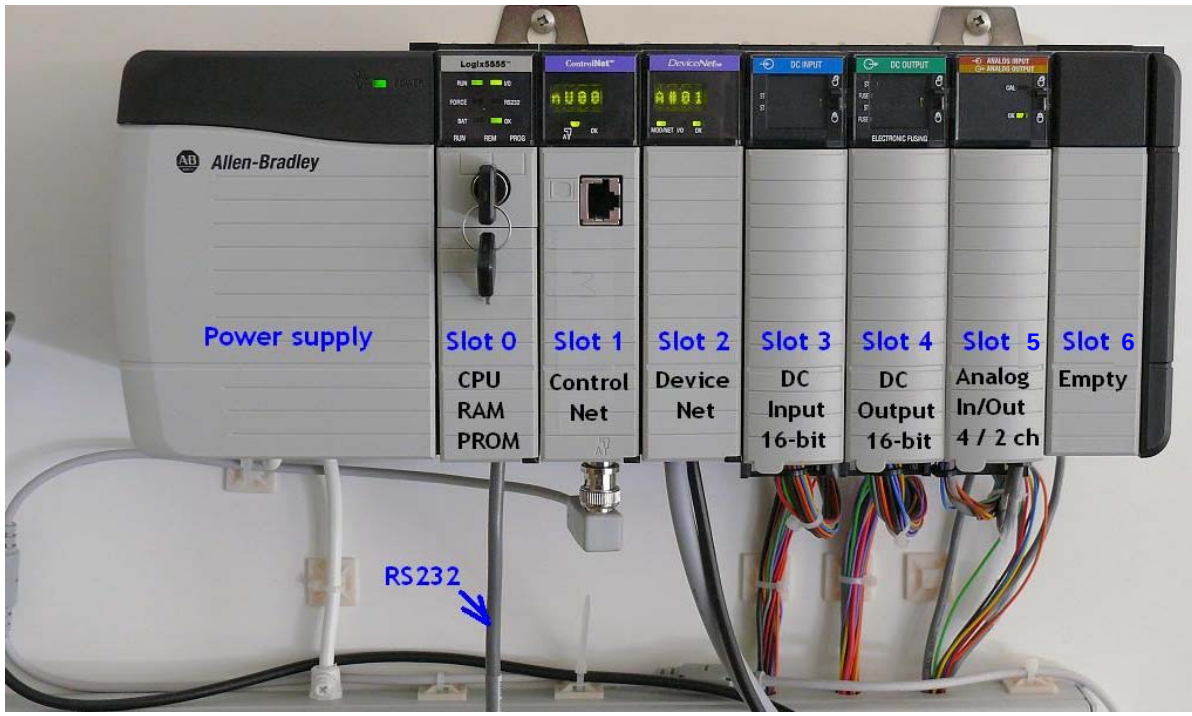
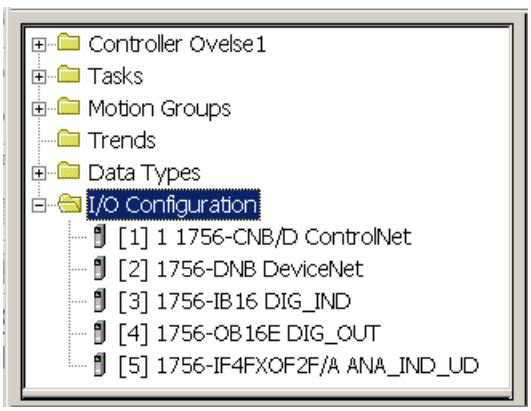
Typisk modul-opbygget PLC system (Allan Bradley)



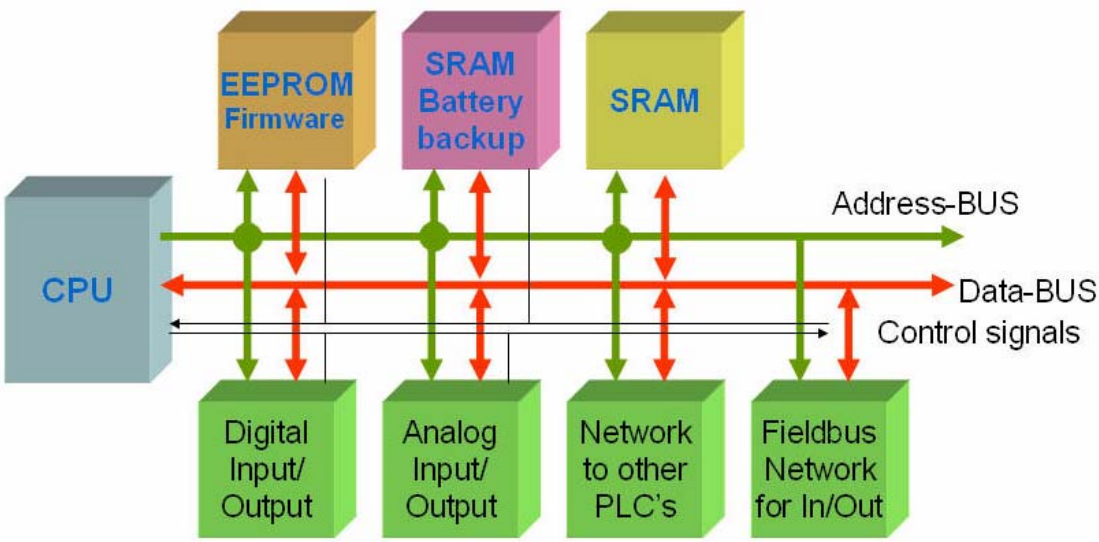


Programmøren er nødt til at definere hvilke moduler systemet består af og i hvilke slots de er placeret.

Et typisk system vil have såvel analoge som digitale ind- og udgange (Slot 3, 4 og 5)

Man vil ofte bruge feltbusser (Fieldbus) i forbindelse med input/output signaler. (Slot 2)

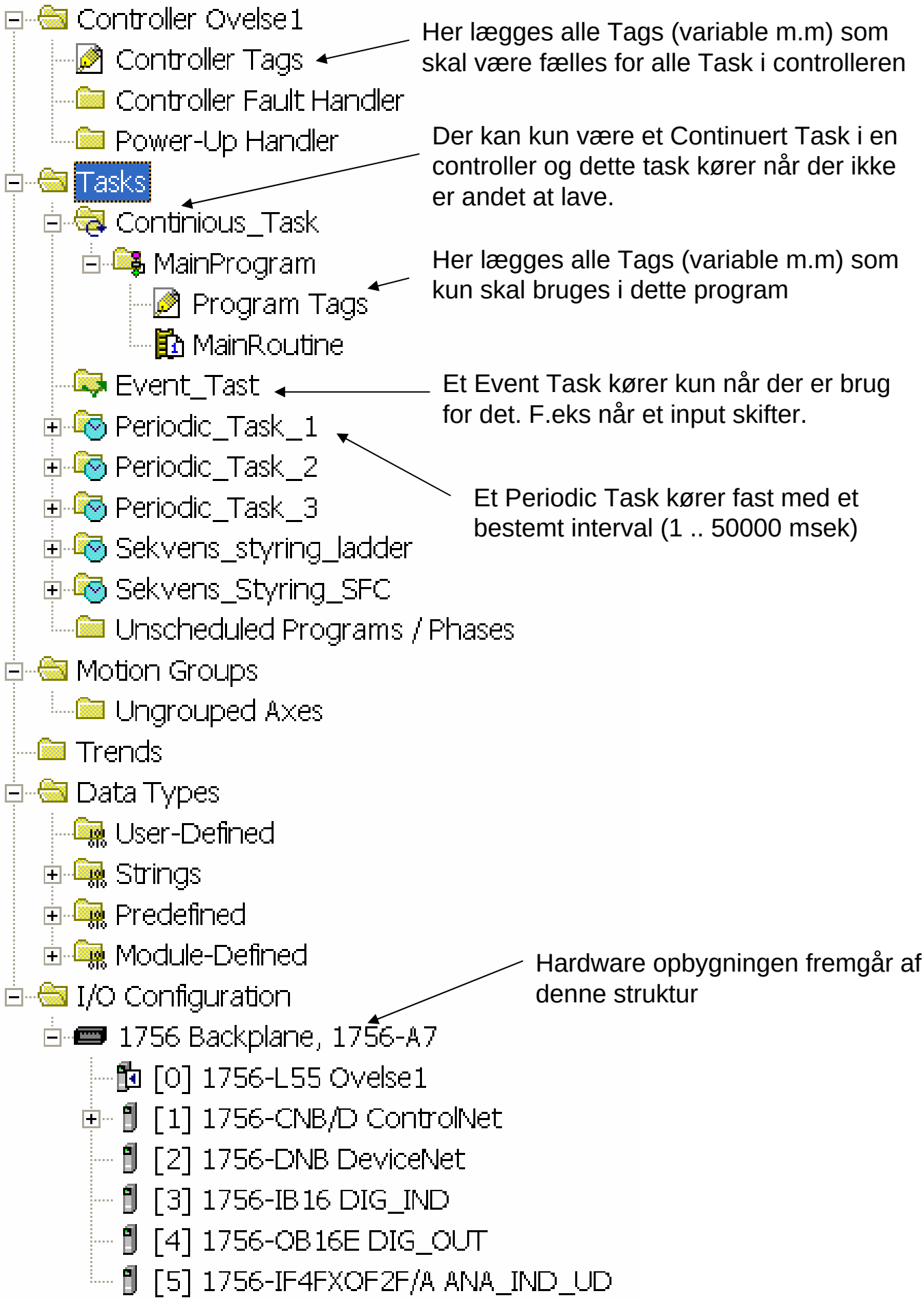
Endelig kan PLCer udveksle data via netværk Som f.eks. Controlnet (Slot 1)

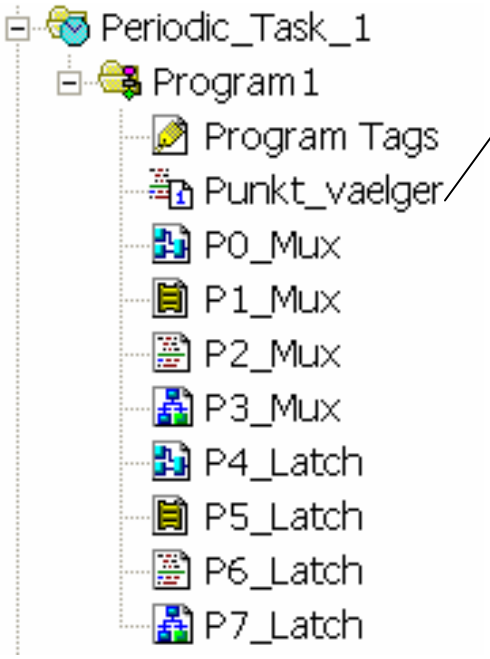


EEPROM = Electrical Erasable Programmable Read Only Memory
SRAM = Static Random Access Memory
Firmware = Software which enables the CPU (PLC) to execute your ladder programs ..

Common structure of a PLC system (Simplified)

Typisk programstruktur for en PLC (Allan Bradley)

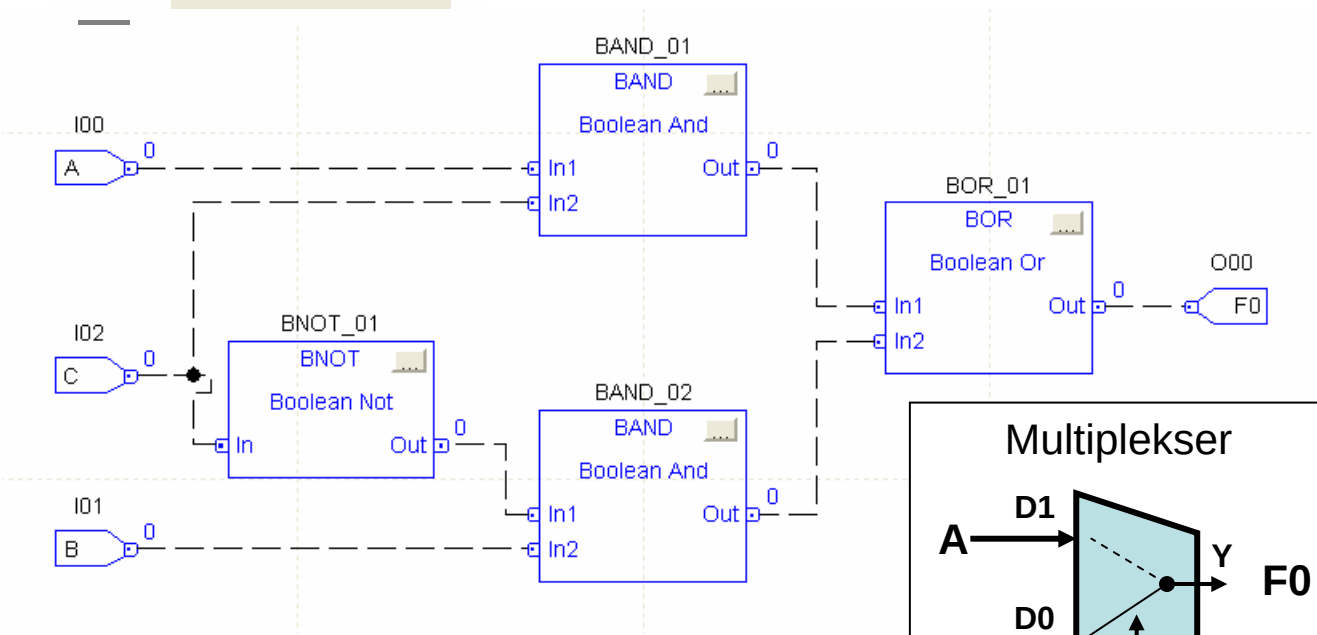




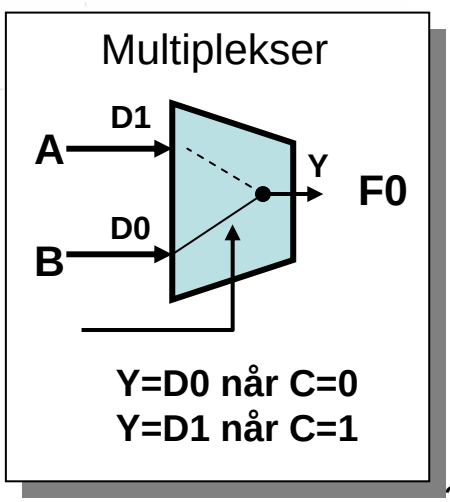

```
Select := (Digin_16bit / 256) and 7;
// Select = x

IF NOT Sel3 and not Sel4 THEN
// Sel3 og Sel4 skal være 0
    CASE Select OF
        0:  JSR( P0_Mux,0);
        1:  JSR( P1_Mux,0);
        2:  JSR( P2_Mux,0);
        3:  JSR( P3_Mux,0);
        4:  JSR( P4_Latch,0);
        5:  JSR( P5_Latch,0);
        6:  JSR( P6_Latch,0);
        7:  JSR( P7_Latch,0);
    END_CASE;
END_IF;
```

Et program indeholder rutiner, hvoraf én er hovedrutinen
I dette tilfælde er hovedrutinen skrevet i Strukturet Tekst

Funktions blokke

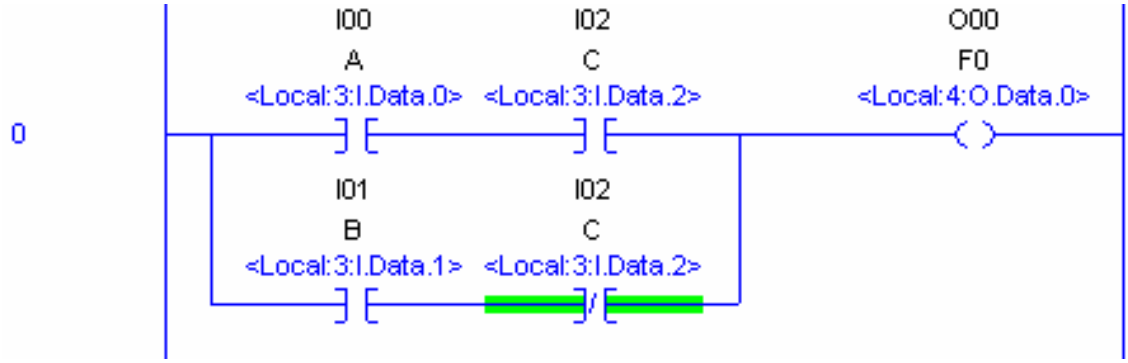


Funktionsblokke er komponenter som kan udføre alt lige fra simpel AND / OR logik og til hele PID regulatorer.



P1_Mux

Ladder diagram



Ladder diagrammer er afledt de gamle relæstyringer og er den klassiske programmeringsform.
Serieforbindelse svarer til AND og parallel svarer til OR

P2_Mux

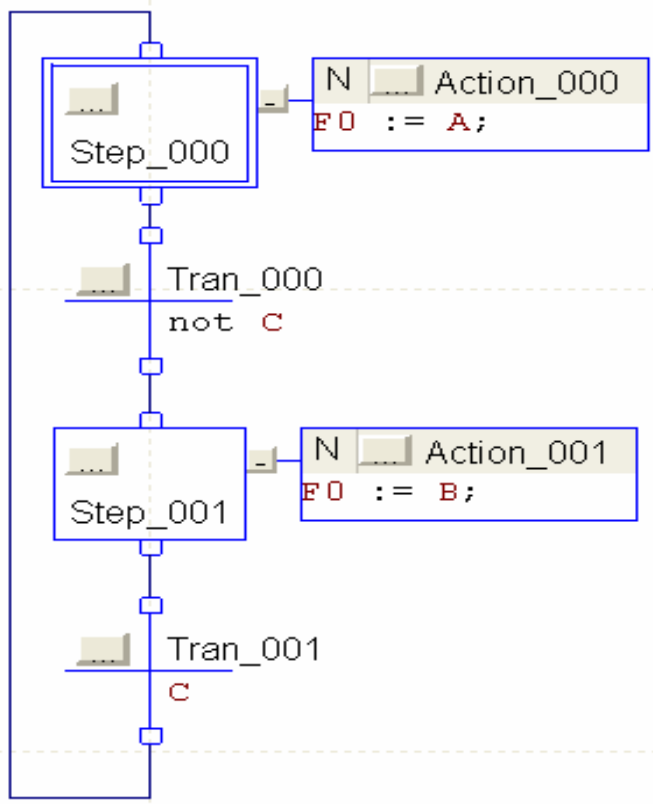
P3_Mux

Strukturet tekst

```
if C then
   F0 := A;
else
   F0 := B;
end_if;

// Alternativt statement -
// Bemærk:
case (Local:3:I.Data/4) and 1  of
   0:
      F0 := A;
   1:
      F0 := B;
end_case;
```

Stukturet tekst svarer til de programmeringssprog der anvendes inden for IT-branchen (C#, Java)

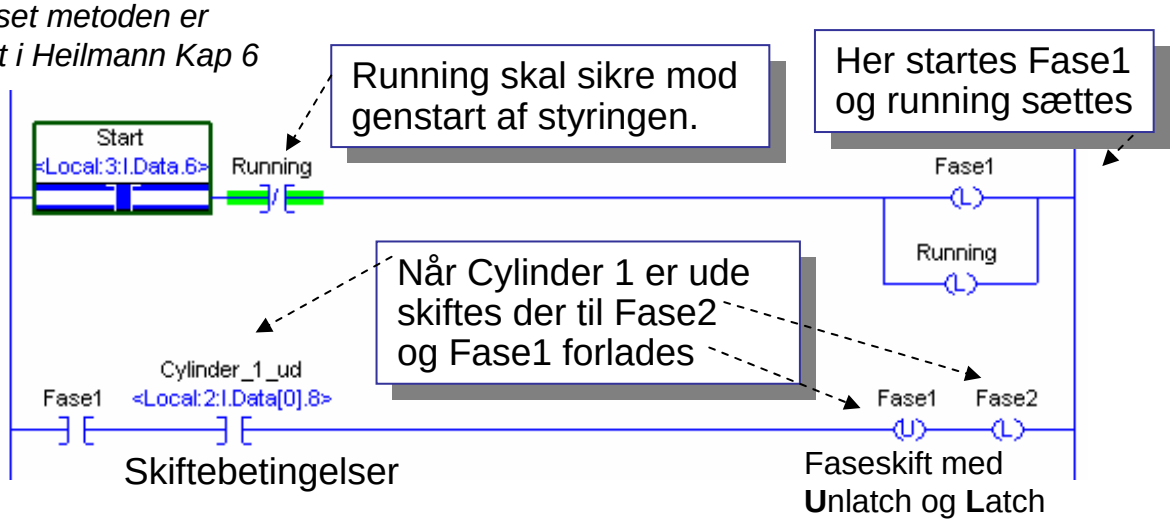
SFC - Sekventiel Function Chart



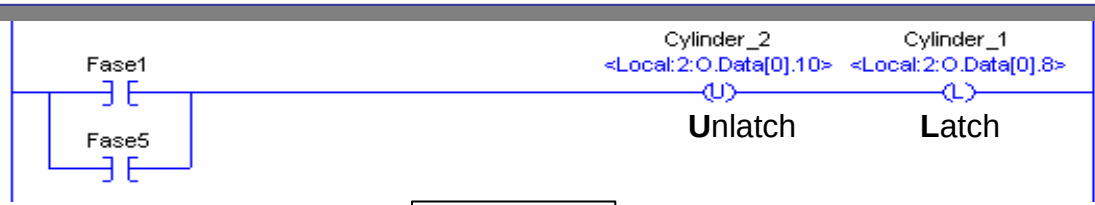
SFC er egentligt beregnet til sekvensstyringer, men kan for eksemplets skyld også lave en multiplekser.

Sekvensstyring med Ladder diagram

Set-Reset metoden er beskrevet i Heilmann Kap 6



Når Fase1 (og Fase5) er aktiveret ønskes Cylinder 2 sendt ind og Cylinder 1 skal sendes ud.



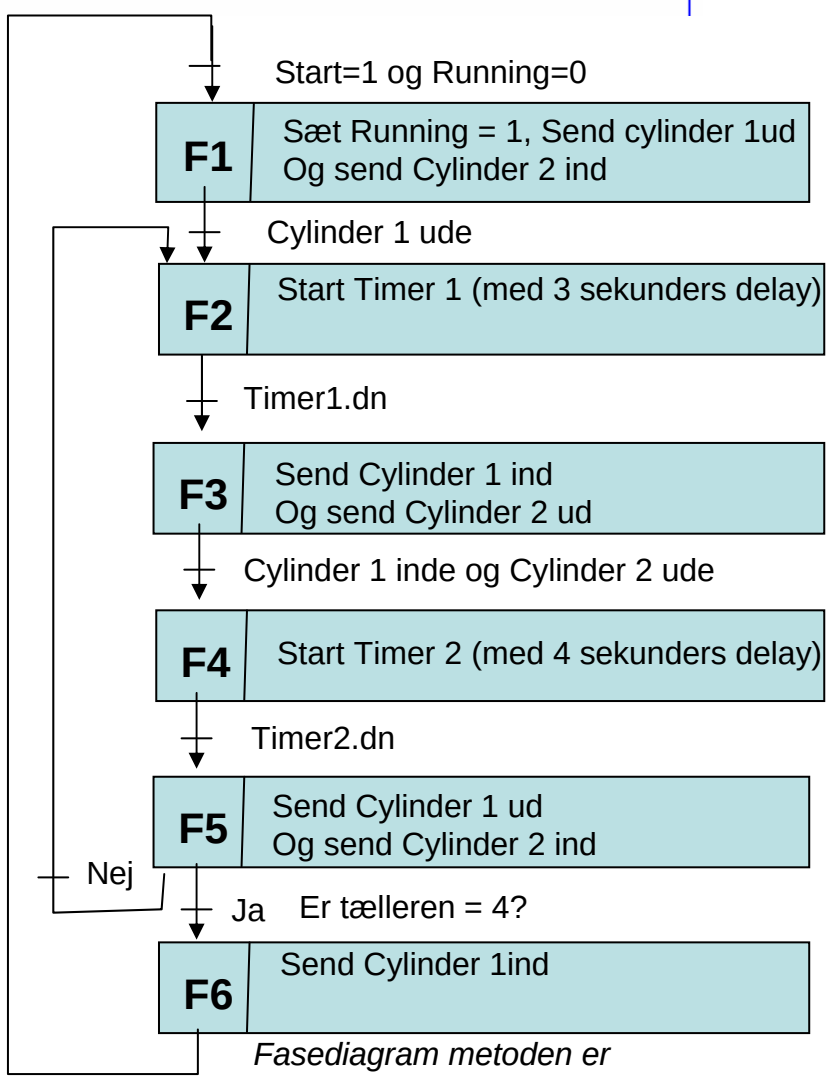
I forbindelse med en maskin-styring er der behov for følgende:

Der skal ventes på at start aktives og man må ikke kunne genstarte styringen mens den arbejder.

Cylinder 1 skal køre ud og blive ude i 3 sekunder.

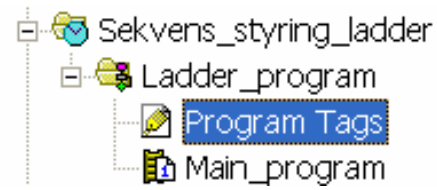
Derefter skal Cylinder 1 køre ind og Cylinder 2 skal køre ud og blive ude i 4 sekunder.

Denne sekvens skal gentages 4 gange hvorefter Cylinder 2 skal køre ind og styringen skal afvente start igen.



Fasediagram metoden er beskrevet i Heilmann Kap 11

Program Tags til Sekvensstyring



En typisk sekvensstyring vil have behov for et antal Faser – **Fase1**, **Fase2** osv.

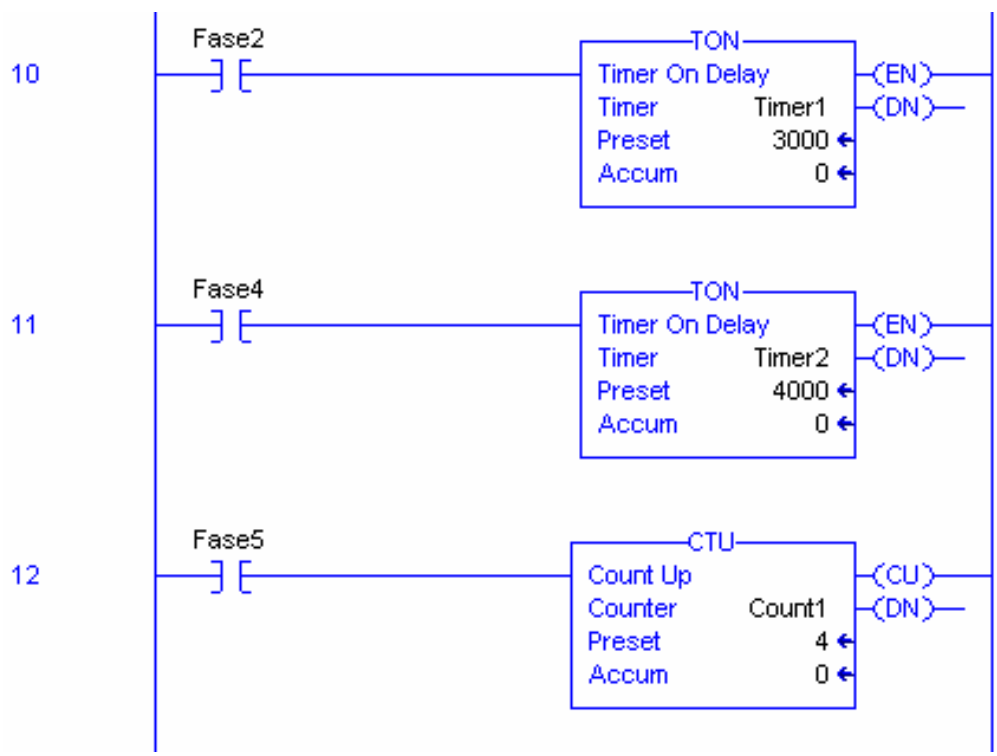
Desuden vil der være brug for timere til at generere tids-forsinkelser:
-**Timer1** og **Timer2**

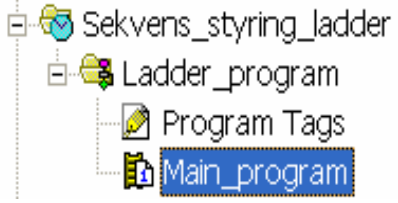
Endvidere er der brug for en tæller:
-**Count1**

Timere og Tæller aktiveres når de respektive faser er aktive.

Scope: Ladder_program | Show... | Show All

Name	Base Tag	Data Type	Style
Count1		COUNTER	
Count1.PRE		DINT	Decimal
Count1.ACC		DINT	Decimal
Count1.CU		BOOL	Decimal
Count1.CD		BOOL	Decimal
Count1.DN		BOOL	Decimal
Count1.OV		BOOL	Decimal
Count1.UN		BOOL	Decimal
Fase1		BOOL	Decimal
Fase2		BOOL	Decimal
Fase3		BOOL	Decimal
Fase4		BOOL	Decimal
Fase5		BOOL	Decimal
Fase6		BOOL	Decimal
Running		BOOL	Decimal
Start	Local:3:I.Data.6(C)	BOOL	Decimal
Timer1		TIMER	
Timer1.PRE		DINT	Decimal
Timer1.ACC		DINT	Decimal
Timer1.EN		BOOL	Decimal
Timer1.TT		BOOL	Decimal
Timer1.DN		BOOL	Decimal
Timer1.FS		BOOL	Decimal
Timer1.LS		BOOL	Decimal
Timer1.OV		BOOL	Decimal
Timer1.ER		BOOL	Decimal
Timer2		TIMER	





Løsningsforslag (til orientering)

Rung #1 .. #6
Her er alle faseskift samlet

#1 Sørger for at der kun kan startes én gang.

#2 Venter på at Cylinder 1 er kommet ud.

#3 Venter på at der er gået 3000 millisekunder.

#4 Venter på at Cyl 1 inde og Cylinder 2 ude.

#5 Venter 4000 msek.

#6 Afventer Cyl1 og 2 … Hvis styringen har kørt 4 gange stoppes og ellers gentages (Fase 2)

Rung #7 .. #12
Her er alle aktioner samlet

#7 Send begge Cylindre ind.

#8 Cyl 1 ud og Cyl2 ind

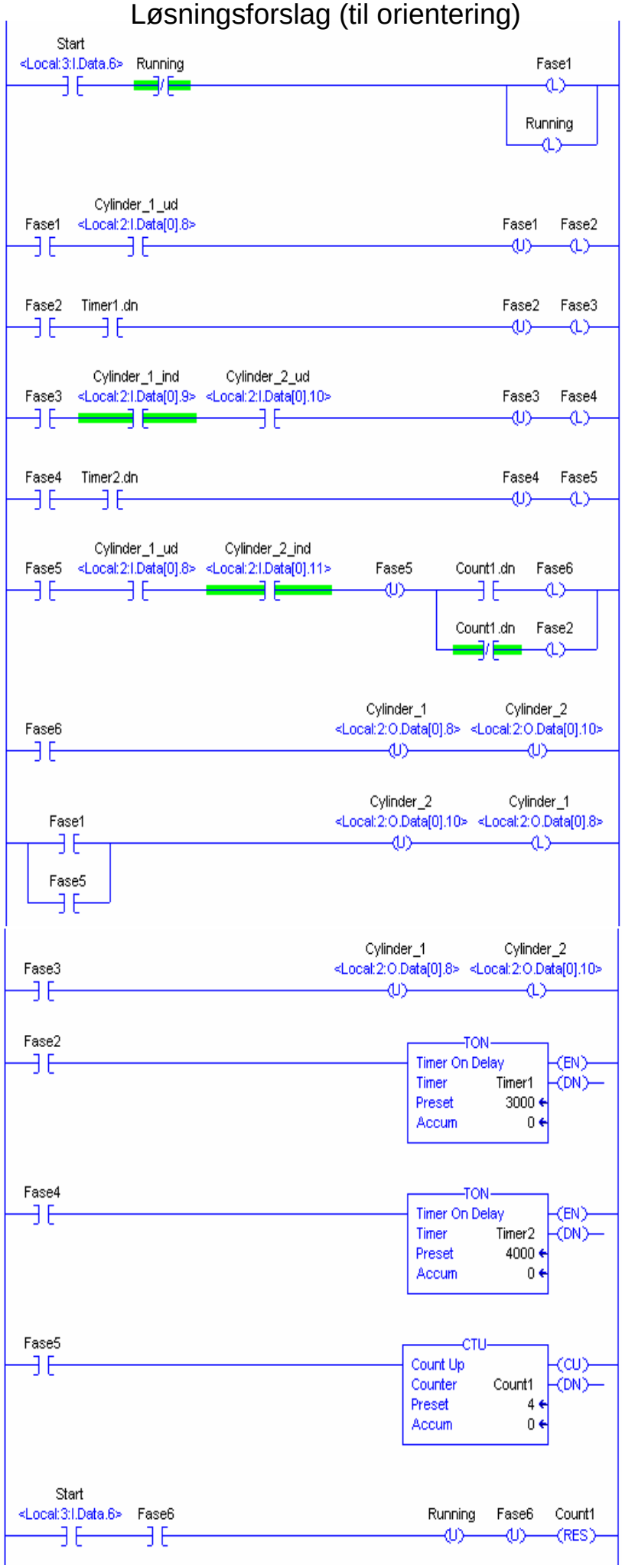
#9 Cyl 1 ind og Cyl 2 ud

#10 Timer1 laver tidsfor-sinkelser på 3000 msek.

#11 Timer2 laver tidsfor-sinkelser på 4000 msek.

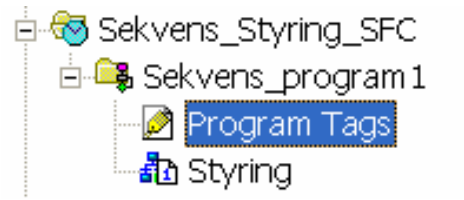
#12 Count1 tæller antallet af genløb i sekvensstyringen (Done sættes ved 4 gange)

#13 Sørger for genstart og initialisering af sekvenstyringen

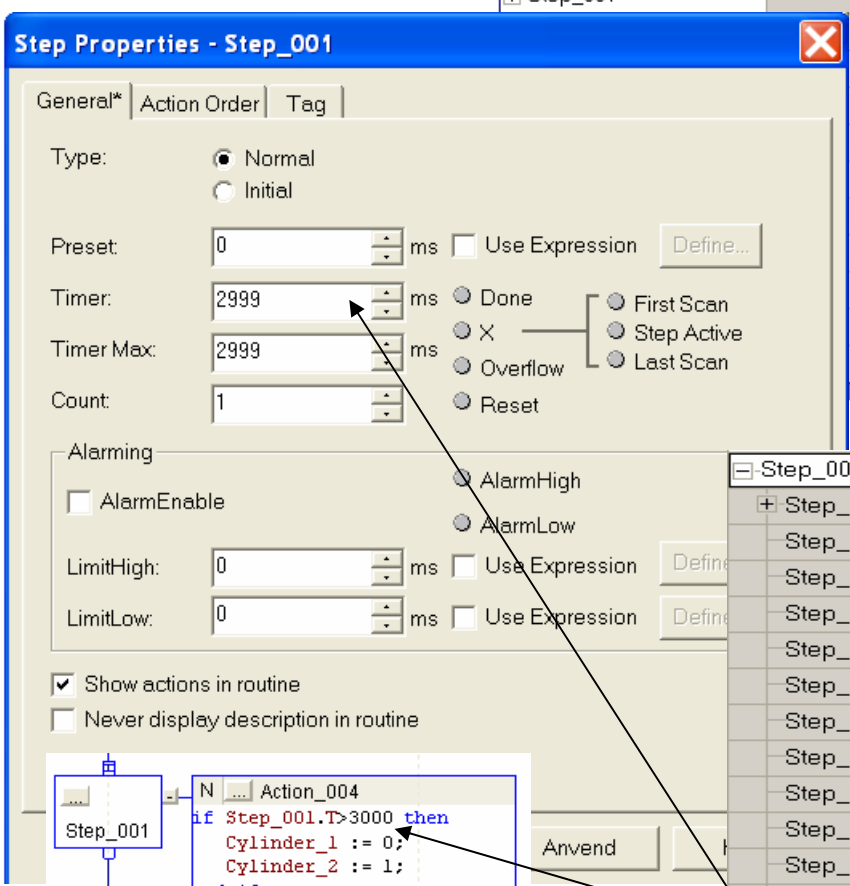


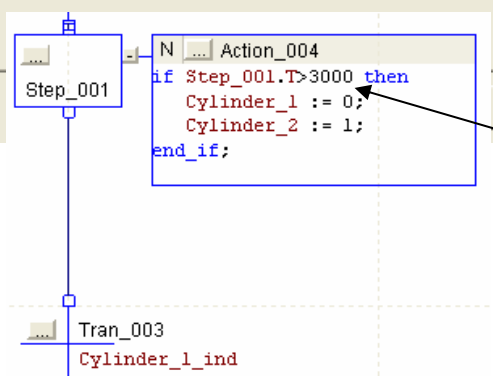
Sekvensstyring med SFC (Grafcet)

Læs mere Heilmann Kap 12



Action_000		SFC_ACTION	
Action_001		SFC_ACTION	
Action_002		SFC_ACTION	
Action_003		SFC_ACTION	
Action_004		SFC_ACTION	
Action_005		SFC_ACTION	
Afvent_Start		SFC_STEP	
Devicenet_run	Local:2:O.Comma...	BOOL	Decimal
Igen		BOOL	Decimal
Start	Local:3:I.Data.7(C)	BOOL	Decimal
Step_000		SFC_STEP	
Step_001		SFC_STEP	
		SFC_STEP	
		SFC_STEP	
		DINT	Decimal
		BOOL	Decimal
		BOOL	Decimal
		BOOL	Decimal
		BOOL	Decimal
		BOOL	Decimal
		BOOL	Decimal
		BOOL	Decimal





Step_001	SFC_STEP
Step_001.Status	DINT
Step_001.X	BOOL
Step_001.FS	BOOL
Step_001.SA	BOOL
Step_001.LS	BOOL
Step_001.DN	BOOL
Step_001.OV	BOOL
Step_001.AlarmEn	BOOL
Step_001.AlarmLow	BOOL
Step_001.AlarmHigh	BOOL
Step_001.Reset	BOOL
Step_001.PRE	DINT
Step_001.T	DINT
Step_001.TMax	DINT
Step_001.Count	DINT
Step_001.LimitLow	DINT
Step_001.LimitHigh	DINT

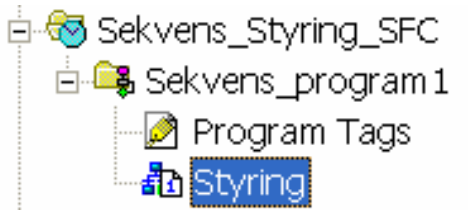
SFC styringer er baseret på 4 forskellige elementer:

Step – Angiver en fase hvor styringen kan opholde sig (evt flere aktive samtidigt)

Transitioner – Er betingelser der skal være opfyldt for at man kan forlade et step.

Aktioner – Er handlinger som ønskes udført i forbindelse med et step.

Forgreninger – Viser sammenhængen mellem de enkelte steps.

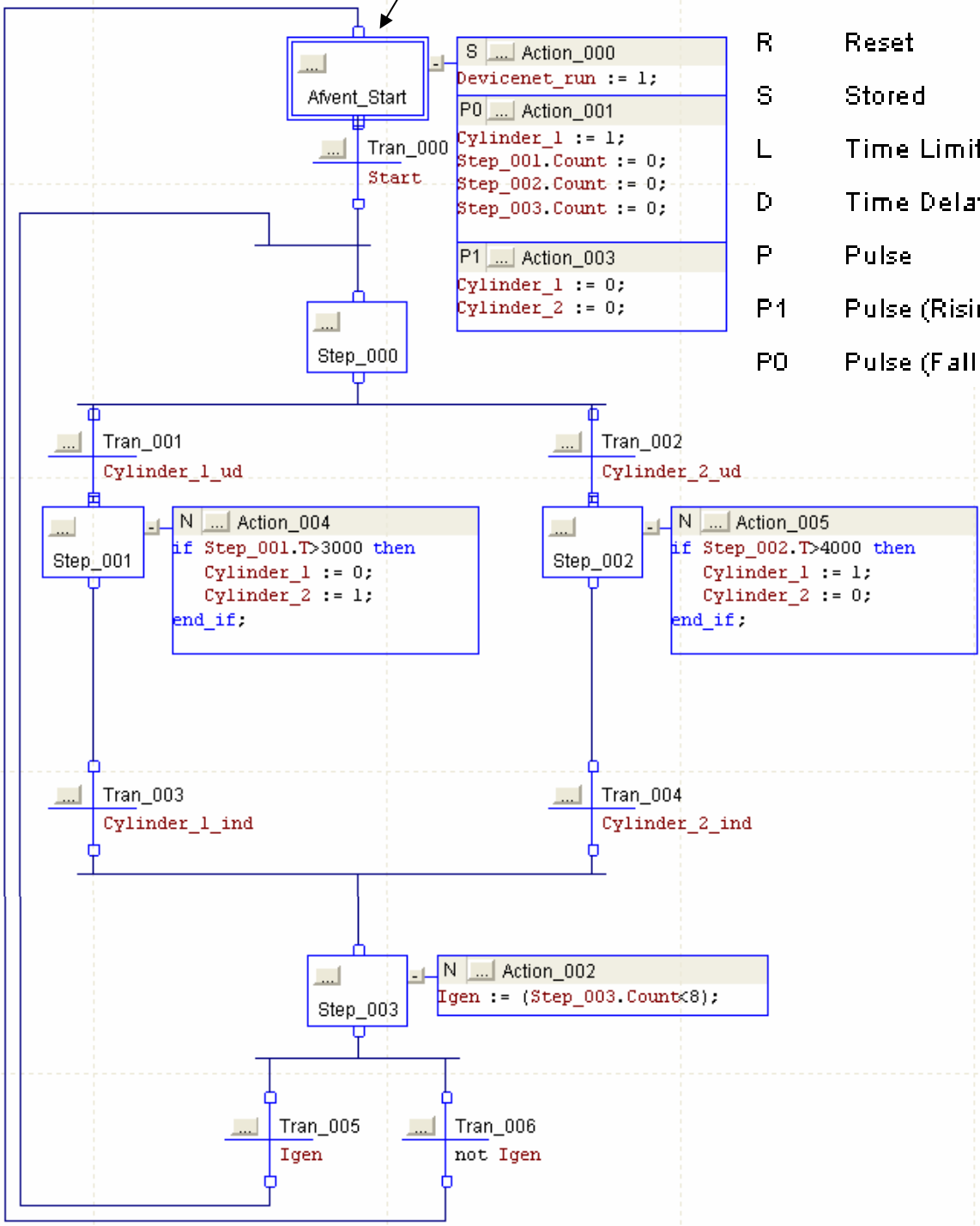


Sekvensstyring med SFC

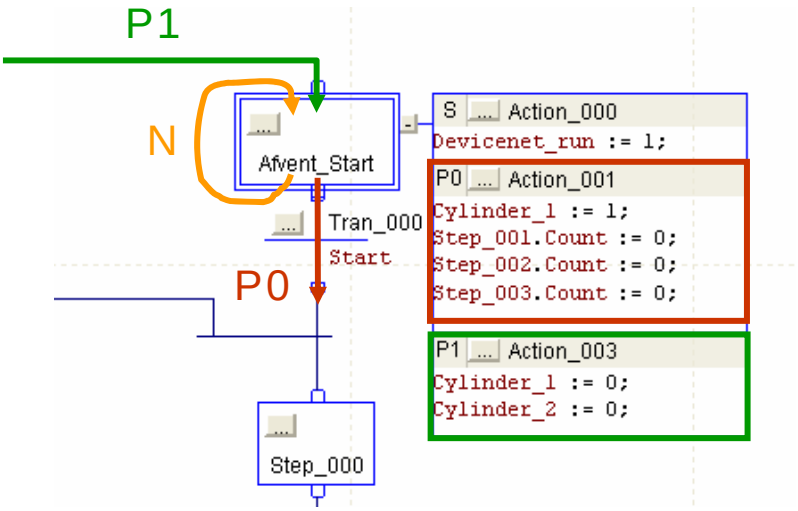
Der startes her
(dobbelt streg)

Action - Properties

N	Non-Stored
R	Reset
S	Stored
L	Time Limited
D	Time Delayed
P	Pulse
P1	Pulse (Rising Edge)
P0	Pulse (Falling Edge)



Sekvensstyring med SFC .. Fortsat 1



Action - Properties

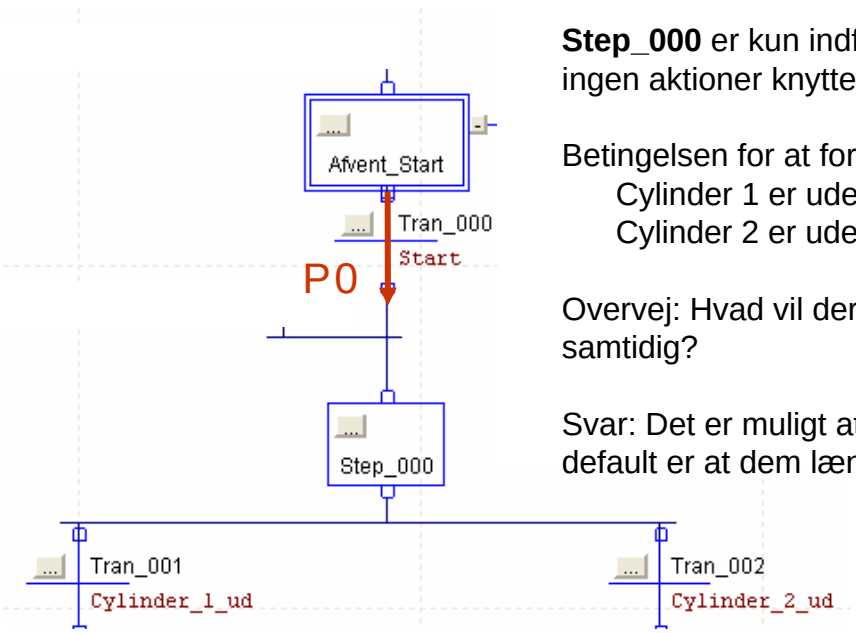
N	Non-Stored
R	Reset
S	Stored
L	Time Limited
D	Time Delayed
P	Pulse
P1	Pulse (Rising Edge)
P0	Pulse (Falling Edge)

En SFC styring består af et antal **Step** (Tilstande, Faser eller hvad nu har lyst til at kalde det). For hvert **Step** kan der defineres et antal aktioner som kan udføres i forbindelse med steppet.

P1 – Aktioner udføres når man ankommer til steppet (det bliver aktiveret igen / første gang)
P0 - Aktioner udføres når man forlader steppet (Skiftebetingelsen er opfyldt)
S - En Stored aktion vil blive udført hele tiden – også når Steppet forlades.
N - En Non-Stored aktion udføres kun sålænge at Steppet er aktivt.
Læs mere om **R**, **L**, **D** og **P** Aktioner i dokumentationen.

Afvent_Start:

S – Aktionen sørger for at Devicenet kører
P1 – Aktionen sender Cylinder 1 og 2 ind (så der klar til næste start)
P0 – Aktionen sender Cylinder 1 ud og initialiserer diverse tællere.



Afvent_Start forlades når der trykkes på start knappen Dette medfører at **Tran_000** bliver sand og samtidig med at Steppet forlades udføres **P0 – Aktionen**.

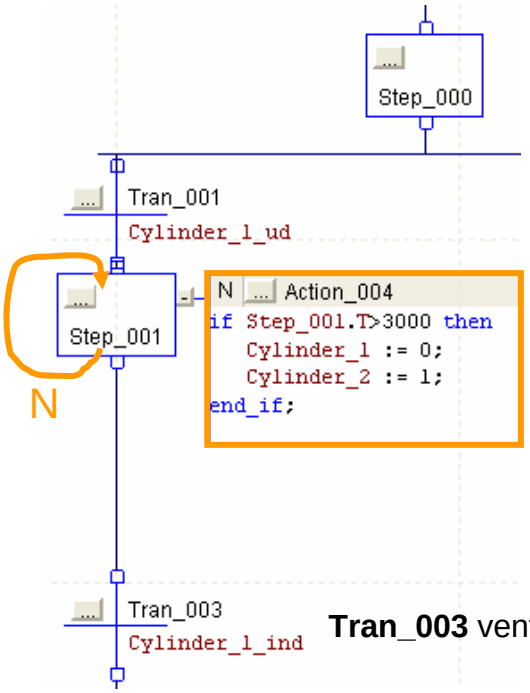
Step_000 er kun indført af praktiske årsager – og der er ingen aktioner knyttet til dette.

Betingelsen for at forlade **Step_000** er at enten:
Cylinder 1 er ude (Tilfældet den første gang) eller at Cylinder 2 er ude (Tilfældet den anden gang)

Overvej: Hvad vil der ske hvis begge Cylindre var ude samtidig?

Svar: Det er muligt at sætte prioritet på en transition men default er at dem længst til venstre har højest prioritet.

Sekvensstyring med SFC .. Fortsat 2



Step_001 vil blive aktivt lige efter at Cylinder 1 er ude.

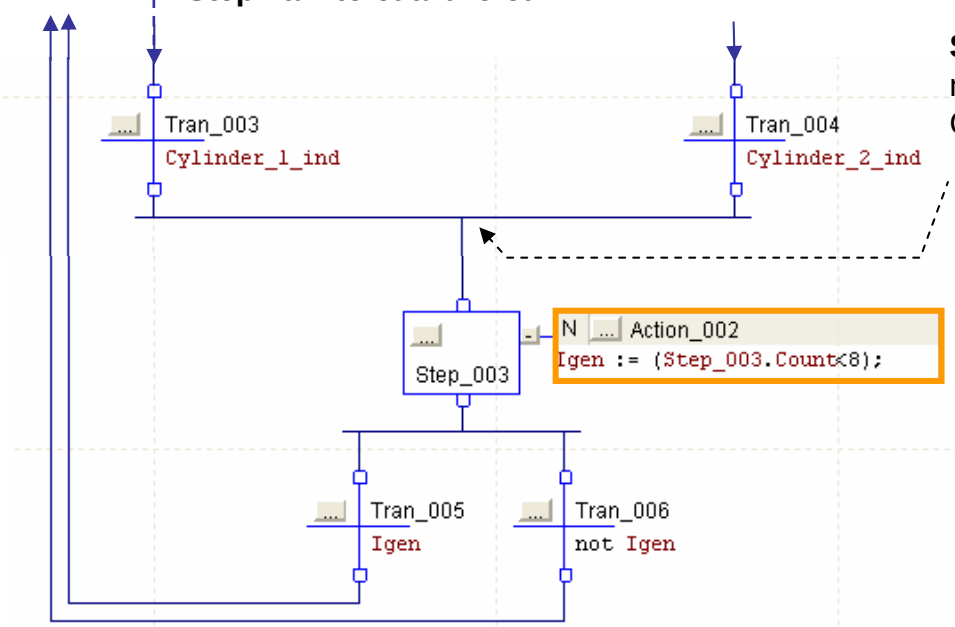
Samtidig med at **Step_001** bliver aktiv startes der en intern timer (**Step_001.T**) som fortæller hvor lang det pågældende step har været aktivt.

N – Aktionen udføres sålænge **Step_001** er aktivt og som det fremgår bliver timeren brugt til at lave en tidsforsinkelse på 3000 msek (3 sekunder)

Når de 3 sekunder er gået sendes Cylinder 1 ind og Cylinder 2 sendes ud.

Tran_003 venter på Cylinder 1 kommer ind igen

Til hvert **Step** er knyttet en tæller som fortæller hvor mange gange det pågældende **Step** har været aktiveret.

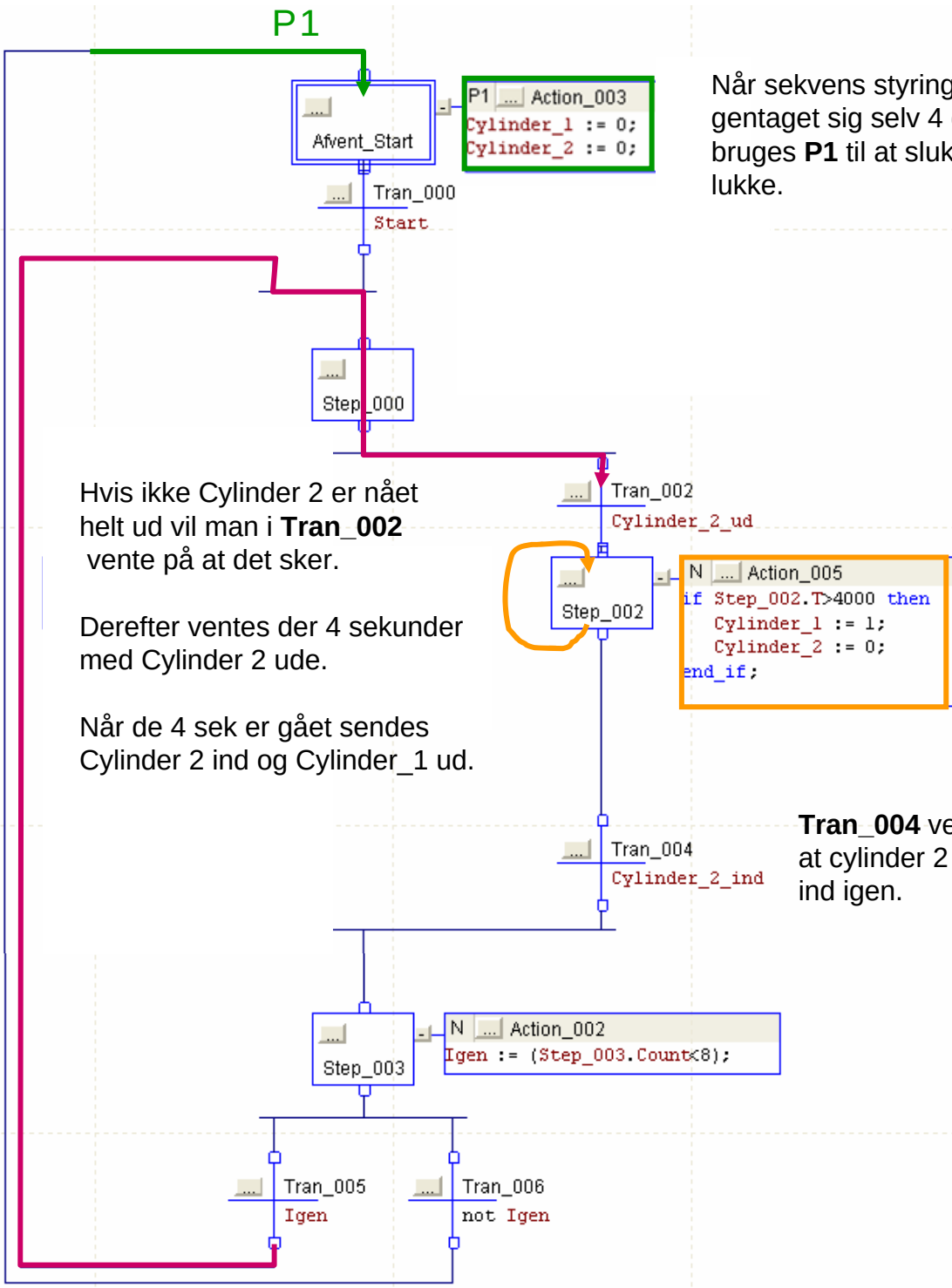


Step_003 vil blive aktiveret når enten Cylinder 1 eller Cylinder 2 er kommet ind.

N - Aktionen i **Step_003** bruges til at afgøre om **Step_003** har været aktiveret mindre end 8 gange

Hvis **Step_003** har været aktiveret mindre end 8 gange er ”Igen” sand og ellers er ”Igen” falsk. (Det vil **Trans_006** vælges p.gr.a. ”not Igen”)

Sekvensstyring med SFC .. Fortsat 3



Når sekvens styringen har gentaget sig selv 4 gange bruges **P1** til at slukke og lukke.

Hvis ikke Cylinder 2 er nået helt ud vil man i **Tran_002** vente på at det sker.

Derefter ventes der 4 sekunder med Cylinder 2 ude.

Når de 4 sek er gået sendes Cylinder 2 ind og Cylinder_1 ud.

Tran_004 venter på at cylinder 2 kommer ind igen.

Konklusion: Dette eksempel viser ikke alle muligheder ved SFC.
For eksempel er det muligt at starte flere parallele forløb (med en dobbelt streg vandret) og alle disse forløb skal så være færdige før der kan fortsættes.

SFC giver altså mulighed for at lave ret advancerede sekvens på et højt niveau og har desuden den fordel at man relativt let kan læse og forstå diagrammet og samtidig bruge dette ved fejlfinding.