waar alle functies beschreven worden, die men later door het PLC-systeem wil uitgevoerd zien. Op basis hiervan kan de PLC-programmeur later dan aan de slag.

Belangrijk achteraf is ook dat de software goed gestructureerd is ontworpen en alles ook degelijk gedocumenteerd wordt. Immers, het is meer regel dan uitzondering, dat een programma achteraf nog geoptimaliseerd of zelfs uitgebreid dient te worden. Bij slecht ontworpen / gedocumenteerde programma's is dit een aanzienlijke extra tijd die vermeden had kunnen worden...

In de praktijk gebeurt het ook dikwijls dat de persoon die een bepaald PLC-programma ontworpen heeft, niet meer op het bedrijf aanwezig is en iemand anders ermee aan de slag moet. Op dat moment kan het zeer zinvol zijn dat de "gedachtegang" van de programmeur ergens voorhanden is. Die 'gedachtegang' of de ontwerpstructuur van het programma kan bv. één of ander softwaremodel zijn (flowchart, grafcet, state machine,...).

Ook het 'debuggen' (fout-zoeken) van een programma is een belangrijke factor in de praktijk. En dit kan bij slecht opgebouwde/gestructureerde programma's voor serieuze tijdsvertragingen zorgen !

Het softwaremodel dat hier besproken zal worden is een 'state machine'. Het is voor een stuk vergelijkbaar met een ander softwaremodel dat in het OPO 'PLC project' aan bod komt, nl. 'grafcet'. Deze modellen worden vooral toegepast voor sequentieel verlopende systemen, m.a.w. systemen die volgens een vast patroon, stap na stap, uitgevoerd worden.

In de volgende paragrafen zal aan de hand van een voorbeeld heel de keten beschreven worden, om vertrekende van een functionele beschrijving een gestructureerd programma, op basis van een state machine te ontwerpen.

Het PLC-programma wordt dan ontworpen in TIA Portal m.b.v. de programmeertaal SCL.

2 De functionele beschrijving (probleemstelling)

Hieronder volgt een functionele beschrijving van een eenvoudig proces waar enkele ledjes dienen aangestuurd te worden via een **bepaalde volgorde** en d.m.v. **specifieke voorwaarden**.

Functionele beschrijving:

Bij opstart van het systeem gaat Lre1 aan.

Gelijktijdig dient in iDisplay1 de waarde 10 weergegeven te worden.

Als de NO-drukknop S1 ingedrukt wordt, gaat Lre1 uit en Lor1 aan.

Gelijktijdig dient in iDisplay1 de waarde 20 weergegeven te worden.

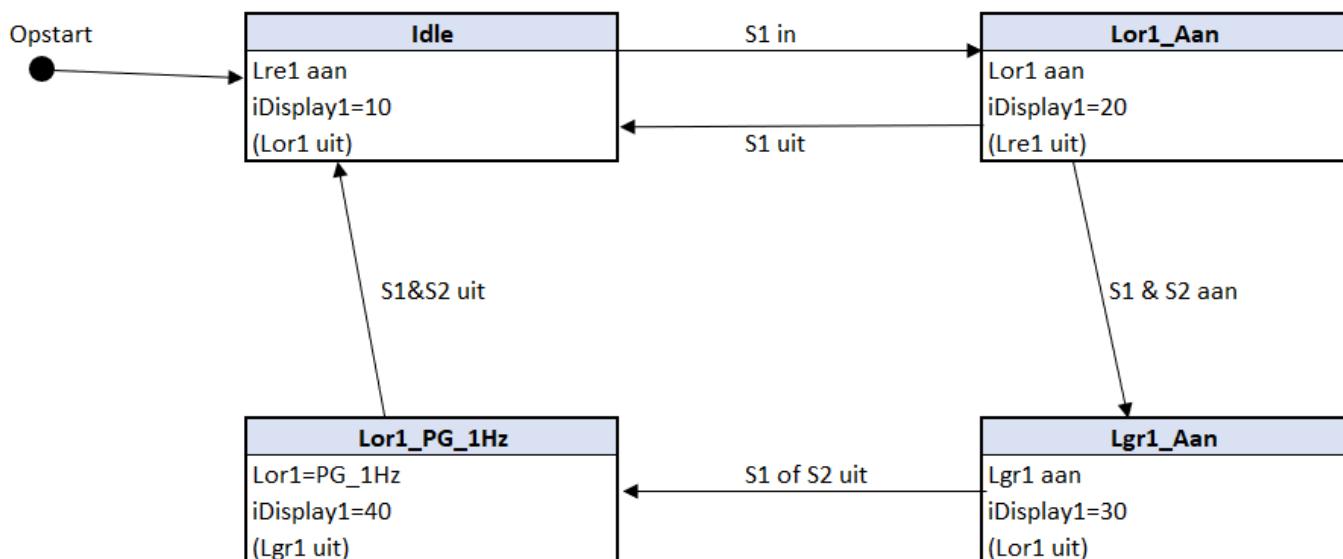
Wordt S1 echter terug losgelaten dooft Lor1 en gaat Lre1 terug aan ($iDisplay1=10$).

Indien S1 ingedrukt blijft en tegelijkertijd wordt S2 (NO-drukknop) ingedrukt, moeten Lor1 uit en Lgr1 aan gaan. In iDisplay1 verschijnt de waarde 30.

Vanaf dat één van de 2 drukknoppen losgelaten wordt: Lgr1 uit en Lor1 begint te knipperen ($f=1\text{Hz}$). In iDisplay1 komt 40. Daarna moet de andere drukknop eveneens losgelaten worden om het systeem terug in de beginstate te brengen, waarna terug kan herstart worden via S1.

3 State machine

In onderstaande figuur wordt de state machine weergegeven van de bovenstaande probleemstelling (functionele beschrijving).



3.1 Syntax en terminologie van een state machine

- **Zwart bolletje met pijl** is **opstart** van het systeem
Dit zal dus overeenkomen met opstart van de PLC (Stop-Run of PowerOn)
- De 4 **rechthoeken** zijn de zogenaamde '**states**'. Ze stellen steeds een bepaalde toestand of moment voor waarin het systeem zich op dat moment bevindt.
De meeste automatisch werkende systemen kunnen, qua werking, altijd in een aantal vooraf gedefinieerde states ingedeeld worden.
Deze rechthoek heeft een bovenste- en onderste gedeelte, gescheiden door een horizontale lijn. Het bovenste gedeelte is voorzien voor de statenaam, het onderste (en ook grootste gedeelte) is voorzien voor de acties die dienen uitgevoerd te worden indien de state actief is/wordt. Hoe meer acties, hoe groter dit vak zal worden uiteraard.

Deze **statenaam** mag je vrij kiezen, maar probeer deze zinvol te houden. In bovenstaand voorbeeld geeft de statenaam bv. weer wat er gebeurt in deze state. Dat is uiteraard niet altijd mogelijk, zeker als er meerdere acties tegelijkertijd gebeuren in één bepaalde state. In dat geval kan/mag je bv. ook State1, State2, enz... gebruiken. **Belangrijk** is dat deze namen later identiek zijn in het PLC-programma, waar de statenamen PLC-variabelen zullen zijn.

De **acties** (handelingen/commando's) stellen voor wat er in het proces/machine dient te gebeuren als de state actief wordt. Dit kan ook een actie intern de PLC zijn, zoals bv. het starten van een timer, het op- en aftellen van een teller, het schrijven naar een database, enz...

Wat betreft het aansturen van uitgangen (Lre1, Lor1,...) moet je in principe niet vermelden dat deze laag (uit) wordt. Enkel wanneer iets actief moet worden. Dit wil ook zeggen dat ENKEL die acties uitgevoerd worden die vermeld staan. Bv. stel in State1 staat de actie Conveyor1=On. Als in State2 deze actie niet meer staat, zal Conveyor1 enkel in State1 draaien (en dus vanaf dat State2 actief wordt, niet meer draaien). Indien gewenst mag het laag worden van een uitgang (bv. een lamp die uit moet zijn in een bepaalde state) vermeld worden, maar dat hoeft dus zeker niet; daarom staat het hier telkens tussen haakjes. Opmerking: in de state Lor1_PG_1Hz geeft de actie Lor1=PG_1Hz aan dat Lor1 knippert met een frequentie van 1 Hz. PG_1Hz is immers één van de knippermerkers ingesteld via de HW-configuratie van de PLC.

Men gebruikt nogal eens het begrip Idle om de beginstate (eerste state na opstart) weer te geven. Maar dat mag ook een andere naam zijn.

- De **pijlen** tussen de verschillende states geven aan HOE het state machine verder evolueert en MET WELKE **VOORWAARDE** dit dient te gebeuren. Zo'n voorwaarde kan dus bestaan uit bv. een logische vergelijking (AND, OR). Er MOET dus bij elke pijl dergelijke voorwaarde staan.

Belangrijke opmerking:

Bovenstaande syntax(regels) dienen gevuld te worden bij het tekenen van een state machine. Deze dienen toegepast te worden tijdens het labo en het examen. Bij de evaluatie van het examen zal hiermee rekening gehouden worden.

Het voordeel van een state machine wordt nu zeer duidelijk. Een beschrijvende tekst, functionele beschrijving, wordt omgezet naar een duidelijk schematisch overzicht (softwaremodel). Iedereen die betrokken is bij het proces kan dan hierop aanmerkingen/verbeteringen aanbrengen op een snelle en duidelijke manier.

Bij complexe systemen zal meer dan één state machine dienen getekend te worden om een werkbaar en duidelijk overzicht te kunnen bewaren.

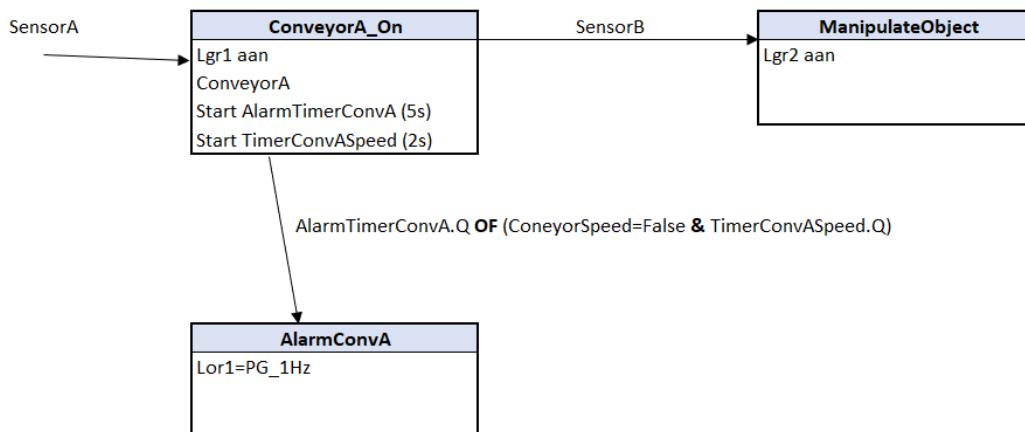
3.2 Gebruik van tijdfuncties (timers) in een state machine

Meestal zal een tijdfunctie gebruikt worden om, als een bepaalde state A actief is, na een bepaalde tijd, over te gaan naar state B. In state A dient dan zeker het starten van deze timer één van de acties (of misschien de enige actie) te zijn.

Het afgelopen zijn van de timer kan dan weergegeven worden als TimerNaam.Q (de letter Q verwijst naar de timeruitgang).

In de PLC wordt dan best een timer gebruikt die begint af te lopen als de state actief wordt (en blijft) en waarbij de timeruitgang hoog wordt na de gekozen tijdwaarde. Dit is bijgevolg het gedrag van een TON-timer. Het starten van de timer doe je best juist na de sectie van de state dewelke het startbevel van de timer is. Als bv. Timer1 gestart dient te worden in State1, dan programmeer je Timer1 (het starten) vlak na de programmasectie van State1.

Voorbeeld: transportband (alarm)



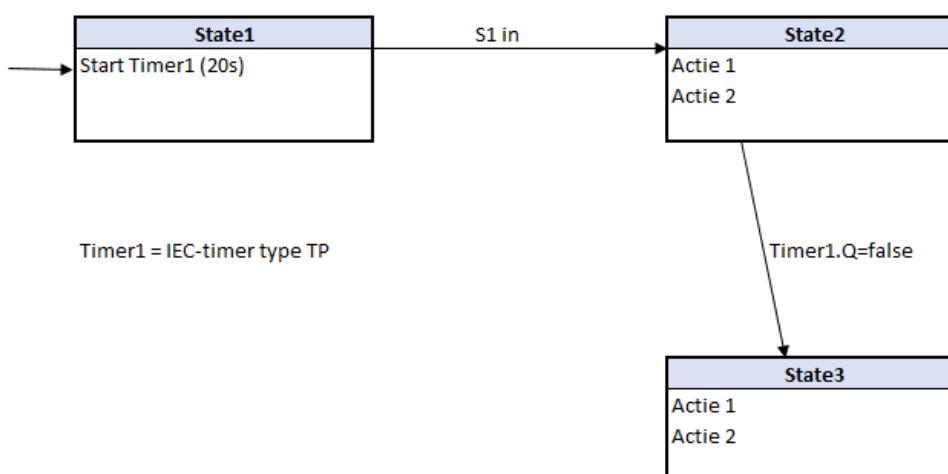
Stel een transportband A moet beginnen draaien indien een sensor A actief wordt (detectie van een object). Men weet nu dat in normale omstandigheden binnen de 3 s het object dient gedetecteerd te worden door sensor B. Indien dit niet gebeurt, dient een alarm actief te worden. In de state machine wordt dus een timer van 5s gestart in state ConveyorA_On (er wordt een tijd gekozen iets groter dan de verwachte tijd). Indien binnen de 5s sensor B bereikt wordt, zal de state ManipulateObject actief worden. Maar als na 5 seconden de sensor B niet bereikt wordt (en dus de timeruitgang AlarmTimerConvA .Q True wordt) zal de alarmstate actief worden.

Bijkomend is er een toerentaldetectie geplaatst op de motor van de transportband (controle dat motor draait; geen toerental: toerentalsensor=0). Er wordt vooropgesteld dat de motor moet draaien, ten laatste 2s nadat het startbevel gegeven is. Dit wordt gecontroleerd d.m.v. TimerConvASpeed.

Door later in de PLC een TON-timer hiervoor te gebruiken, zullenl als één van de 2 states bereikt worden, beide timeruitgangen ook automatisch FALSE worden.

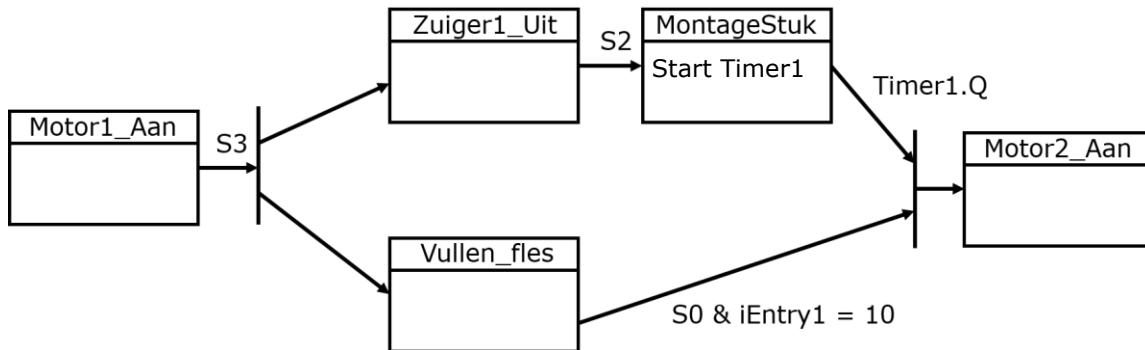
Gebruik van TP-timers:

Indien in een state machine een timer dient verder af te lopen, nadat de state waarin de timer gestart is inactief wordt, moet in het PLC-programma een timer van het type TP (IEC-timer) gekozen te worden. Deze blijft immers verder aflopen als de startvoorwaarde van deze timer wegvalt. Om te testen dat de timer afgelopen is, moet dan wel de timeruitgang op False afgevraagd worden.



3.3 Vanuit een state meerder andere states gelijktijdig activeren

Dit wordt volgens onderstaand voorbeeld getekend

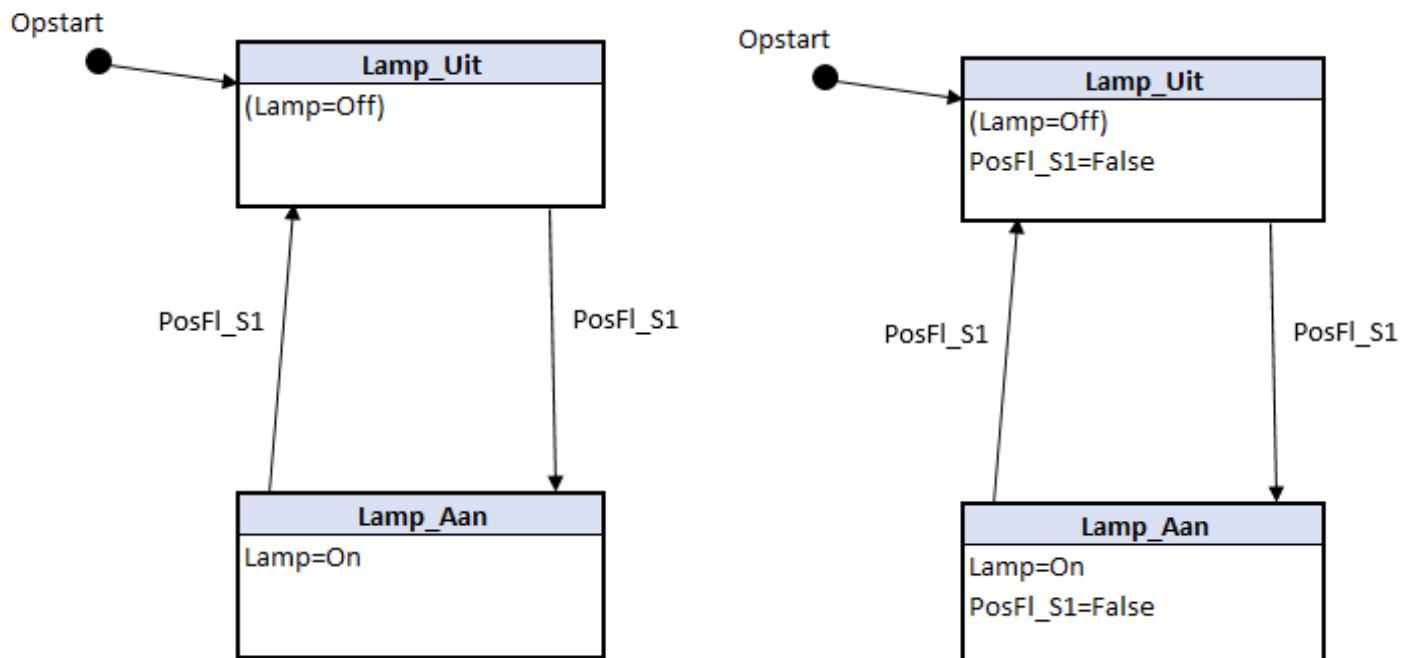


Vanuit de state 'Motor1_Aan' zullen er, als S3 True wordt, 2 states direct actief worden: 'Zuiger1_Uit' en 'Vullen_fles'. Maar de state 'Motor2_aan' kan pas actief worden als :

$S0 = 1 \text{ EN } iEntry1=10 \text{ EN state 'MontageStuk' actief is EN Timer1.Q} = 1$

3.4 Gebruik van flankdetectie in een state machine

Om het gebruik van flankdetectie toe te lichten, wordt als voorbeeld het state machine van een teleruptorschakeling genomen.



Eerst vooral dient opgemerkt dat PosFl_S1 een PLC-variabele voorstelt, een bool-variabele dewelke het resultaat is van een positieve flankdetectie op de variabele S1. Zoals geweten is PosFl_S1, bij het optreden van een positieve flank van S1, één volledige PLC-cyclus True (zie ook module 8 STEP7). Het is nu ook zo dat elke state machine later een PLC-programma voorstelt, dus een state machine wordt in totaliteit telkens binnen één totale PLC-cyclus uitgevoerd.

Veronderstel het linkse state machine. Na opstart is de state Lamp_Uit actief en is de lamp uit.

Indien er een positieve flank op S1 gedetecteerd wordt (bv. indrukken van een NO-drukknop), zal overgegaan worden naar de state Lamp_Aan, de lamp wordt aangeschakeld. Maar, zoals eerder gesteld wordt heel de state machine uitgevoerd binnen één cyclus. Dus binnen dezelfde cyclus is, na het actief worden van de state Lamp_Aan, de flankdetectievariabele PosFl_S1 nog steeds True en zal onmiddellijk overgegaan worden naar de state Lamp_Uit. Gevolg: lamp terug uit. Dus de teleruptorschakeling werkt niet zoals het moet.

Dit gebeurt wel in de rechtse state machine. Daar zal telkens als een state actief wordt, d.m.v. de variabele PosFl_S1, deze variabele direct in die state op False gezet worden. Er moet dus opnieuw een nieuwe positieve flank gegenereerd worden, door terug op S1 te drukken, om naar de volgende state te kunnen gaan.

De variabele PosFl_S1 kan tijdens eenzelfde cyclus op deze manier maar éénmaal gebruikt worden, wat de bedoeling is om de teleruptorschakeling correct uit te voeren. Hoe flankdetecties programmeren in SCL : zie paragraaf 4.3.3.

4 Van state machine naar een SCL-programma in TIA Portal: de richtlijnen

Het opstellen van het state machine vergt het meeste denkwerk. Het state machine is immers een grafische oplossingsmethode, van waaruit het PLC-programma kan ontworpen worden. Als de state machine correct ontworpen is, dienen enkel onderstaande richtlijnen gevuld te worden om tot een werkend PLC-programma te komen.

Onderstaande richtlijnen dienen dan ook STRIKT gevuld te worden in het labo en op het examen.

In wat volgt zal het state machine vertaald worden naar een SCL-programma in TIA Portal.

Twee zaken zijn hierbij belangrijk:

- Correcte keuze en benaming van de variabelen
- Opbouw/volgorde en structuur van het SCL-programma

4.1 Keuze en benaming van de gebruikte PLC-variabelen

Zoals eerder aangehaald dient een state machine beschreven te worden met PLC-variabelen, waarvan de namen gedeclareerd dienen te worden in het TIA-programma. Alle namen (van states, voorwaarde, acties) gebruikt in het state machine **moeten** dezelfde zijn als gebruikt in het TIA-programma.

Alle interne variabelen dienen in datablokken gedeclareerd te worden, met uitzondering van de pulsgeneratoren die via de HW-configuratie toegewezen zijn aan MB0.

Gebruikte timers worden automatisch aan een DB toegewezen.

Volgende datablokken (symbolische namen) dienen aangemaakt te worden (adressen van de DB's worden door TIA-portal automatisch toegekend en zijn niet van belang):

'State'	:	voor de verschillende states
'PosFI'	:	voor de positieve flankdetecties
'NegFI'	:	voor de negatieve flankdetecties
'NR'	:	voor de niet-remanente variabelen Opstart en Zero@Start en eventueel andere nodige variabelen

Alle DB's dienen bij het aanmaken op niet-remanent gezet worden (Retain-kolom uitvinken), zodat bij (her)opstart van de PLC alle variabelen geïnitialiseerd worden (dus op 0 komen).

Indien bij een bepaalde machine/proces de actieve state behouden moet blijven bij wegvallen van de spanning, zodat na wederkeren van de spanning het proces verder aanvangt met deze state, dienen voor de states remanente variabelen gekozen te worden. Hiervoor kan dan een remanente DB (met bv. de naam 'REM') aangemaakt worden.

4.2 Opbouw/volgorde en structuur van het SCL-programma

Het programma dient in SCL ontworpen te worden en voorzien te zijn van de nodige commentaarregels (via //).

Het SCL-programma dient via de commentaarregels ook in verschillende secties worden ingedeeld, om de leesbaarheid te vergroten. Elke sectie dient een relevante titel te krijgen.

De volgorde (van de verschillende programmasecties) is als volgt:

- Eerst de opstart (activeren van de initiële state)
- Dan het programmeren van eventueel gebruikte flankdetecties
- Dan achter elkaar de vergelijkingen van de verschillende states in chronologische volgorde
- Dan als laatste het aansturen/activeren van de gebruikte uitgangen; hierbij dient per aparte uitgang een aparte sectie voorzien te worden.

De gebruikte timers (normaal TON, TP indien nodig) gebruik je best juist na de programmasectie van de state dewelke de startvoorwaarde van de timer is !

4.3 Voorbeelden van een SCL-programma volgens state machine

4.3.1 Teleruptorschakeling (gebruik flankdetectie, state machine van p. 7)

In het project zijn, zoals volgens de richtlijnen, 3 DB's aangemaakt.

In alle DB's is voor alle variabelen de selectie Retain uitgevinkt (dus allen zijn zo niet-remanent).

De namen van de verschillende DB's zijn: NR, PosFl en State.

Het programma is geschreven in een FC en wordt hieronder weergegeven.

De gebruikte lamp is Lgr1.

```
0001 // Opstart
0002 "NR".Opstart := NOT "NR"."Zero@Start";
0003 "NR"."Zero@Start" := true;
0004
0005 // Positieve flankdetectie op S1
0006 "PosFl".S1 := "S1" AND NOT "NR".S1_VorigeCyclus;
0007 "NR".S1_VorigeCyclus := "S1";
0008
0009 // State Lamp_Uit
0010 IF "NR".Opstart OR ("State".Lamp_Aan AND "PosFl".S1)
0011 THEN
0012     "PosFl".S1 := false;
0013     "State".Lamp_Uit := true;
0014     "State".Lamp_Aan := FALSE;
0015 END_IF;
0016
0017 // State Lamp_Aan
0018 IF "State".Lamp_Uit AND "PosFl".S1
0019 THEN
0020     "PosFl".S1 := false;
0021     "State".Lamp_Aan:= true;
0022     "State".Lamp_Uit := false;
0023 END_IF;
0024
0025 //Schakelen lamp
0026 "Lgr1" := "State".Lamp_Aan;
```

4.3.2 Gebruik van TON-timers (state machine van p. 5)

```
0001 // State ConveyorA_On
0002 IF "SensorA" THEN
0003   "State".ConveyorA_On := true;
0004 END_IF;
0005
0006 // Starten AlarmTimerConvA (5s)
0007 "AlarmTimerConvA".TON(IN:="State".ConveyorA_On,
0008                         PT:=t#5s);
0009
0010 // State ManipulateObject
0011 IF "State".ConveyorA_On & "SensorB"
0012 THEN
0013   "State".ManipulateObject := true;
0014   "State".ConveyorA_On := false;
0015 END_IF;
0016
0017 // State AlarmConvA
0018 IF "State".ConveyorA_On & ("AlarmTimerConvA".Q OR NOT "ConveyorSpeed")
0019 THEN
0020   "State".AlarmConvA := true;
0021   "State".ConveyorA_On := false;
0022 END_IF;
0023
0024 // Acties in state ConveyorA_On
0025 IF "State".ConveyorA_On THEN
0026   "Lgr1" := true;
0027   "ConveyorA" := true;
0028 END_IF;
0029
0030 // Actie Lgr2
0031 "Lgr2" := "State".ManipulateObject;
0032
0033 // Actie Lor1
0034 "Lor1" := "State".AlarmConvA & "PG_1Hz";
```

4.3.3 Programmeren van flankdetecties

Bij gebruik van een S7-300-PLC is er in TIA Portal geen flankdetectie beschikbaar in de programmeertaal SCL.

Wel in LAD en FBD : P_TRIG , N_TRIG; alsook in STL : FP, FN.

In SCL dient een flankdetectie dus zelf ontworpen te worden.

Hiervoor dienen per flankdetectie 2 variabelen aangemaakt worden.

De ene variabele bevat de flankdetectie (positief of negatief), dewelke dus 1 cyclus True is. Deze variabele dient gedeclareerd te worden in de datablok "PosFI" of "NegFI" en krijgt de naam van de variabele waarop de flankdetectie dient te gebeuren.

Stel bv. een positieve flankdetectie op S0.

In de datablok "PosFI" maak je een variabele S0 en deze heeft dan de volledige symbolische naam "PosFI".S0 .

De andere variabele bevat altijd de waarde van de variabele waarop de flankdetectie moet uitgevoerd worden, maar dan van de vorige programmacyclus. Deze variabele dient gedeclareerd te worden in de datablok "NR".

Dus voor een flankdetectie op S0 wordt dit de variabele "NR".S0_VorigeCycl .

Stel een negatieve flankdetectie op S1.

In de datablok "NegFI" maak je een variabele S1 aan en in de datablok "NR" een variabele S1_VorigeCycl.

Een positieve- en negatieve flankdetectie kan dan geprogrammeerd worden als volgt:

// Positieve flankdetectie op S0

"PosFI".S0 := "S0" AND NOT "NR".S0_VorigeCycl;

"NR".S0_VorigeCycl := "S0";

// Negatieve flankdetectie op S1

"NegFI".S1 := NOT "S1" AND "NR".S1_VorigeCycl;

"NR".S1_VorigeCycl := "S1";

Belangrijk !!

In het programma maak je voor flankdetecties aparte segmenten en plaats je deze bovenaan. Achter het segment voor opstart.

5 Opgaven

Belangrijke opmerking vooraf:

Alle onderstaande opdrachten dienen volledig uitgevoerd te worden conform de eerder geziene richtlijnen in paragraaf 4.

Opgaven

- Module 5 , blz. 3 : laatste twee opdrachten
- Module 7 , blz. 4
 - Opgave uitvoeren zonder gebruik te maken van de aanwezige tellerfuncties in de instructielijst, dus zelf een telfunctie creëren in je state machine.
- Module 8 , blz. 8 : twee laatste opgaven
- Module 10 , opdrachten blz. 15 , 16 & 17
- Module 11 , blz. 7

Extra opgave 1: verkeersverlichten

Functionele beschrijving

Een verkeersverlichting kan op 2 manieren functioneren via schakelaar S10, nl. S10 open is normale werking en S10 dicht is knipperlichtwerking.

Normale werking:

In deze werking wordt automatisch de volgende cyclus chronologisch doorlopen: Lre1 (2s) , Lgr1 (3s) , Lor1 (1s).

Knipperlichtwerking:

Hier zal het oranje licht (Lor1) constant knipperen met een frequentie van 2 Hz, de aan-tijd is 150 ms en de uit-tijd is 350 ms. Dus je kan hier de pulsgenerator van 2 Hz in MB0 niet gebruiken, want daar zijn aan- en uit-tijd gelijk aan elkaar.

Je mag/kan voor deze pulsgenerator een apart State machine tekenen !

Bij opstart van het systeem bepaalt de stand van S10 dus hoe de verkeersverlichting begint.

Extra opgave 2: verkeersverlichten, uitbreiding

Indien bij normale werking wordt overgeschakeld naar knipperlichtenwerking, blijft de normale cyclus verder werken totdat het oranje licht zou aangestuurd worden. Dan begint het oranje licht te knipperen.

Indien bij knipperlichtenwerking wordt overgeschakeld naar normale werking, dient er steeds begonnen te worden met rood.

Extra opgave 3: garagepoortsturing

Functionele beschrijving:

Met een NO-drukknop S0 kan een garagepoort geopend en gesloten worden.

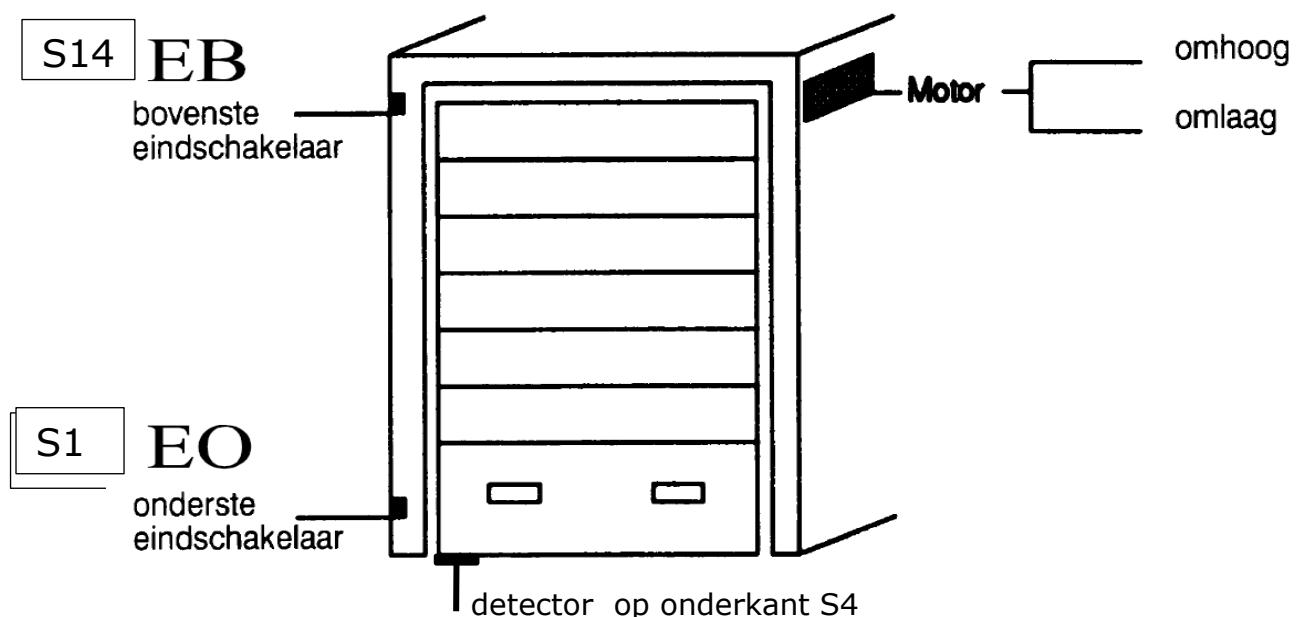
Als de poort dicht is en de drukknop S0 wordt bediend gaat de poort omhoog via contactor K1 (Lre1). Als er terug op de drukknop gedrukt wordt stopt de poort met bewegen en blijft op deze positie staan. Terug op de drukknop drukken laat de poort in tegengestelde richting bewegen, dus nu naar omlaag via contactor K2 (Lre2).

Terug drukken en de poort stopt terug, enz...

Indien de poort volledig open (gedetecteerd door de NC-eindeloopschakelaar S14) of volledig gesloten (gedetecteerd door de NC-eindeloopschakelaar S17) is, stopt de motor.

Indien de poort omlaag gaat en de detector S4 (NC-contact) wordt bediend (poort drukt op voorwerp of persoon) dient de poort automatisch terug naar boven te bewegen.

Er is ook een thermische beveiliging (F1) voorzien (NC-contact) om de motorcontactoren uit te schakelen indien de motor overbelast wordt (F1 = S15). Als de thermische beveiliging terug in haar begintoestand komt, moet eerst op de drukknop gedrukt worden vooraleer de poort terug kan bewegen, in de richting dewelke bezig was.



6 Tekenken van een state machine op PC

Alle state machines in dit document zijn getekend in Excel.

Via de gratis online-tool Draw.io kan je, naast allerhande andere diagrammen, ook een state machine ontwerpen.

Eens aangemeld kan je bv. kiezen waarop je aangemaakte diagrammen dienen bewaard te worden.

Indien je kiest voor Open Existing Program, kan je via het bestand "StateMachine.drawio" (via Canvas) al gebruik maken van een startproject, waarmee je direct aan de slag kan voor het tekenen van state machines.

Zet dit bestand vooraf eerst ergens op je PC.

Via het Scratchpad links (als het niet zichtbaar is → via View aanvinken) kan je de meest gangbare symbolen om een state machine te ontwerpen hier invoegen, via het bestand "Pallet_StateMachine.xml" (Canvas).

Dit doe je door op de schrijftool (Edit) te klikken en daarna Import te kiezen en daarna af te sluiten met Save.

Onderstaande symbolen heb je dan ter beschikking:

