BLOK 3

Basis - besturingsfuncties

Veel besturingsfuncties zijn een combinatie van een zeer beperkt aantal basisfuncties: de JA-, de NIET-, de EN- en de OF-functie. Deze fundamentele bouwelementen kunnen worden uitgevoerd met de elektrische relaistechniek, met elektronische poortschakelingen of met pneumatische ventielen. Zoals in de vorige blokken werd beschreven, worden deze ontwerpmethodes ook als basis gebruikt bij het opstellen van PLC-programma's.

In dit blok gebruiken we hoofdzakelijk in- en uitgangen. Bedoeling is om op een eenvoudige manier de verschillende functies te testen met een programmeerapparaat en een PLC. Men krijgt voortdurend een inzicht in de toestand van de PLC functies, doordat de in- en uitgangen van LED-aanduidingen voorzien zijn.

Om didactische redenen is het raadzaam om deze verschillende functies niet uitsluitend theoretisch te benaderen. Men beheerst ze grondiger wanneer ze ook met een programmeerapparaat, waarbij een PLC toestel is aangesloten, worden verwerkt.
De bestandsnaam wordt iedere keer gewijzigd. Op die manier worden de verschillende oefeningen op diskette bewaard en kunnen ze met een printer worden afgedrukt.

Bij het opstellen van een programma kan men dit best eerst op de harde schijf of diskette plaatsen. Nadien kan het programma naar de PLC (AG) worden overgedragen (download).
Het overdragen van een programma van het geheugen van het programmeerapparaat naar de PLC kan alleen als de PLC in de STOP-toestand staat.

Voor het verwerken van de alinea's 1 tot en met 9 raden wij u aan om schakelaars of drukknoppen met NO-contacten (in rust open contact) aan de ingangen van de PLC aan te sluiten. Deze hint is enkel op didactische overwegingen gebaseerd.

Enkel voor klasgebruik

Doelstellingen

- De verschillende besturingsfuncties zoals identiteit, inversie, EN- en OF-functie bij het opstellen van PLC-programma's kunnen toepassen
- Verklaren waarom het NG-contact van detectoren aan de PLC moeten worden aangesloten, als het activeren van de detector onderbrekingen moet veroorzaken
- Het doel en gebruik van de stack kunnen verklaren
- Bij het opstellen van PLC-programma's, de voorrangsregels tussen EN-functie en OF-functie juist toepassen
- Op verschillende manieren programma's opstellen, waarbij de geheugenwerking door de PLC wordt gerealiseerd
- De functie van flankdetectie en dynamische verwerking van ingangssignalen verklaren

1 JA-functie of identiteit

Een logische besturingsfunctie wordt **identiteit** genoemd als het uitgangssignaal dezelfde toestand aanneemt als het ingangssignaal.

Deze logische functie kan op verschillende manieren worden gerealiseerd.
Fig. 3.1 toont de identiteit met een relaisschakeling uitgevoerd.
Fig. 3.2 toont dezelfde functie voorgesteld door een logisch symbool.

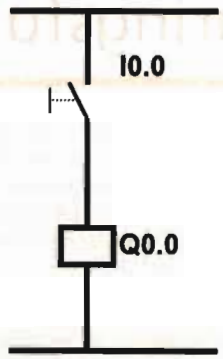


Fig. 3.1 JA-functie met relaisschakeling

Fig. 3.2 Logisch functiesymbool voor een identiteit

Enkel voor klasgebruik

Is de schakelaar I0.0 open, dan wordt het relais niet bekrachtigd. Het uitgangssignaal Q0.0 van de besturing is dan laag. Het commando dat vanuit de besturing naar het aandrijfmechanisme vertrekt, bezit bijgevolg een laag niveau. Het is gebruikelijk om een logische gedachtegang op een eenvoudige manier voor te stellen. Daarom kent men aan een verschijnsel dat "niet waar" is of met "neen" wordt beantwoord, de waarde "0" toe. Is het fenomeen "waar" of moet met "ja" geantwoord worden, dan wordt de waarde "1" gegeven.

Waarheidstabel

Deze logische redenering kan ook nog in een waarheidstabel worden samengevat. Dit is een tabel waarin in de eerste kolommen alle mogelijke digitale waarden aan de ingangssignalen worden gegeven. In de laatste kolom schrijft men de toestand die de uitgangssignalen voor elke ingangscombinatie aannemen. Voor het gegeven geval is dit zeer eenvoudig omdat men beschikt over een digitale functie die slechts één ingang en één uitgang bezit.
Voor ons voorbeeld ziet de waarheidstabel er als volgt uit:

Ingangssignaal I0.0		**Uitgangssignaal Q0.0**	
Schakelaar I0.0 is open	0	Relais is niet bekrachtigd	0
Schakelaar I0.0 is gesloten	1	Relais is bekrachtigd	1

Meestal gebruikt men de verkorte schrijfwijze:

I0.0	**Q0.0**
0	0
1	1

STL

Bij de STL-voorstelling begint elke nieuwe instructie met het laadbevel.
Opdat de CPU het programma zou herkennen en programmabewerking mogelijk zou worden, moet er duidelijk een programmaeinde worden geschreven. Dit kan door een commando MEND (Main Program End). De MEND-instructie moet in een nieuw Network worden ingegeven.
Het programma, dat beantwoordt aan bovenstaande waarheidstabel, is:

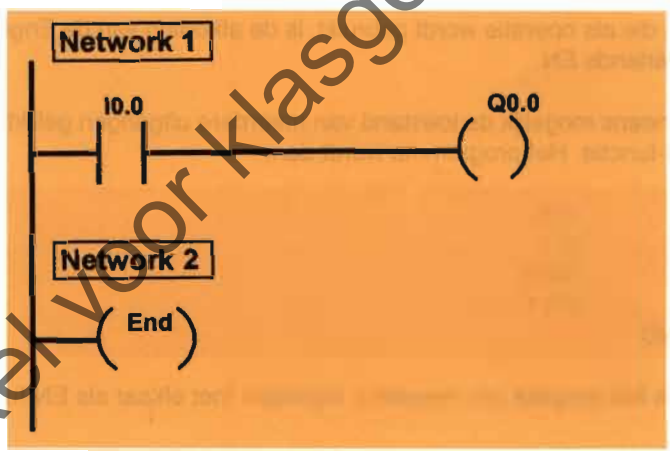
```
LD      I0.0
=       Q0.0
MEND
```

Het is ook mogelijk de toestand van één ingang gelijktijdig aan de toestand van meerdere uitgangen toe te wijzen. Beide uitgangen nemen alleen de toestand van I0.0 over. Het programma wordt dan:

```
LD      I0.0
=       Q0.0
=       Q0.1
MEND
```

LAD

De identiteit kan ook in de vorm van een ladderdiagram worden geprogrammeerd en/of voorgesteld.
Om een identiteit voor te stellen wordt volgend ladderdiagram gebruikt:

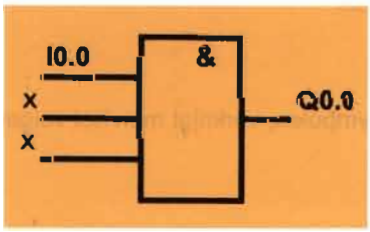


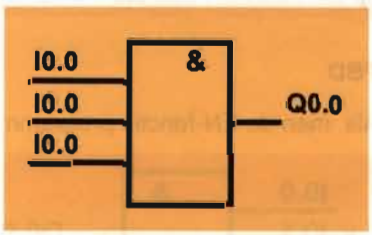
Enkel voor klasgebruik.

Bij dit eerste programmavoorbeeld wordt het programmaeinde weergegeven. Bij de volgende besturingsfuncties is deze instructie eveneens noodzakelijk maar wordt 'END' niet iedere keer herhaald.

FBD

Als de PLC beschikt over de identiteitsfunctie dan maakt men gebruik van het logisch symbool zoals hieronder voorgesteld. Om het aantal symbolen te beperken is het logisch symbool van een identiteit bij sommige PLC's niet voorzien. Om deze basisfunctie te programmeren maakt men gebruik van het EN- of OF-functie met één ingang en één uitgang.





Bij de niet gebruikte ingangen wordt een x-teken toegevoegd. Dit duidt aan dat deze ingangen van de EN-poort niet operationeel zijn. Men kan ook dezelfde ingang vermelden bij elke aansluiting.
Dit geeft hetzelfde resultaat.

2 EN-functie

Een besturingsfunctie wordt een EN-functie genoemd, als het uitgangssignaal de toestand "1" aanneemt, terwijl alle ingangssignalen de toestand "1" bezitten.

Waarheidstabel

Het resultaat van de EN-functie wordt weergegeven in de onderstaande waarheidstabel:

I0.0	I0.0	Q0.0
0	0	0
0	1	0
1	0	0
1	1	1

STL

```
LD      I0.0
A       I0.1
=       Q0.0
MEND
```

De letter A die als operatie wordt gebruikt, is de afkorting van de Engelse aanduiding AND. AND betekent in het Nederlands EN.

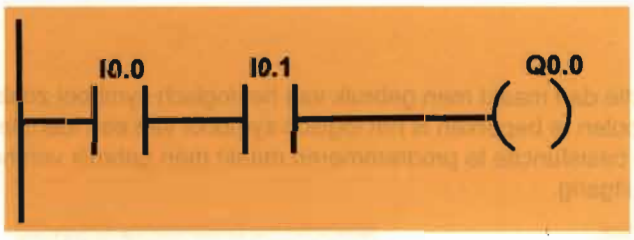
Het is eveneens mogelijk de toestand van meerdere uitgangen gelijktijdig te koppelen aan het resultaat van de EN-functie. Het programma wordt dan:

```
LD      I0.0
A       I0.1
=       Q0.0
=       Q0.1
MEND
```

Uiteraard is het mogelijk om meerdere ingangen met elkaar als EN-functie te verknopen.

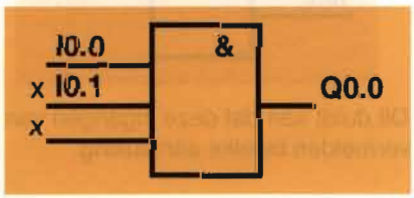
LAD

De hoger vermelde waarheidstabel kan ook in de vorm van een ladderdiagram worden geprogrammeerd en/of voorgesteld. Men verkrijgt dan de volgende voorstelling:



FBD

Als men de EN-functie programmeert met logische symbolen, verkrijgt men het volgende schema:



Enkel voor klasgebruik.

3 OF-functie

Een besturingsfunctie wordt OF-functie genoemd, als het uitgangssignaal een toestand "1" aanneemt terwijl minstens één van de ingangssignalen de toestand "1" bezit.

Waarheidstabel

Het resultaat van de OF-functie wordt weergegeven in de onderstaande waarheidstabel:

I0.0	I0.0	Q0.0
0	0	0
0	1	1
1	0	1
1	1	1

STL

Het programma dat overeenkomt met bovenstaande waarheidstabel wordt:

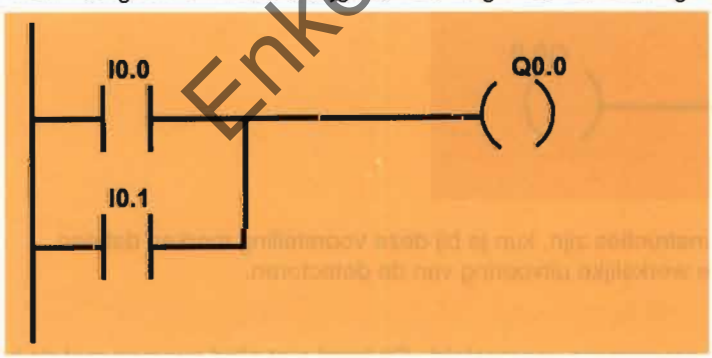
```
LD      I0.0
O       I0.1
=       Q0.0
MEND
```

De letter O die als operatie wordt gebruikt, is de afkorting van de Engelse aanduiding OR. OR betekent in het Nederlands OF.

Het is eveneens mogelijk de toestand van meerdere uitgangen gelijktijdig te koppelen aan de toestand van een ingang. Het is uiteraard mogelijk om meerdere ingangen met elkaar als OF-functie te verknopen.

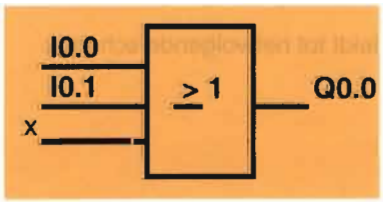
LAD

De hoger vermelde waarheidstabel kan ook in de vorm van een ladderdiagram worden geprogrammeerd en/of voorgesteld. Men verkrijgt dan de volgende voorstelling:



FBD

Als men de OF-functie programmeert met logische symbolen verkrijgt men het volgende schema:



De letter "x" duidt aan dat deze ingang niet operationeel is.

Enkel voor klasgebruik.

4 NIET-functie of inversie

4.1 Inversie van operand

Een logische besturingsfunctie wordt een inversie of negatie genoemd als het uitgangssignaal steeds de tegenovergestelde logische waarde aanneemt van het ingangssignaal.

Waarheidstabel

De werking van deze schakeling kan voorgesteld worden door onderstaande waarheidstabel:

I0.0	Q0.0
0	1
1	0

STL

Om weer te geven dat de toestand van het signaal moet geïnverteerd worden, gebruikt men de aanduiding "N". De letter N wordt gebruikt als afkorting van het Engelse woord NOT. Net zoals bij de identiteit moet de instructie als een volledig geheel worden ingegeven; men begint een nieuw programma met de instructie LOAD.

```
LDN     I0.0
=       Q0.0
MEND
```

Het is eveneens mogelijk het inverse signaal te gebruiken in een EN-functie of OF-functie. Men gebruikt dan de instructie AN (and not)of ON (or not).

LAD

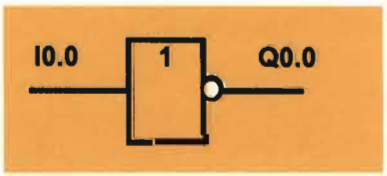
Wanneer men de inversie programmeert in de vorm van een ladderdiagram, dan verkrijgt men de volgende voorstelling:

Alhoewel het zeer eenvoudige basisinstructies zijn, kun je bij deze voorstelling merken dat een programma geen weergave is van de werkelijke uitvoering van de detectoren.

Let op:
In een ladderdiagram wordt het PLC-programma voorgesteld. Dit komt niet altijd overeen met de fysische toestand van de ingangen die aangesloten zijn aan de ingangen van de PLC.

FBD

Het behandelde programma voorgesteld met logische symbolen, leidt tot het volgende schema:



Enkel voor klasgebruik.

4.2 Inversie van instructie

Met de NIET-instructie kan men om het even welk resultaat van één Network inverteren. Het berekende resultaat wordt omgekeerd, dit wil zeggen de toestand "0" wordt "1"; de toestand "1" wordt "0". Personen met een elektronische vooropleiding kennen zeker de NAND (NIET EN) en de NOR (NIET OF) instructie. Door gebruik te maken van de "NOT"-instructie kan men hetzelfde resultaat bereiken.

NIET EN

Waarheidstabel

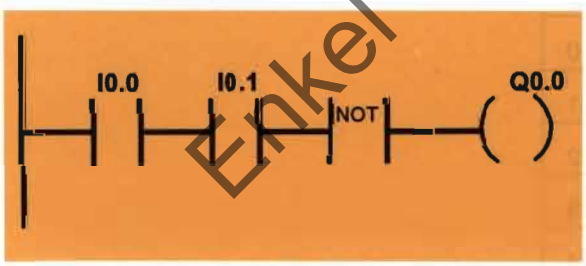
I0.0	I0.0	EN	NEN
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

STL

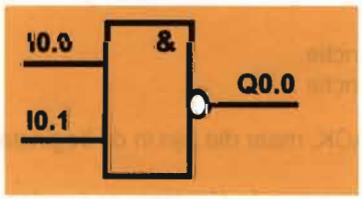
Het programma dat overeenkomt met bovenstaande waarheidstabel wordt:

```
LD      I0.0
A       I0.1
NOT
=       Q0.0
MEND
```

LAD



FBD



NIET OF

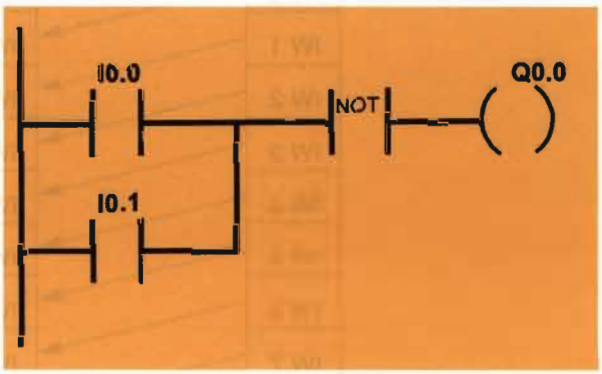
Waarheidstabel

I0.0	I0.0	OF	NOF
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

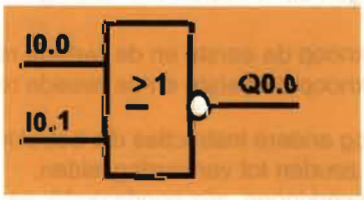
Het programma dat overeenkomt met bovenstaande waarheidstabel wordt:

```
LD      I0.0
O       I0.1
NOT
=       Q0.0
MEND
```

LAD



FBD



Enkel voor klasgebruik.

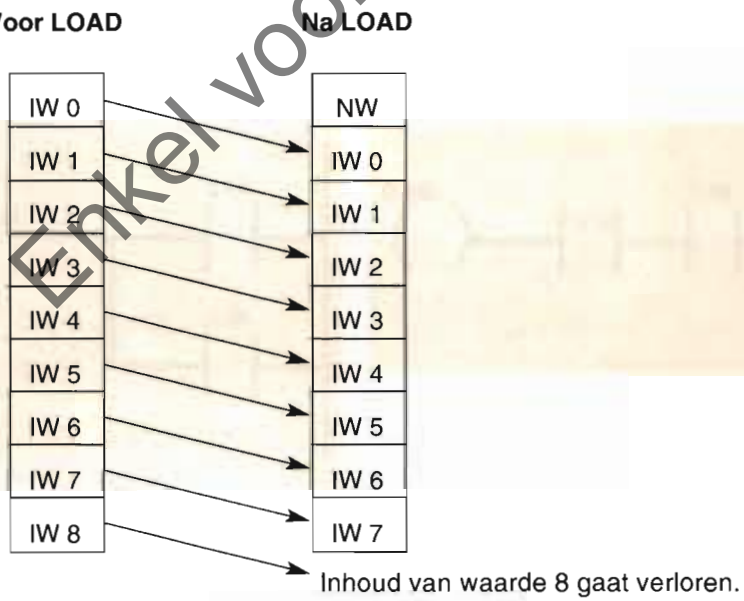
5 Prioriteitsregels

5.1 Bewerkingen met logisch register

Stack is de benaming die men gebruikt om een groep geheugenplaatsen aan te duiden die door de CPU wordt gebruikt om het programma te verwerken. Alleen wanneer men een programma opstelt met de STL programmeertaal moet men rekening houden met de stack. Als men een programma opstelt met de programmeertaal LAD, zal de CPU zelf de geheugenplaatsen organiseren. Deze geheugenplaats is 9 bits diep en 1 bit breed. De Nederlandse vertaling van stack komt overeen met een stapel. En dit verklaart enigszins het gebruik. Bij elke LOAD operatie wordt de meest recente waarde boven op de stapel geplaatst. Schematisch kan men een stack best als volgt voorstellen:

S0	Stack 0	1ste niveau
S1	Stack 1	2de niveau
S2	Stack 2	3de niveau
S3	Stack 3	4de niveau
S4	Stack 4	5de niveau
S5	Stack 5	6de niveau
S6	Stack 6	7de niveau
S7	Stack 7	8ste niveau
S8	Stack 8	9de niveau

De meeste instructies verwerken alleen de logische toestanden die voorkomen in het programma. Wanneer de combinatie van EN- en OF-functie het noodzakelijk maken dat er tussenresultaten tijdelijk worden onthouden, dan worden die in het stack-register bewaard.
In de volgende figuur worden de volgende afkortingen gebruikt:
IW staat voor initiële waarde;
NW staat voor nieuwe waarde.



Om bewerkingen uit te voeren met deze geheugenplaatsen, maakt men gebruik van de volgende instructies:

- ALD verknoop de eerste en de tweede registerplaats als EN-functie
- OLD verknoop de eerste en de tweede registerplaats als OF-functie

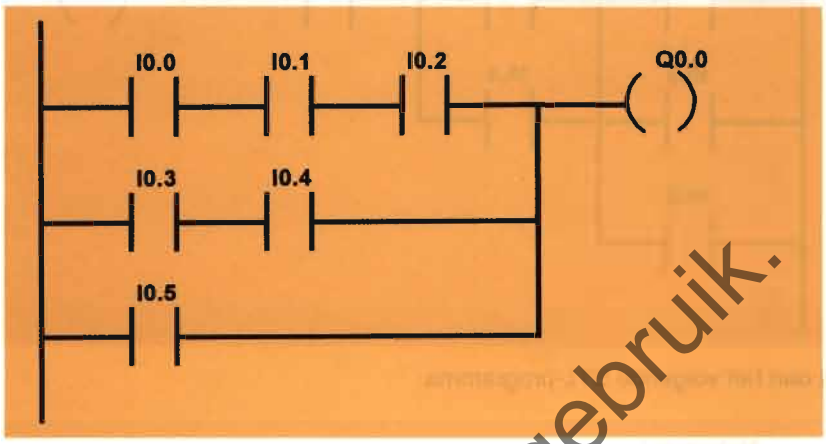
Er bestaan nog andere instructies die betrekking hebben op de STACK, maar die zijn in dit beginstadium overbodig en zouden tot verwarring leiden.
Deze stackbewerkingen zijn noodzakelijk om vertakkingen in het geheugen te plaatsen; ze worden door de CPU sneller verwerkt dan met andere programmeermethodes die hetzelfde doel nastreven.

Enkel voor klasgebruik.

5.2 EN- voor OF-functie

Bij complexere besturingen nemen de schakelvoorwaarden steeds ingewikkelde vormen aan. Bij nader onderzoek blijkt dat zij meestal slechts bestaan uit combinaties van drie basisfuncties : de NIET-, de EN-, en de OF-functie.
Wanneer men deze functies bij elkaar gebruikt dan heeft de NIET-functie voorrang op de EN-functie, en de EN-functie voorrang op de OF-functie.

De EN- voor OF-functie komt overeen met een parallelschakeling van meerdere seriecontacten in de relaistechniek. In onderstaand voorbeeld krijgt uitgang Q0.0 de toestand "1" (is ingeschakeld) indien in minstens één van de takken alle seriecontacten gesloten zijn (de toestand "1" hebben) of als het enkelvoudig contact I0.5 gesloten is (de toestand "1" heeft).

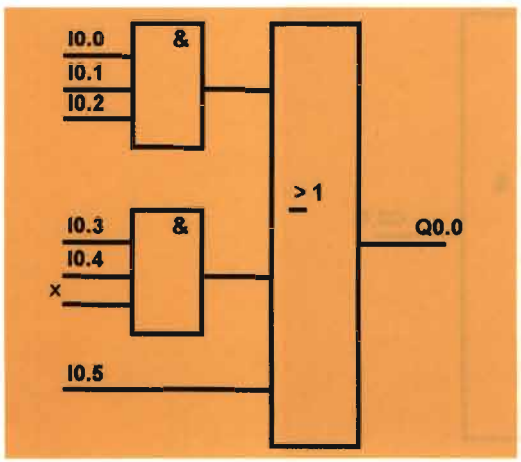


Men verkrijgt dan het volgende STL-programma:

```
Network 1
        LD      I0.0
        A       I0.1
        A       I0.2
        LD      I0.3
        A       I0.4
        OLD
        O       I0.5
        =       Q0.0
Network 2
        MEND
```

Eerst worden de EN-functies verwerkt van de ingangen I0.0, I0.1 en I0.2. Het resultaat wordt bewaard in het stackregister. Een nieuwe LOAD-instructie zorgt voor een tweede verknoping waarbij de ingangen I0.3 en I0.4 als EN verknoopt worden. Het resultaat wordt weer "bovenop" de geheugenplaats (stack) gelegd. Met de instructie "OLD" worden beide resultaten met elkaar verknoopt. Vervolgens wordt dit resultaat als OF verknoopt met de toestand van ingang I0.5. Het resultaat van deze bewerkingen wordt overgedragen naar de uitgang Q0.0.

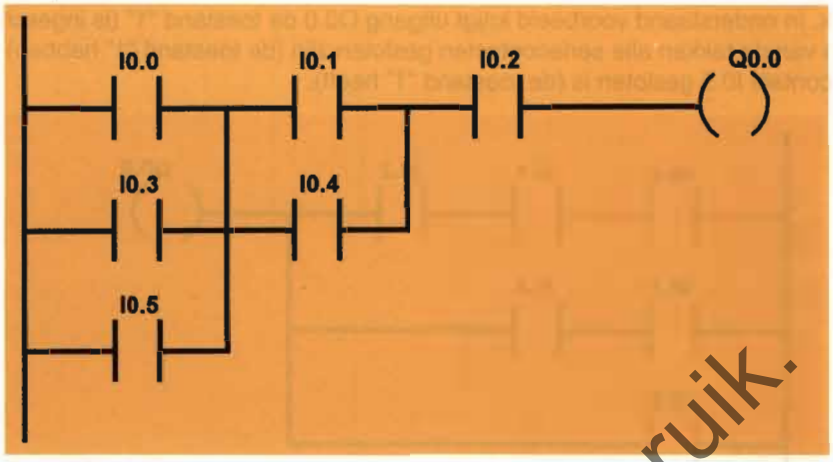
Wanneer men hetzelfde programma opstelt door gebruik te maken van logische symbolen, dan verkrijgt men:



Enkel voor klasgebruik.

5.3 OF- voor EN-functie

In de vorige paragraaf werd bekeken hoe men het programma moet opstellen om de EN-functie voorrang te geven op de OF-functie. Er zijn eveneens schakelingen waarbij de OF-functie voorrang moet hebben op de EN-functie. Deze schakeling komt in relaistechniek overeen met een serieschakeling van meerdere parallelcontacten.
In onderstaand schema krijgt uitgang Q0.0 de toestand "1", indien in elke paralleltak minstens één contact gesloten is (toestand "1" heeft) en, als het enkelvoudig contact I0.2 gesloten is (toestand "1" heeft).

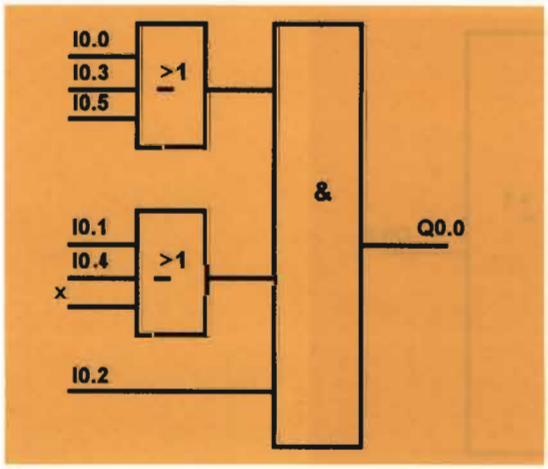


Men verkrijgt dan het volgende STL-programma:

```
Network 1
    LD      I0.0
    O       I0.3
    O       I0.5
    LD      I0.1
    O       I0.4
    ALD
    A       I0.2
    =       Q0.0
Network 2
    MEND
```

Eerst worden de OF-functies van de ingangen I0.0, I0.3 en I0.5 verwerkt. Het resultaat wordt bewaard in het stackregister. Een nieuwe LOAD-instructie zorgt voor een tweede verknoping waarbij de ingangen I0.1 en I0.4 als OF worden verknoopt. Het resultaat wordt weer "bovenop" de geheugenplaats (stack) gelegd. Met de instructie "ALD" worden beide resultaten met elkaar verknoopt. Vervolgens wordt dit resultaat als EN verknoopt met de toestand van ingang I0.2. Het resultaat van deze bewerkingen wordt overgedragen naar de uitgang Q0.0.

Wanneer men hetzelfde programma opstelt door gebruik te maken van logische symbolen, dan verkrijgt men:



Enkel voor klasgebruik.

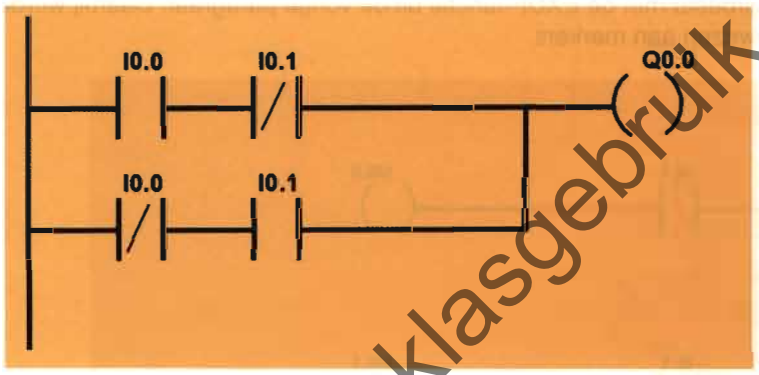
6 EXOF-functie

De exclusieve OF-verbinding is niet als basisfunctie bij het programmeren van elke PLC aanwezig. Deze functie is eenvoudig uit een combinatie van EN- en OF-functies samen te stellen. Het is een toepassing van de EN-functie die voorrang heeft op de OF-functie.

Bij een EXOF-functie heeft de uitgang alleen de toestand "1" als één van beide ingangen de toestand "1" heeft. Hebben de beide ingangen de toestand "1" dan wordt de toestand van de uitgang "0".
Het resultaat van de EXOF-functie wordt weergegeven in onderstaande waarheidstabel:

I0.0	I0.1	Q0.0
0	0	0
0	1	1
1	0	1
1	1	0

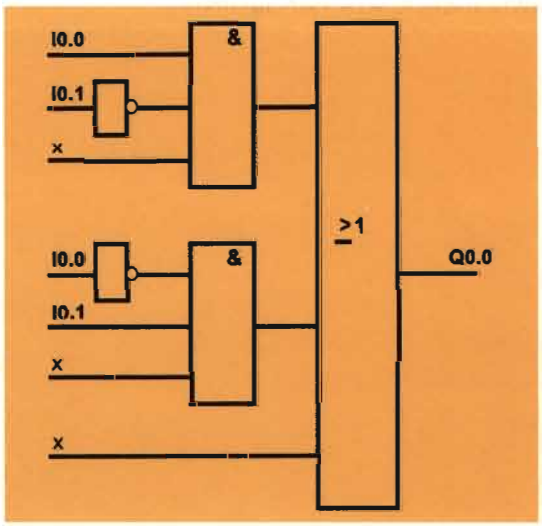
Onderstaand schema komt overeen met een EXOF functie.



Geeft men het programma weer in de vorm van een STL-instructielijst dan verkrijgt men:

```
Network 1
      LD      I0.0
      AN      I0.1
      LDN     I0.0
      A       I0.1
      OLD
      =       Q0.0
Network 2
      MEND
```

Maakt men gebruik van de logische symbolen dan zal het schema er als volgt uitzien:



Enkel voor klasgebruik.

7 Merkers

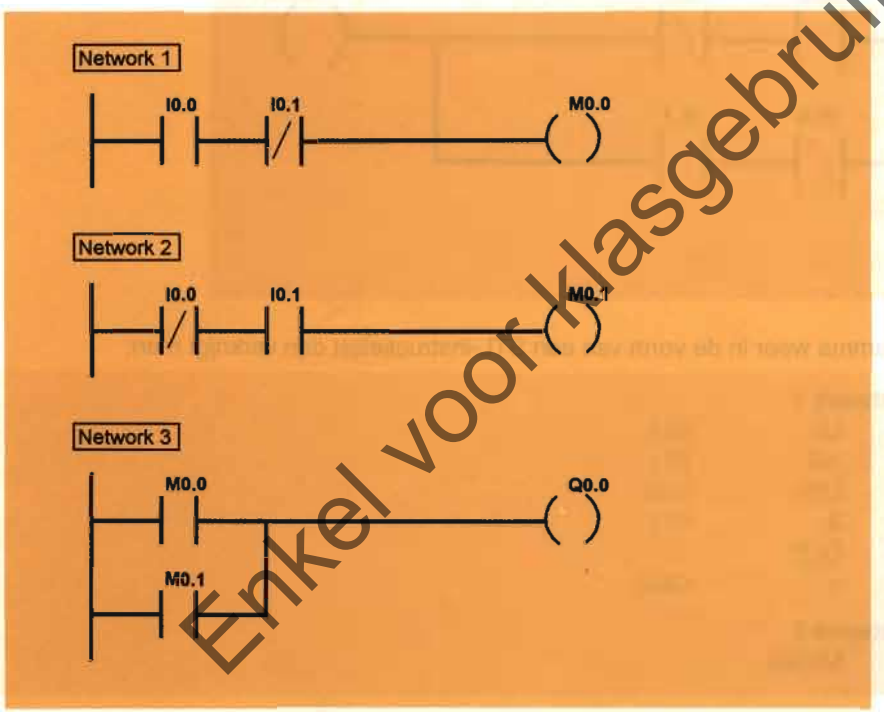
Het is in omvangrijke binaire functies soms nodig de tussentijdse resultaten te bewaren. Zo kunnen die ook in andere gedeelten van het programma worden gebruikt. Voor het bewaren van dergelijke binaire tussenresultaten zijn de merkers beschikbaar. In blok 2, paragraaf 4.8 werd uitvoerig op het doel en de uitvoeringsvormen (remanent en niet-remanent) ingegaan. In deze paragraaf gaan we dieper in op de manier waarop we dergelijke merkers programmeren.

Gebruik maken van merkers bij het opstellen van programma's heeft een aantal voordelen:

- De verbindingen worden overzichtelijker en zijn eenvoudiger te programmeren.
- Het testen van tussenresultaten, en daarmee het zoeken van fouten, wordt eenvoudiger.
- De tussenresultaten, opgeslagen in de merkers, kunnen in andere delen van het programma worden verwerkt.

Merkers worden op dezelfde manier als uitgangen geprogrammeerd. De parameters van de operand zijn op dezelfde wijze als die voor de in- en uitgangen opgebouwd. Merkers worden met de kenletter 'M' aangeduid.
Elk type PLC beschikt over een groot aantal merkers. Daarom is het sterk af te raden één merker voor meerdere verschillende functies in één programma te gebruiken.

We nemen het voorbeeld met de EXOF-functie uit de vorige paragraaf. Daarbij worden delen van het programma toegewezen aan merkers.



Enkel voor klasgebruik.

Wanneer hetzelfde programma opgesteld wordt in STL verkrijgt men:

```
Network 1
    LD      I0.0
    AN      I0.1
    =       M0.0
Network 2
    LDN     I0.0
    A       I0.1
    =       M0.0
Network 3
    LD      M0.0
    O       M0.1
    =       Q0.0
Network 4
    MEND
```

8 Detectoren met breekcontacten

Tot hiertoe werden uitsluitend schakelaars behandeld waarvan de contacten normaal open zijn in rusttoestand. Men steunde hierbij op het arbeidsstroomprincipe. In catalogi worden deze schakelaars met NO-contacten (normaal open contacten) aangeduid.

Wanneer uit veiligheidsoverwegingen een installatie altijd afgeschakeld moet kunnen worden, dan past men het ruststroomprincipe toe. Voor de stopdrukknop, beveiligingsrelais, enz... worden detectoren met **normaal** gesloten contacten gebruikt. In catalogi worden ze als schakelelementen met NG-contacten aangeduid.

Ter illustratie gebruiken we het voorbeeld van een motorsturing.
Aan ingang I0.0 wordt het contact van het thermisch relais aangesloten. Dit schakelelement zorgt ervoor dat bij een te hoge temperatuur de motor automatisch wordt uitgeschakeld. Aan de ingang I0.1 sluit men de schakelaar aan. Is de uitgang Q0.0 actief, dan zal de motor draaien.

De kans dat er zich een fout voordoet in de sturingskast waarin de PLC is ingebouwd, is vrij gering. Deze kast is meestal op een veilige plaats opgesteld. De PLC en bijbehorende bedrading zitten beschermd in een kast met een hoge beschermingsgraad tegen stof, binnendringend vocht en mechanische beschadiging. Het komt veel frequenter voor dat de kabel tussen deze sturingskast en een detector wordt onderbroken.

We veronderstellen, in eerste instantie, dat beide detectoren met een NO-contact aan de ingangen van de PLC zijn verbonden.

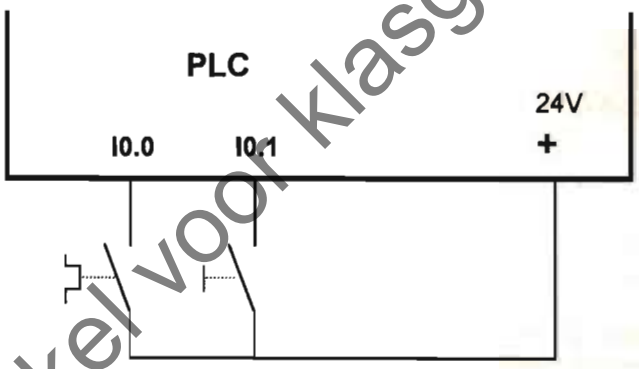


Fig. 3.3 Aansluiten van detectoren met NO-contacten aan de ingangen van de PLC

Enkel voor klasgebruik.

Stelt men het programma op, dat aan deze voorwaarden voldoet, dan krijgt men

```
LDN     I0.0
A       I0.1
=       Q0.0
```

We nemen aan dat het thermisch relais in rusttoestand de toestand 0 is en dat de motorschakelaar is ingeschakeld, dus de toestand 1 bezit. De uitgang wordt bekrachtigd omdat de ingang I0.0 wordt geïnverteerd.

Onvoorspelbaar treedt er in de draad tussen het thermisch relais en de ingang van de PLC een onderbreking op terwijl de motor draait. De PLC blijft bij ingang I0.0 nog altijd een 0-signaal lezen. In het programma wordt dit verder tot een toestand "1" geïnverteerd; de motor blijft zonder onderbreking draaien. Ook als het thermisch relais zijn schakelcontact door een overstroom omschakelt, merkt de PLC dit niet. De motor blijft draaien.

Op die manier verkrijgt men een programma dat in normale omstandigheden optimaal werkt. Zelfs bij een eventuele draadbreuk merkt men in eerste instantie nog niets. Het is pas op het ogenblik dat het proces moet worden onderbroken, dat men zal vaststellen dat de sturing foutief reageert. Daarom is het noodzakelijk dat men alle onderbrekingsvoorwaarden met een normaal gesloten contact uitvoert.
Dit contact wordt op dezelfde manier als het normaal open contact aangesloten. Alleen bij het opstellen van het programma moet men er rekening mee houden.
Wordt het normaal gesloten contact van het thermisch relais met de ingang I0.0 verbonden, dan krijgen we de volgende aansluitingen:

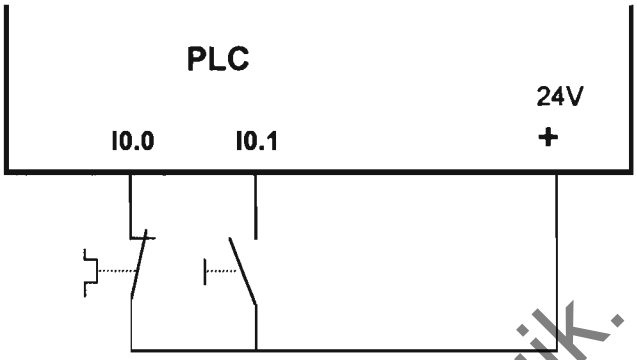


Fig. 3.4 Aansluiten van onderbreking met NG-contact aan ingang van PLC

Het programma wordt dan:

```
LD      I0.0
A       I0.1
=       Q0.0
```

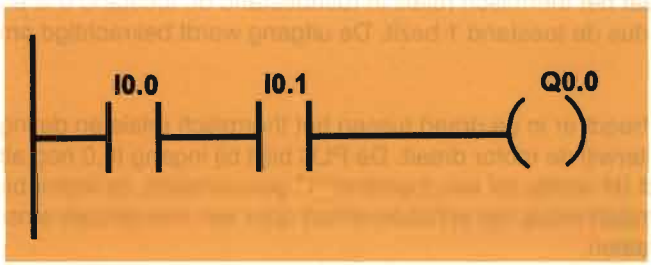
Omdat men hardwarematig de ingang inverteert, mag men in het programma geen inversie meer toepassen. Zolang er geen abnormale temperatuur door het thermisch relais wordt waargenomen, blijft de toestand van ingang I0.0 gelijk aan "1".

Treedt een draadbreuk op, dan zal de ingang van de PLC dezelfde toestand detecteren als bij het activeren van het thermisch relais. In beide gevallen zal de toestand van ingang I0.0 gelijk worden aan "0". Rekening houdend met het hoger vermelde programma zal de PLC de motor uitschakelen.

Zeer belangrijk

Om veiligheidsredenen is men dus verplicht een normaal gesloten contact aan de ingang van de PLC aan te sluiten voor alle beveiligingsfuncties zoals: stopdrukknop, overdrukventiel, thermische beveiliging, enz...; kortom voor alle onderbrekingsvoorwaarden van een proces.
Bovendien zal men alle noodstopfuncties, verplicht, hardwarematig moeten aansluiten. Deze problemen worden later uitvoerig besproken.

Hoewel reeds vroeger vermeld, moet hier nog eens duidelijk benadrukt worden dat het ladderdiagram het PLC-programma visualiseert. Het is echter geen weergave van de fysische karakteristieken van de detectoren. Het programma, met het NG-contact voor het thermisch relais, geeft het volgende ladderdiagram:



Enkel voor klasgebruik.

9 Geheugenfuncties

Met de instructies die tot hiertoe besproken werden, is het mogelijk de meest uiteenlopende combinatorische schakelproblemen te realiseren.
Bij industriële sturingen verloopt een groot deel van de processen in hoofdzaak sequentieel.
Het zal dus noodzakelijk zijn dat het proces bepaalde toestanden in zijn geheugen bewaart.
Bij gebruik van geheugenschakelingen komen drie fasen voor, namelijk:

- Het activeren (setten) van het geheugensignaal
- Het onthouden van het geheugensignaal
- Het deactiveren (resetten) van het geheugensignaal

9.1 Stop voorrang

Een veel voorkomende schakeling in de relaistechniek is een schakeling met houdcircuit. Op deze manier kan, door gebruik te maken van het overneemcontact, een geheugenfunctie worden geprogrammeerd. Rekening houdend met de vroeger vermelde veiligheidsoverwegingen, worden normaal gesloten schakelelementen voor de stopfunctie gebruikt.
Parallel met een commando-element, dikwijls de startvoorwaarde, wordt een contact van het te sturen relais geplaatst. Dat contact neemt, na een éénmalige bediening, de functie van het commando-element over. Deze schakeling wordt voorgesteld door figuur 3.5.

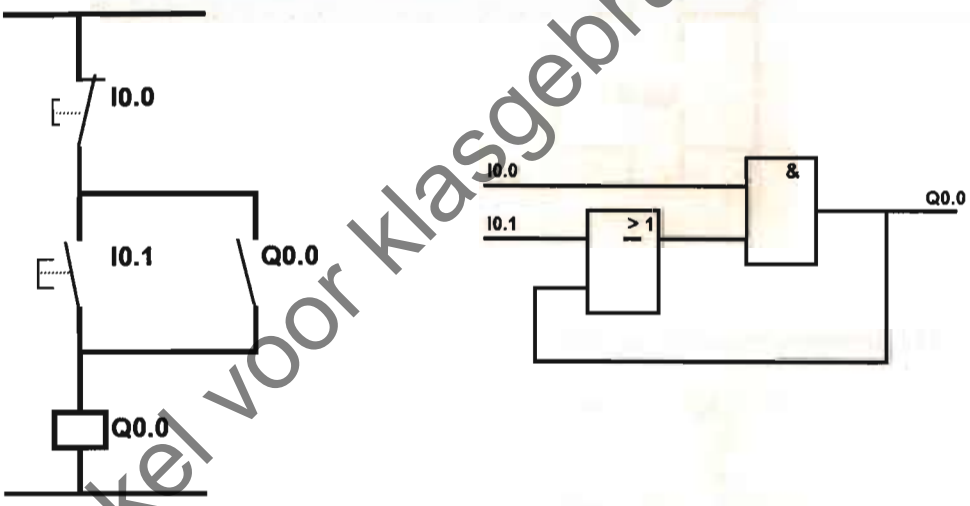
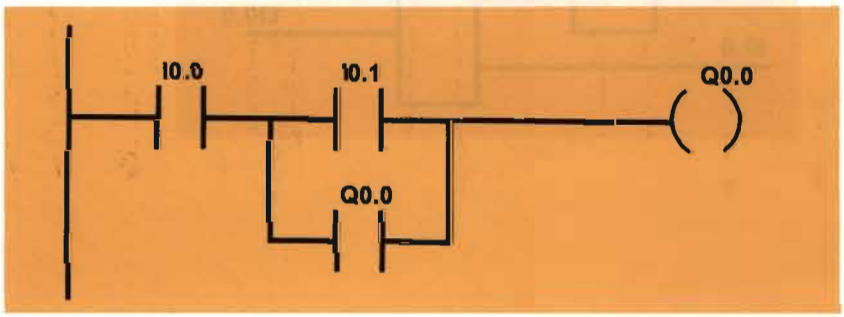


Fig. 3.5 Geheugenschakeling met stop-prioriteit

Enkel voor klasgebruik.

Als de setknop I0.1 wel en de resetknop I0.0 niet wordt bediend, dan zal de uitgang Q0.0 worden bekrachtigd. Hierop sluit het overneemcontact Q0.0. Wordt daarna schakelaar I0.1 terug geopend, dan blijft de uitgang Q0.0 geactiveerd. Dit komt omdat de bekrachtigingsstroom doorgang vindt via het overneemcontact van Q0.0. Pas na het bedienen van de resetknop I0.0 wordt de uitgang Q0.0 gereset en keert het contact Q0.0 naar zijn begintoestand terug.

Bij gelijktijdig bedienen van drukknoppen, aangesloten aan de ingangen I0.0 en I0.1, wordt de uitgang niet bekrachtigd. Met andere woorden kan men stellen dat het resetsignaal prioriteit bezit op het setsignaal. Het ladderdiagram ziet er als volgt uit:

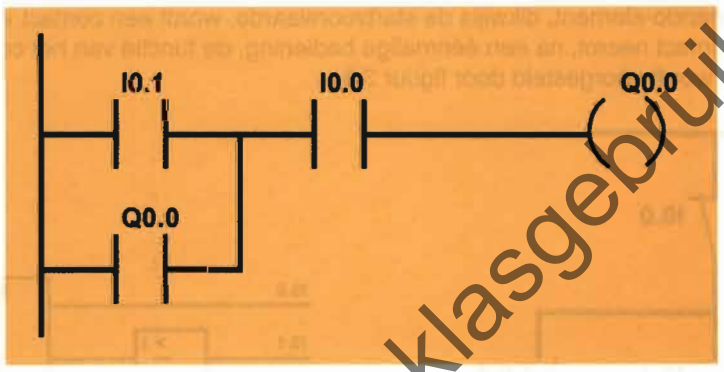


Als men bovenstaand schema vertaalt naar een PLC-programma, opgesteld in STL, dan verkrijgt men:

```
LD      I0.0
LD      I0.1
O       Q0.0
ALD
=       Q0.0
```

Personen die redeneren vanuit een elektrische vooropleiding zijn geneigd om de stopdrukknop uiterst links te plaatsen. Eén van de principes in de klassieke schakeltechniek bestond erin om bij een onderbreking een zo groot mogelijk deel van de installatie spanningsloos te maken.
Bij het programmeren van een PLC is dit principe niet zo essentieel. De programmalijnen worden toch na elkaar berekend en het resultaat wordt in het PIQ-geheugen geplaatst. Aan het einde van de cyclus wordt dit resultaat aan de periferie doorgegeven.

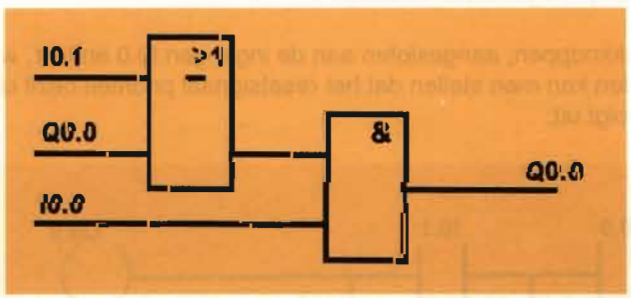
Het STL programma wordt eenvoudiger, als men de stopdrukknop na de parallelvertakking plaatst.
Zo krijgen we het volgende ladderdiagram:



Het overeenkomstige STL-programma wordt:

```
LD      I0.1
O       Q0.0
A       I0.0
=       Q0.0
```

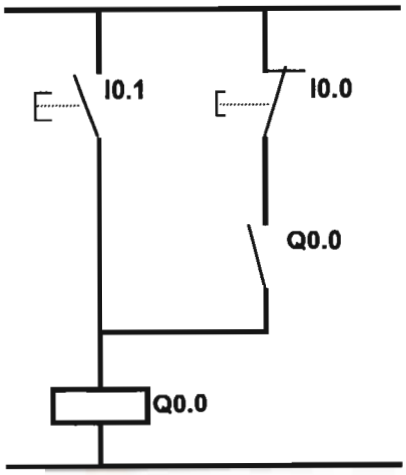
Onmiddellijk valt op dat deze programmeerwijze beknopter is. De werking is volledig identiek aan de vorige versie.
Programmeert men met logische functies, dan gebruikt men het volgende programma:



Enkel voor klasgebruik.

9.2 Start voorrang

De schakeling, voorgesteld in figuur 3.6, is een voorbeeld van een geheugenschakeling met set-prioriteit.



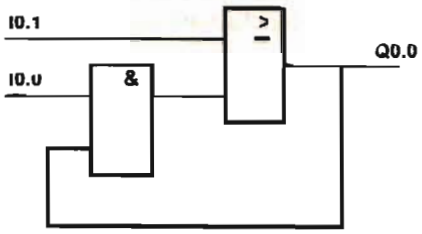
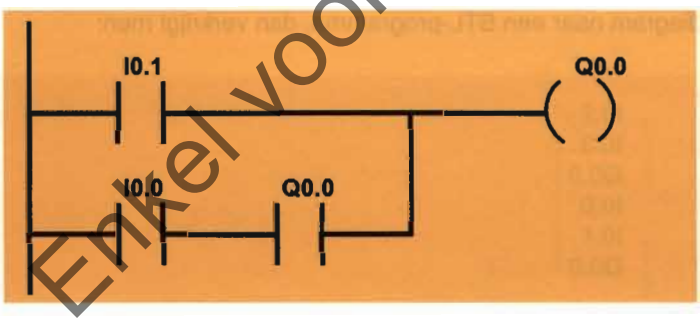


Fig. 3.6 Geheugenschakeling met set-prioriteit

Wordt de startdrukknop, aangesloten aan de ingang I0.1, bediend, dan zal het relais, aangesloten aan de uitgang Q0.0, bekrachtigd worden. Hierdoor wordt het contact Q0.0 gesloten. Na het loslaten van de startdrukknop wordt de stroomkring blijvend gesloten via het overneemcontact Q0.0. De kring blijft gesloten totdat de resetknop I0.0 wordt bediend.

Bij gelijktijdig bedienen van de start- en stopdrukknop wordt de uitgang toch bekrachtigd. De overneem-functie wordt echter niet geactiveerd, tenzij men de startdrukknop vroeger dan de stopdrukknop loslaat. Men moet wel beseffen dat deze schakeling uit veiligheidsoverwegingen verboden is. Ze voldoet niet aan de normen EN 60 204 en EN 292.2 " Richtlijnen arbeidsmiddelen".

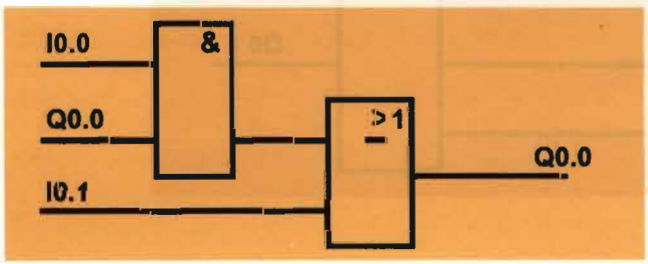
Het ladderdiagram ziet er als volgt uit:



Hetzelfde programma geeft in STL:

```
LD      I0.1
LD      I0.0
A       Q0.0
OLD
=       Q0.0
```

Maakt men gebruik van SFB, dan krijgt men:



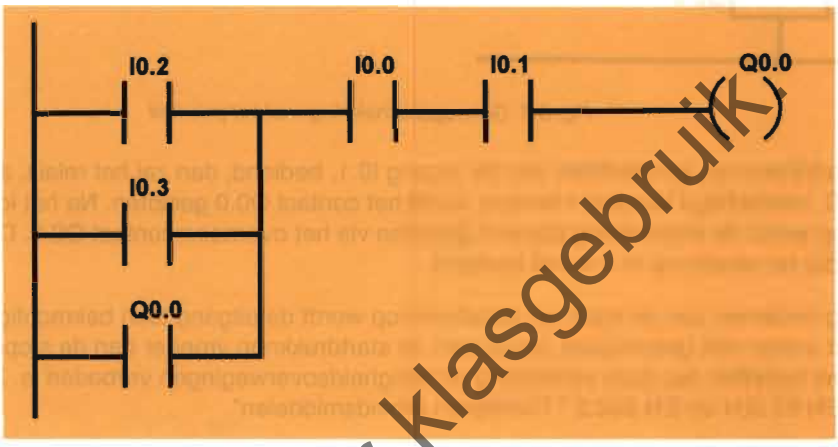
Enkel voor klasgebruik.

9.3 Bediening vanuit meerdere plaatsen

Het is dikwijls noodzakelijk dat een proces vanuit meerdere plaatsen kan worden gestart en gestopt. Omdat in de meeste gevallen de installatie bij het terugkeren naar de rustsituatie in een veilige toestand verkeert, kan men gebruik maken van de volgende algemene regel:

> Alle onderbrekingsfuncties worden als EN-functie met elkaar verknoopt, alle startvoorwaarden worden als OF-functie verknoopt, samen met het overneemcontact.

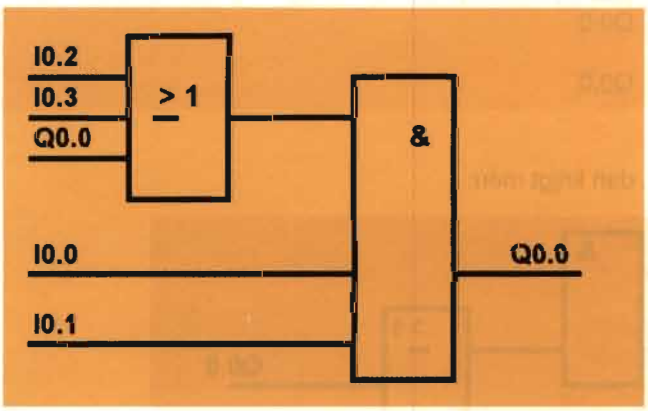
De volgende schakeling wordt gebruikt om een installatie van op 2 plaatsen te starten en te stoppen. Men kan het aantal bedieningsplaatsen uitbreiden door het hoger vermelde principe toe te passen.
De schakelelementen aan de ingangen I0.0 en I0.1 hebben hier een resetfunctie. De detectoren, aangesloten aan de ingangen I0.2 en I0.3 hebben een setfunctie.



Vertaalt men dit ladderdiagram naar een STL-programma, dan verkrijgt men:

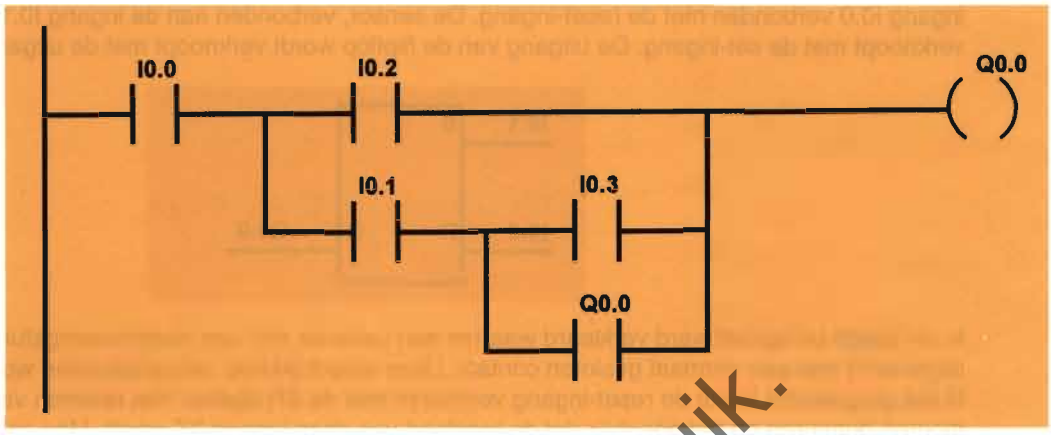
```
LD   I0.2
O    I0.3
O    Q0.0
A    I0.0
A    I0.1
=    Q0.0
```

Maakt men gebruik van SFB dan krijgt men:



Enkel voor klasgebruik

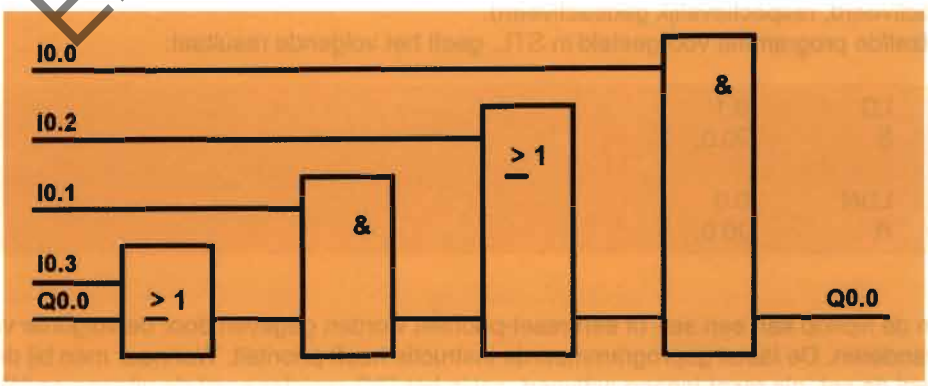
Bij veel industriële besturingen komt het voor dat een proces vanuit meerdere plaatsen wordt gestuurd. Alle bedieningen worden in één centrale plaats gecentraliseerd. Men beschikt er ook over een bedieningspost die in de installatie zelf is opgesteld. Afhankelijk van het te sturen proces kan men zelf bepalen welk van beide bedieningsplaatsen voorrang moet hebben. In onderstaand schema heeft de bedieningspost, met de detectoren aangesloten aan I0.0 en I0.2 voorrang op de bedieningspost met de detectoren I0.1 en I0.3.
Het onderstaande ladderdiagram voldoet aan deze voorwaarden.



Vertaalt men het ladderdiagram naar een STL-programma dan verkrijgt men:

```
LD      I0.0
LD      I0.2
LD      I0.1
LD      I0.3
O       Q0.0
ALD
OLD
ALD
=       Q0.0
```

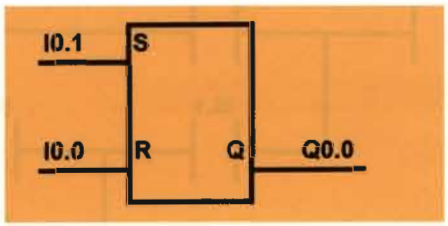
Hetzelfde schema In FBD voorgesteld:



Enkel voor klasgebruik.

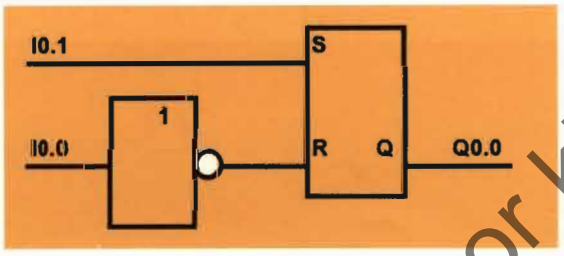
9.4 Set/Reset-functie

Een set/reset-functie wordt ook SR-flipflop genoemd.
De signaalwisseling van toestand "0" naar toestand "1" op de set-ingang zorgt ervoor dat de uitgang Q toestand "1" krijgt en deze toestand behoudt. Wanneer de set-ingang weer toestand "0" krijgt, blijft de toestand van de uitgang hoog. De uitgang Q kan alleen weer de toestand "0" aannemen wanneer een commando gegeven wordt aan de resetingang. Bij een toestandsverandering aan de reset-ingang van toestand "0" naar toestand "1" zal de uitgang gedeactiveerd worden. In onderstaand voorbeeld wordt de ingang I0.0 verbonden met de reset-ingang. De sensor, verbonden aan de ingang I0.1 van de PLC, wordt verknoopt met de set-ingang. De uitgang van de flipflop wordt verknoopt met de uitgang Q0.0 van de PLC.

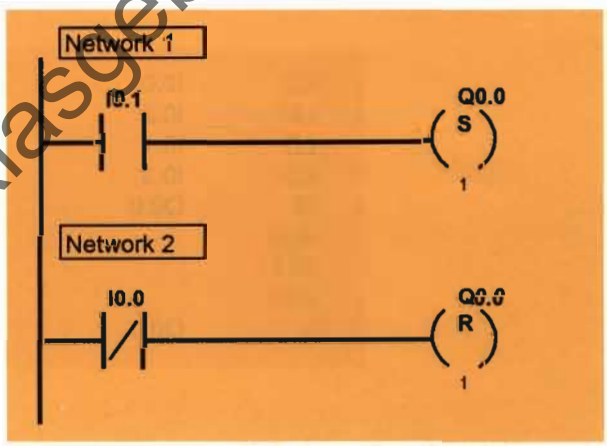


In de vorige paragraaf werd verklaard waarom een detector met een onderbrekingsfunctie moet worden uitgevoerd met een normaal gesloten contact. Deze stopdrukknop zal aangesloten worden aan de ingang. In het programma wordt de reset-ingang verknoopt met de SR-flipflop. Het resetten van de flipflop zal moeten gebeuren op het ogenblik dat de toestand van deze ingang "0" wordt. Men zal de resetingang dus moeten inverteren. Men verkrijgt dan het volgende programma:

FBD



LAD



Onder het set- en resetsymbool staat het aantal uitgangen dat wordt geactiveerd of gedeactiveerd.
Als het cijfer 1 wordt vervangen door het cijfer 3, dan worden de uitgangen Q0.0, Q0.1 en Q0.2 geactiveerd, respectievelijk gedeactiveerd.
Hetzelfde programma voorgesteld in STL, geeft het volgende resultaat:

```
LD      I0.1
S       Q0.0,  1

LDN     I0.0
R       Q0.0,  1
```

Aan de flipflop kan een set- of een reset-prioriteit worden gegeven door de volgorde van programmeren te veranderen. De laatst geprogrammeerde instructie heeft prioriteit. Wanneer men bij de reset-prioriteit zowel de set- als reset-ingang activeert, zal in het PIQ-register eerst de uitgang op "1" geset worden. In het volgende netwerk wordt deze al terug gereset. Omdat slechts op het einde van de programmacyclus een kopie van het PIQ-register naar de uitgangsmodule wordt gemaakt, zal de periferie-uitgang altijd laag of gereset blijven wanneer men beide detectoren activeert.
Bij de hoger vermelde voorbeelden heeft de reset prioriteit.

Het geven van meerdere setimpulsen tijdens het actief zijn van de uitgang, of het sturen van meerdere reset-impulsen als de toestand van de uitgang "0" is, resulteert niet in een verandering van het uitgangsniveau.

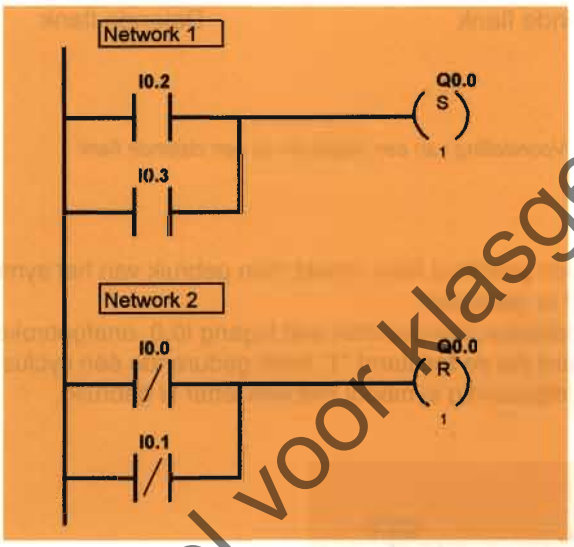
Enkel voor klasgebruik.

9.5 Set/Reset-functie gestuurd vanuit meerdere plaatsen

Om een proces te starten en te stoppen vanuit meerdere plaatsen, kan men ook gebruik maken van de set/reset-functie. Men heeft dan het voordeel dat men de geheugenfunctie (overneemcontact) niet zelf moet programmeren.

Bij het opstellen van een PLC-programma, gebaseerd op een relaisschema, bepaalt men de voorwaarden die noodzakelijk zijn om de uitgang te activeren. Bij het gebruik van een set/reset-instructie zal men bij het programmeren van de resetvoorwaarden bepalen wanneer de uitgang wordt gedeactiveerd.
Stelt men het programma op, zoals bij de klassieke relaisschakelingen, dan plaatst men de stopvoorwaarden in serie. Wordt er gebruik gemaakt van een set/reset-functie dan gaat men ervan uit dat de uitgang wordt gereset als men op de ene OF de andere stopdrukknop drukt. Men verkrijgt dus een OF-functie. Bovendien moet er rekening mee gehouden worden dat de stopdrukknop wordt uitgevoerd met een normaal gesloten contact. De PLC leest in de ruststand een logisch "1"-signaal. De resetingangen moeten worden geïnverteerd.

In het volgende voorbeeld maakt men gebruik van dezelfde aansluitingen als vermeld in punt 9.3.
De detectoren aan de ingangen I0.0 en I0.1 hebben een stopfunctie. De sensoren, aangesloten aan de ingangen I0.2 en I0.3 bezitten een startfunctie.
Heeft de stop voorrang dan verkrijgen we het volgende programma:

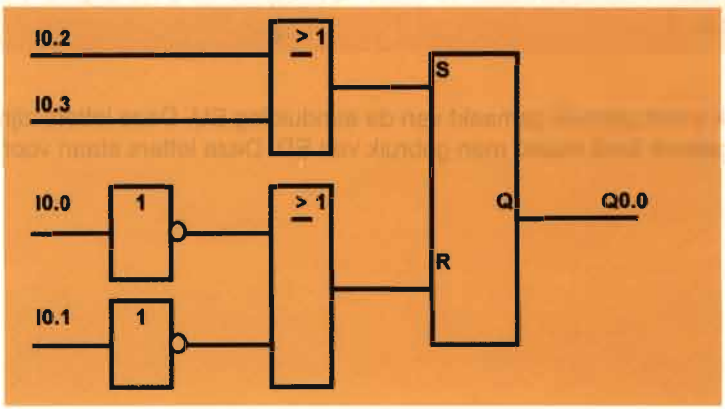


Als men hetzelfde programma voorstelt in STL, krijgt men:

```
LD      I0.2
O       I0.3
S       Q0.0,  1

LDN     I0.0
ON      I0.1
R       Q0.0,  1
```

Maakt men gebruik van FBD dan krijgt men:



Enkel voor klasgebruik.

10 Flankdetectie

Naast het detecteren van een statische toestand "0" of "1" is het vaak nodig op een signaalverandering te reageren. Niet het absolute niveau van het signaal is belangrijk, maar het moment waarop zich een signaalverandering voordoet. In een dergelijk geval wordt de term flankgestuurd gebruikt, in tegenstelling met de aanduiding niveausturing.
Er wordt een onderscheid gemaakt tussen een stijgende en een dalende flank. De stijgende flank treedt op als de toestand van "0" naar "1" evolueert. Als er een toestandswijziging optreedt van "1" naar "0" dan spreekt men van een dalende flank.

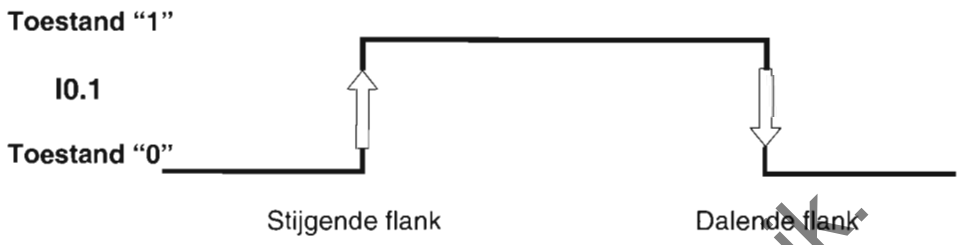
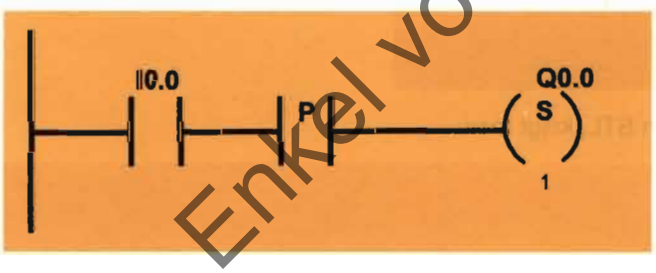


Fig. 3.7 Voorstelling van een stijgende en een dalende flank

Enkel voor klasgebruik

Om een signaal om te vormen tot een positieve flank maakt men gebruik van het symbool van een voorwaarde, waartussen de letter P is geplaatst.
Ook al blijft de toestand van de schakelaar, aangesloten aan ingang I0.0, onafgebroken "1", toch zal er slechts een puls worden gegenereerd die de toestand "1" heeft gedurende één cyclus.
Bij een negatieve flank wordt een gelijksoortig symbool met een letter N gebruikt.



Hetzelfde programma, voorgesteld in STL, geeft:

```
LD      I0.0
EU
S       Q0.0,  1
```

Bij een positieve flank wordt gebruik gemaakt van de aanduiding EU. Deze letters zijn de afkorting van Edge Up. Bij een negatieve flank maakt men gebruik van ED. Deze letters staan voor Edge Down.

10.1 Positieve flank

Sommige PLC's beschikken over geen standaardinstructie om een flank te detecteren. De gebruiker kan op een eenvoudige manier zelf een programma schrijven voor flankdetectie.

Er wordt gebruik gemaakt van het feit dat een PLC het programma cyclus per cyclus verwerkt.
De volgorde van de programmadelen is in dit geval essentieel.

Als men het programma opstelt in STL dan verkrijgt men:

```
Network 1:
    LD      I0.0
    A       M10.0
    =       M10.1
Network 2:
    LDN     I0.0
    =       M10.0
```

Het signaal-tijddiagram, voorgesteld in fig. 3.8, verduidelijkt de werking van de positieve flankdetectie. De verticale as stelt de verschillende signalen voor; op de horizontale worden na elkaar de opeenvolgende cycli weergegeven. Merker M10.1 wordt gedurende één cyclus actief en dit op het ogenblik van een stijgende flank van ingang I0.0.

In het signaal-tijddiagram wordt het begin van een nieuwe cyclus weergegeven door een verticale lijn. De toestandsveranderingen worden weergegeven juist voor of juist na de verticale lijn. Op deze manier wordt weergegeven dat er een zeer kleine vertraging optreedt door de cyclische verwerking van het PLC-programma.
Tijdens één cyclus worden:

- de toestanden van de ingangen gelezen en bewaard in het PII-geheugen;
- het programma berekend en de resultaten bewaard in het PIQ-geheugen;
- een kopij van het PIQ-geheugen overgedragen naar de uitgangen.

Enkel voor klasgebruik.

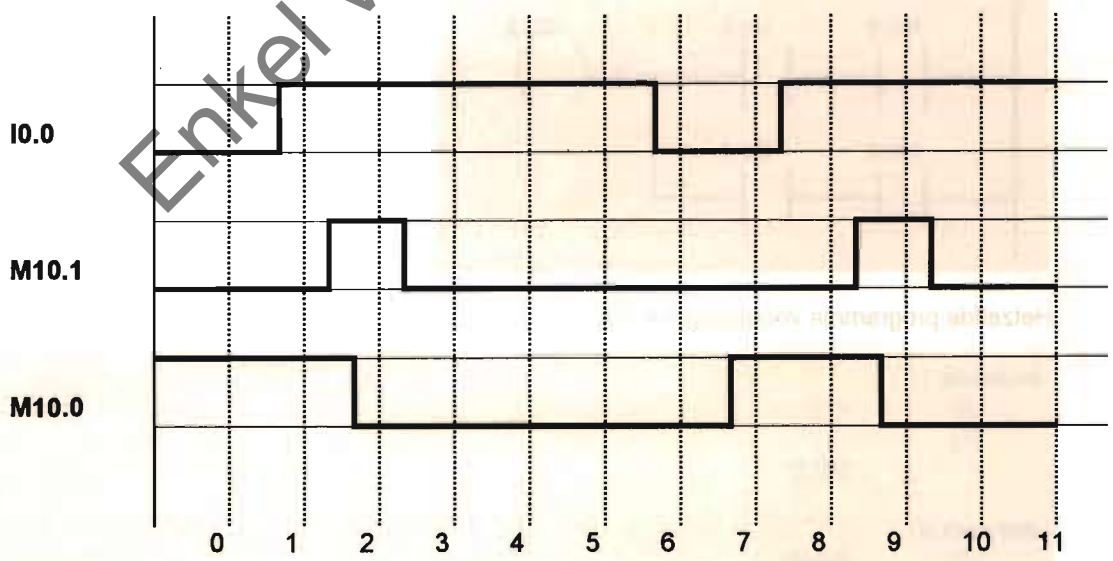


Fig. 3.8 Signaal-tijddiagram: M 10.1 reageert op stijgende flank.

Merker M10.1 wordt hoog gedurende één cyclus. Dit gebeurt op het ogenblik dat een stijgende flank is waargenomen aan ingang I0.0.

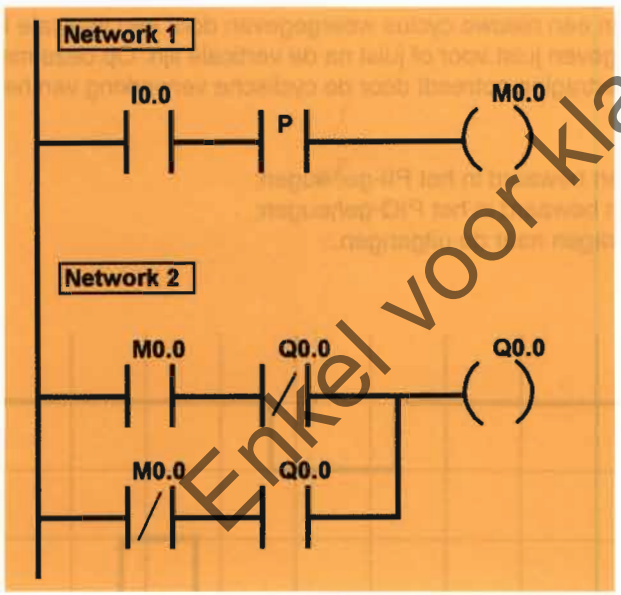
10.2 Dynamische ingangen

In een vorige paragraaf werd aangetoond dat men voor een onderbrekingsfunctie een detector met een normaal gesloten contact moet gebruiken. Om gelijksoortige redenen zal men voor sommige toepassingen moeten overwegen om de sturing te starten door gebruik te maken van een flanksturing. Bij een defect schakelcontact van de detector kan een permanent "1" signaal doorgestuurd worden naar de PLC.
De sturing zal iedere keer opnieuw herstarten of eventueel geactiveerd blijven. Als men de sturing laat reageren op de stijgende flank, dan belet men dat een defecte detector het proces ongewenst beïnvloedt. In de regeltechniek spreekt men dan van een dynamische ingang.

10.3 Teleruptorschakeling

Bij de volgende schakeling maakt men gebruik van een dynamische ingang. Deze schakeling wordt gebruikt als impulsschakeling. In woningen wordt een soortgelijke schakeling gebruikt om de verlichting in en uit te schakelen door gebruik te maken van parallel geschakelde drukknoppen. Men gebruikt eveneens de benamingen: teleruptor of afstandsschakelaar.
Deze schakeling kan men eveneens classificeren in de groep van geheugenschakelingen. Het is een variante van een SR-flipflop, maar men gebruikt hier maar één pulsgever.

Als de uitgang de toestand "0" bezit op het ogenblik dat er een puls gegeven wordt, dan zal de uitgang de toestand "1" aannemen. Zelfs als er aan de ingang geen toestand "1" aanwezig is, blijft de uitgang toch geactiveerd. Als de toestand van de uitgang "1" is en er volgt op dezelfde ingang een nieuwe puls, dan wordt de uitgang terug "0". Het volgende programma voldoet aan de gestelde eisen:



Enkel voor klasgebruik.

Hetzelfde programma voorgesteld in STL:

```
Network 1:
    LD      I0.0
    EU
    =       M0.0

Network 2:
    LD      M0.0
    AN      Q0.0
    LDN     M0.0
    A       Q0.0
    OLD
    =       Q0.0
Network 3:
    MEND
```

11 Samenvatting

1 Een logische besturingsfunctie wordt identiteit genoemd wanneer het uitgangssignaal dezelfde toestand als het ingangssignaal aanneemt.

2 Een logische functie wordt een inversie of negatie genoemd wanneer het uitgangssignaal steeds de tegenovergestelde logische waarde ten opzichte van het ingangssignaal aanneemt.

3 Een besturingsfunctie wordt een EN-functie genoemd als het uitgangssignaal de toestand "1" aanneemt terwijl alle ingangssignalen de toestand "1" bezitten.

4 Men spreekt van een OF-functie als het uitgangssignaal de toestand "1" aanneemt terwijl minstens één van de ingangssignalen toestand "1" bezit.

5 De NIET-functie heeft prioriteit op de EN-functie, de EN-functie heeft voorrang op de OF-functie.

6 Het ladderdiagram visualiseert het PLC-programma. Het is GEEN weergave van de fysische toestand van de detectoren.

7 Merkers zijn gedefinieerde geheugenplaatsen die de toestand "0" of "1" kunnen aannemen. Ze worden op een gelijkaardige manier als de uitgangen geprogrammeerd.

8 Voor alle onderbrekingsvoorwaarden moeten er detectoren met normaal gesloten contacten aan de ingangen van de PLC worden aangesloten. Men mag uiteraard deze functie niet meer inverteren in het programma.

9 Met de set/reset-instructie kan men geheugenelementen op een eenvoudige manier programmeren. De volgorde van de set- en resetvoorwaarde is bepalend voor de prioriteit.

10 Als de sturing op de toestand zelf reageert, is de sturing niveaugevoelig. Als een sturing op de verandering van de toestand reageert, spreken we van flankdetectie.

Enkel voor klasgebruik.

12 Opdrachten

12.1 Vragen

1 Geef de verschillende elementen die men moet definiëren om een merker te gebruiken in zijn PLC-programma.

2 Wanneer gebruikt men de LOAD-instructie?

3 Verklaar het begrip "Stack". Geef een voorbeeld en verklaar wanneer men de stack gebruikt in een PLC-programma.

4 Waarom moet men een programma opdelen in verschillende netwerken?

5 Bespreek het verschil tussen een OF- en een EXOF-functie.

6 Bespreek en verantwoord waarom men een detector met een normaal gesloten contact moet gebruiken bij een onderbrekingsfunctie. Wat zijn de gevolgen voor het PLC-programma?

7 Als men gebruik maakt van een set/reset-functie, hoe zorgt men er dan voor dat bij het opstellen van zijn PLC-programma de reset-functie prioriteit heeft?

8 Verklaar het verschil tussen flankgevoelig en niveaugevoelig. Geef een voorbeeld en verduidelijk je antwoord met behulp van een signaal-tijddiagram.

12.2 Opgaven

1 Voer voor de onderstaande schema's de volgende opdrachten uit:
- Stel het programma op in de vorm van een ladderdiagram.
- Stel het programma op in de vorm van een instructielijst (STL).
- Teken het logische schema (FBD).

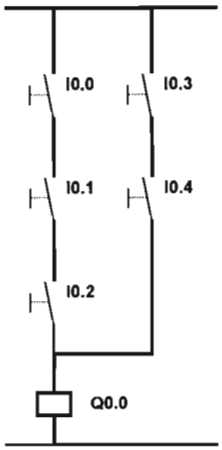


Fig. 3.10

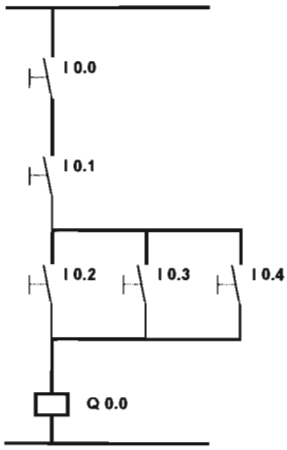


Fig.3. 11

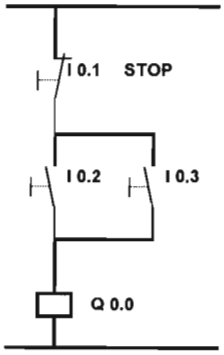


Fig. 3.12

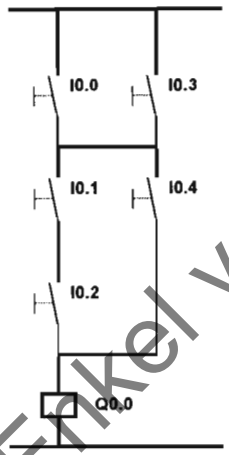


Fig. 3.13

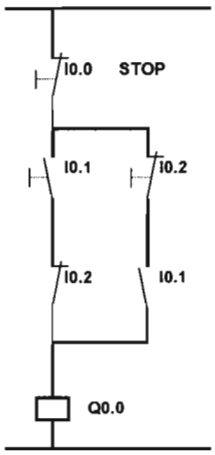


Fig.3. 14

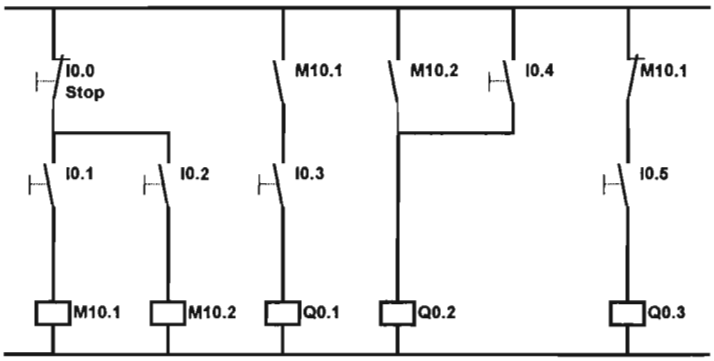


Fig. 3.15

Enkel voor klasgebruik.

2 Ontwerp een PLC-programma dat aan de volgende eisen voldoet.
In een reactorvat zijn 3 signaalgevers gemonteerd.
Als het niveau in het vat te hoog is, reageert de detector aangesloten aan ingang I0.1. Een thermostaat reageert als de temperatuur boven een bepaalde waarde is gestegen. De thermostaat is aangesloten aan ingang I0.2. Een drukschakelaar, aangesloten aan ingang I0.3, is zodanig afgesteld dat hij reageert als de druk boven de ingestelde waarde stijgt.
Zodra één van de genoemde signaalgevers wordt bediend, moet een oranje lampje, aangesloten aan uitgang Q0.0, branden. Zodra echter 2 van de signaalgevers bediend zijn, dan moet de rode lamp, aangesloten aan de uitgang Q0.1, branden. Zijn alle drie de signaalgevers bediend, dan moeten de beide lampen branden, terwijl ook een bel rinkelt. Deze bel is aangesloten aan de uitgang Q0.2.
Deze bel blijft rinkelen totdat de stopdrukknop, aangesloten aan ingang Q0.0, wordt geactiveerd.
Vatten we een lijst van de aansluitingen samen, dan krijgen we:

I0.0	stopdrukknop, normaal gesloten contact
I0.1	niveau
I0.2	thermostaat
I0.3	druk
Q0.0	oranje lamp
Q0.1	rode lamp
Q0.2	bel

Gebruik, in samenspraak met de lesgever, één van de drie programmeertalen.

3 Maak een PLC-programma dat aan de volgende eisen voldoet.
Men wil de signalen A en B met elkaar vergelijken. Als A=1 en B=0 moet de lamp X branden, lamp Y moet branden als B=1 en A=0. De lampen moeten allebei branden als de signalen een gelijke waarde hebben.
De volgende aansluitingen worden gebruikt:

I0.0	signaal A
I0.1	signaal B
Q0.0	lamp X
Q0.1	lamp Y

Er kan voor om het even welke programmeertaal gekozen worden.

Enkel voor klasgebruik.

4 De afsluiters van vat A, vat B en vat C zijn voorzien van de signaalgevers A-open, B-open en C-open. Deze signaalgevers worden bediend als de desbetreffende afsluiters open staan. De motor mag maar draaien als de startschakelaar bediend is en tenminste twee van de afsluiters open staan. Zodra de installatie gestart is, blijft de motor draaien tot men op de stopdrukknop drukt.
Men gebruikt de volgende aansluitingen:

I0.0	stopdrukknop
I0.1	startdrukknop
I0.2	vat A open
I0.3	vat B open
I0.4	vat C open
Q0.0	motor pomp

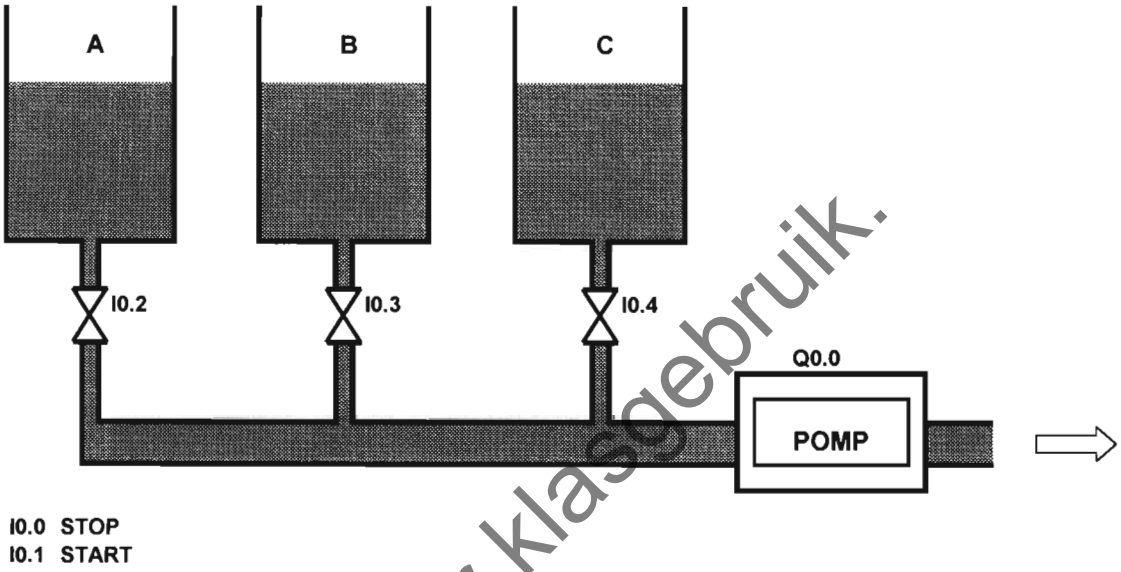


Fig. 3.16 Installatie met 3 vaten

Enkel voor klasgebruik.

5 In een reactorvat zijn 3 drukschakelaars aangebracht, die zo zijn ingesteld dat de schakelaar I0.0 wordt bediend bij een druk van 1 bar, schakelaar I0.2 bij 2 bar en schakelaar I0.3 bij 3 bar.
Op een paneel zijn 3 lampen gemonteerd. De lamp, aangesloten aan uitgang Q0.1, moet branden als de druk in het vat hoger is dan 1 bar, maar lager dan 2 bar. De lamp, aangesloten aan de uitgang Q0.2 moet branden als de druk groter is dan 2 bar maar kleiner dan 3 bar. De lamp, aangesloten aan de uitgang Q0.3, moet branden als de druk groter is dan 3 bar.
Als een schakelaar defect is of de bedrading onklaar is geraakt, moeten de drie lampen samen branden.