Cat. No. W353-NL2-02

SYSMAC C-series
Programmeerbare besturingen

# CPM1/CPM1A

# PROGRAMMEERHANDLEIDING

OMRON

# *Mededeling*

OMRON apparatuur wordt gefabriceerd voor gebruik volgens de juiste procedures door een gekwalificeerde gebruiker en alleen voor de doeleinden die in deze handleiding worden beschreven.

De volgende conventies worden gebruikt om voorzorgsmaatregelen te tonen en te classificeren. Schenk altijd aandacht aan de informatie die getoond wordt. Het geen aandacht schenken aan of negeren van waarschuwingen kan leiden tot het gewond raken van mensen of schade aan het product.

| | |
|---|---|
| **Gevaar** | Geeft informatie aan die, wanneer er geen acht op wordt geslagen, zeer waarschijnlijk zal leiden tot ernstige verwonding of verlies van leven. |

| | |
|---|---|
| **Waarschuwing** | Geeft informatie aan die, wanneer er geen acht op wordt geslagen, mogelijk kan leiden tot ernstige verwonding of verlies van leven en vrijwel zeker tot schade aan het product. |

| | |
|---|---|
| **Voorzichtig** | Geeft informatie aan die, wanneer er geen acht op wordt geslagen, mogelijk kan leiden tot relatief ernstige verwonding of letsel, schade aan het product of verkeerde werking van het product. |

# OMRON *product verwijzingen*

Namen van OMRON producten beginnen met een hoofdletter in deze handleiding.

Het woord unit wordt gebruikt om een OMRON product aan te duiden, onafhankelijk van het feit of het woord unit in de naam van het product voorkomt.

Het woord CPM1(A) wordt gebruikt wanneer de betreffende tekst op zowel de CPM1 als de CPM1A van toepassing is, ook al wordt maar één van beide PLC typen in een bijpassende figuur afgebeeld. Wordt in de tekst CPM1 of CPM1A gebruikt dan is de besproken optie alleen op dat type van toepassing.

Gebruikte afkortingen en termen zijn verklaard in de appendix.

# *Visuele hulpmiddelen*

De volgende koppen verschijnen in de linkerkolom van de handleiding om u verschillende soorten informatie snel te laten vinden.

**Opmerking** Geeft informatie weer die in het bijzonder praktisch is voor efficiënt en handig gebruik van het product.

**1, 2, 3...** 1. Geeft diverse soorten lijsten weer zoals procedures, controlelijsten etc.

**Noot** Geeft een noot weer. Wordt gebruikt in combinatie met tabellen.

# OMRON *manual referenties*

Voor het gebruik van de CPM1(A) kunt U de volgende handleidingen raadplegen.

| | |
|---|---|
| Nederlandstalig | CPM1(A) installatie handleiding<br>CPM1(A) Programmeerhandleiding<br>SYSWIN Handleiding |
| Engelstalig | W228 CQM1/CPM1/CPM1A/SRM1 Programming manual<br>W262 CPM1 Operation manual<br>W317 CPM1A Operation manual |

Naast de diverse handleidingen die voor de CPM1(A) beschikbaar zijn kunt u SYSTOOLS gebruiken voor het maken van instellingen in de PC Setup van de CPM1(A). Naast programma's voor het maken van instellingen in PLC's en speciale kaarten bevat SYSTOOLS ook SYSHELP. Dit is een help bestand waarin diverse wetenswaardigheden over OMRON PLC's zijn gebundeld.

OMRON CPM1/CPM1A Programmeerhandleiding Publicatie november 2000

Document referentie NLMAN-CPM1-programmeerhandleiding Revisie. 2

# *Inhoudsopgave*

# *Voor wat betreft deze handleiding*

De CPM1(A) is een compacte, snelle PLC die ontworpen is voor geavanceerde besturingen met een bereik tot 100 I/O. Deze beknopte handleiding beschrijft de werking van de CPM1(A). Instructies die in deze handleiding niet worden besproken verwijzen wij u naar de overige handleidingen van deze PLC.

| | |
|---|---|
| **Voorzichtig** | Lees deze handleiding nauwkeurig en wees er zeker van dat u de hierin weergegeven informatie goed begrijpt voor u begint met het programmeren van een OMRON PLC. |

**Sectie 1:** Schrijven en invoeren van het programma

Deze sectie verklaart de basisstappen en -concepten die bekend moeten zijn bij het schrijven van een eenvoudig ladderdiagram programma. Daarnaast wordt het invoeren van dat programma en het uitvoeren door de PLC ervan behandeld. De instructies die nodig zijn om de basisopzet van het ladderdiagram vast te leggen en die de executie beïnvloeden worden uitgelegd.

**Sectie 2:** CPM1(A) PC Setup

De PC Setup bevat diverse operating parameters die de werking van de CPM1(A) bepalen. Om maximaal gebruik te maken van de CPM1(A) functionaliteit wanneer interrupt processing en communicatie functies gebruikt worden kan de PC Setup "op maat" ingesteld worden, afhankelijk van de taak die uitgevoerd moet worden.

**Sectie 3:** Geheugengebieden

Verschillende typen data zijn nodig om een besturing effectief en correct te kunnen laten werken. Om met deze verschillende typen data overweg te kunnen is de PLC voorzien van een aantal geheugengebieden voor data opslag, waarbij elk gebied een andere functie heeft. Deze verschillende gebieden worden hier individueel besproken waarbij alle informatie die nodig is om het te kunnen gebruiken wordt gegeven.

**Sectie 4:** Instructieset

De OMRON SYSMAC CPM1(A) PLC beschikt over een uitgebreide instructieset die het mogelijk maakt dat gecompliceerde processen eenvoudig geprogrammeerd kunnen worden. Deze sectie beschrijft de instructies individueel en geeft het ladderdiagram symbool, de data gebieden die gebruikt kunnen worden en de vlaggen die door de instructie beïnvloed worden.

**Appendix:**

In de appendix is diverse informatie over de CPM1(A) opgenomen.

Aan deze handleiding is de grootst mogelijke zorg besteed. Mochten er ondanks deze zorg nog onjuistheden of onduidelijkheden vermeld zijn, dan stellen wij ons uitdrukkelijk niet aansprakelijk voor eventuele gevolgen. Voor suggesties ter verbetering houden wij ons aanbevolen.

# 1 Schrijven en invoeren van het programma

Deze sectie verklaart de basisstappen en -concepten die bekend moeten zijn bij het schrijven van een eenvoudig ladderdiagram programma. Daarnaast wordt het invoeren van dat programma en het uitvoeren door de PLC ervan behandeld. De instructies die nodig zijn om de basisopzet van het ladderdiagram vast te leggen en die de executie beïnvloeden worden uitgelegd. De complete instructieset die tijdens het programmeren gebruikt kan worden, is beschreven in het hoofdstuk over de instructieset.

## 1.1 Terminologie

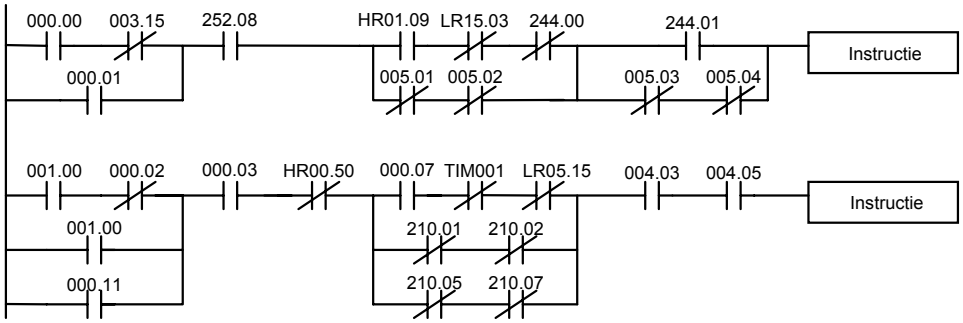
Er worden in beginsel twee typen instructies gebruikt bij ladderdiagram programmeren. Instructies die verbanden leggen tussen de voorwaarden (condities) in het ladderdiagram en instructies die aan het einde van de executieconditie geplaatst worden. De eerste zijn in instructievorm alleen zichtbaar zijn wanneer een programma wordt omgezet naar mnemonics.

De meeste instructies hebben minimaal 1 of meer operands. Operands wijzen of geven de data aan waarop de instructie uitgevoerd moet worden. Deze data wordt soms ingevoerd als constante numerieke waarde, maar is meestal het adres van het woord of bit die de te gebruiken data bevat. Bijvoorbeeld, een MOVE instructie die 000 als bron (source) operand heeft, verplaatst de inhoud van woord 000 naar een andere locatie. Deze andere locatie wordt ook als operand ingevoerd. Een bit waarvan het adres is gebruikt als operand wordt een operandbit genoemd; een woord waarvan het adres is gebruikt als operand wordt een operandwoord genoemd. Als de waarde wordt ingevoerd als constante dan wordt het voorafgegaan door # om aan te geven dat het geen adres is.

Andere termen die gebruikt worden voor het omschrijven van instructies worden geïntroduceerd in het hoofdstuk "Instructieset" op pagina 71.

## 1.2 Basis ladderdiagrammen

Een ladderdiagram bestaat uit één verticale lijn aan de linkerzijde waaruit één of meer horizontale lijnen naar rechts met vertakkingen ontstaan. De verticale lijn aan de linkerzijde heet de “bus-bar”; de horizontale lijnen heten instructieregels of rungs. Een onderling verbonden groep van instructieregels wordt een netwerk genoemd. Op de instructieregels zijn condities geplaatst die de instructies aan de rechterzijde van het diagram aansturen. De logische combinaties van deze condities bepalen wanneer de instructies aan de rechterzijde worden uitgevoerd. Hieronder wordt een ladderdiagram dat uit twee netwerken bestaat getoond.




Zoals getoond in het bovenstaande diagram kunnen instructieregels aftakken en weer bij elkaar komen. De verticale paren lijnen worden condities of contacten genoemd. Condities zonder diagonale lijn er doorheen worden normaal open condities genoemd en corresponderen met een LD (load) AND, of OR instructie. De condities met een diagonale lijn erdoor worden normaal gesloten condities genoemd en corresponderen met een LD NOT, AND NOT, of OR NOT instructie. Het nummer boven elke conditie geeft het operandbit voor deze instructie aan. Een naam onder de conditie geeft het label van het operandbit aan. De status van het

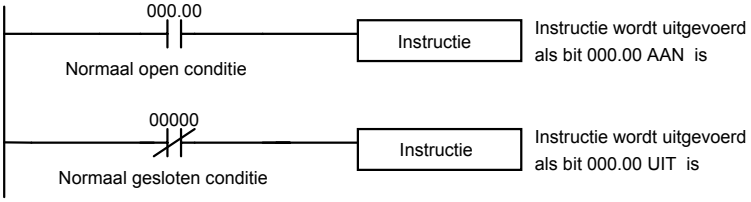
met de conditie geassocieerde bit bepaalt de executieconditie voor de er op volgende instructies of condities.

De manier waarop de uitvoering van de instructies leiden tot een bepaalde executieconditie wordt hieronder beschreven. Voor we ons hier echter mee bezig gaan houden zullen eerst een aantal basisbegrippen uitgelegd worden.

### *1.2.1 Basis begrippen*

**Normaal open / normaal gesloten condities**

Elke conditie in een ladderdiagram is aan of uit, afhankelijk van de status van het operandbit dat eraan is toegewezen. Een normaal open conditie is aan als het operandbit aan is en uit als het operandbit uit is. Een normaal gesloten conditie is aan als het operandbit uit is en uit als het operandbit aan is. Normaal gesproken gebruikt u een normaal open conditie wanneer u wilt dat iets gebeurd als een bit aan is en een normaal gesloten conditie wanneer u iets wilt laten gebeuren als een bit uit is.




**Executiecondities**

Bij ladderdiagramprogrammering bepaalt de logische combinatie van condities voor een instructie de voorwaarde waardoor de instructie wordt uitgevoerd. Deze voorwaarde, die aan of uit kan zijn, wordt de executieconditie voor de instructie genoemd. In principe hebben alle instructies anders dan de LD instructies executiecondities.

**Operandbits**

Het operand dat aan een ladder instructie wordt toegewezen kan elk bit in de IR, SR, HR, AR, LR, of TC gebieden zijn. Dit betekent dat de status van condities in een ladderdiagram kan worden bepaald door I/O bits, vlaggen, werkbits, timers/counters, etc. LOAD (LD) en OUTPUT (OUT) instructies kunnen ook bits uit het TR gebied gebruiken, maar dit kan alleen in een aantal speciale gevallen. Zie hiervoor "Vertakkende instructie regels" op pagina 19 voor details.

**Logische blokken**

De manier waarop condities instructies aansturen wordt bepaald door de relatie tussen de condities in de instructieregels die voor deze instructies zijn geplaatst. Elke reeks condities die samen een logisch resultaat creëren wordt een logisch blok genoemd. Deze term moet niet verward worden met de blokken die in SYSWIN gebruikt kunnen worden om een programma te structureren. Alhoewel netwerken geschreven kunnen worden zonder de individuele logische blokken waaruit ze zijn opgebouwd te begrijpen, is het analyseren van de logische blokken noodzakelijk om efficiënt te kunnen programmeren en is het zelfs essentieel wanneer programma's ingevoerd moeten worden in mnemonic code.

### *1.2.2 Mnemonic code*

Het ladderdiagram kan niet direct in de PLC ingevoerd worden. Het is noodzakelijk om het ladderdiagram om te zetten naar mnemonic code. Deze mnemonic code voorziet in exact dezelfde informatie als het ladderdiagram, maar dan in een vorm die direct op de PLC ingevoerd kan worden. Feitelijk kunt u een programma direct schrijven in mnemonic code, alhoewel het niet aan te raden is voor beginners of voor complexe programma's. Samengevat, onafhankelijk van het gebruikte programmeerapparaat wordt het programma in het PLC geheugen opgeslagen in mnemonic formaat. Dit maakt het belangrijk om deze mnemonic code te begrijpen.

Vanwege het belang van mnemonic code voor het compleet begrijpen van een programma, wordt de mnemonic code tegelijk met het ladderdiagram uitgelegd. Onthoud dat het niet noodzakelijk is om mnemonic code te gebruiken als u het programma met SYSWIN invoert. Alhoewel u mnemonic code, als u er de voorkeur aan geeft, wel kan gebruiken.

**Programmageheugen**

Het programma wordt ingevoerd op adressen in het programmageheugen (UM). Adressen in het programmageheugen verschillen iets van de adressen in andere geheugengebieden, omdat elk adres niet per se dezelfde hoeveelheid data hoeft te bevatten. Elk adres bevat één instructie met alle constanten, labels (definers) en

operands (dit wordt later in detail beschreven) die nodig zijn voor die instructie. Omdat sommige instructies geen operands nodig hebben, terwijl andere tot maximaal drie operands nodig hebben kunnen, programmageheugen adressen één tot vier woorden lang zijn.

Programmageheugen adressen starten op regel 00000 en lopen door tot de capaciteit van het programmageheugen uitgeput is. Het eerste woord van elke regel definieert de instructie. Alle operands van de instructie worden er achter geprogrammeerd, op dezelfde regel. De overige woorden die door een instructie benodigd zijn bevatten de operands die bepalen welke data gebruikt moet worden. Wanneer een programma wordt omgezet naar mnemonic code, worden instructies in dit formaat genoteerd, één instructie op een regel, zoals ze in het ladderdiagram getoond worden. Een voorbeeld van mnemonic code wordt hieronder getoond. De gebruikte instructies worden beschreven in hoofdstuk "Instructieset" op pagina 71.

| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | HR00.01 | |
| 00001 | AND | 000.01 | |
| 00002 | OR | 000.02 | |
| 00003 | LD NOT | 001.00 | |
| 00004 | AND | 001.01 | |
| 00005 | AND LD | 001.02 | |
| 00006 | MOV(21) | 000 | DM0000 |
| 00007 | CMP(20) | DM0000 | HR00 |
| 00008 | LD | 255.05 | |
| 00009 | OUT | 005.01 | |
| 00010 | MOV(21) | DM0000 | DM0500 |
| 00011 | DIFU(13) | 005.02 | |
| 00012 | AND | 000.05 | |
| 00013 | OUT | 005.03 | |

Tijdens het invoeren van mnemonic code in SYSWIN wordt de instructienaam gescheiden van de operands door een Tab of spaties. De operands onderling worden ook gescheiden door spaties of Tab.

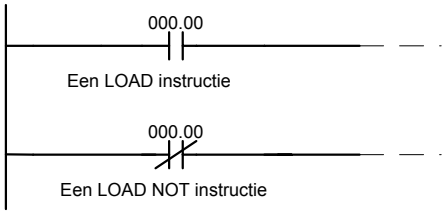
Tijdens het programmeren wordt het adres waarop de instructie wordt geplaatst automatisch bepaald. Boven in de editor laat SYSWIN het eerste programmageheugen adres van het netwerk zien. Wanneer een programma omgezet wordt naar mnemonic code, is het aan te raden om op programma geheugen adres 00000 te beginnen tenzij er een specifieke reden is om ergens anders te beginnen. SYSWIN begint altijd op adres 00000.

### *1.2.3 Ladder instructies*

De ladder instructies zijn die instructies die aangestuurd worden door de condities in het ladderdiagram. Ladder instructies onafhankelijk of in combinatie met de hieronder beschreven logische blok instructies, vormen de executiecondities waarop de uitvoering van alle andere instructies is gebaseerd.

**LOAD en LOAD NOT**

De eerste conditie waarmee elk logisch blok begint in een ladderdiagram is de LOAD of LOAD NOT instructie. Elk van deze instructies heeft één regel mnemonic code nodig. "Instructie" is gebruikt als een dummy instructie in de volgende voorbeelden en kan elke van de aan de rechterkant in het ladderdiagram geplaatste instructies zijn. Deze instructies worden ook wel aangeduid met “right-hand” of “uitvoerende” instructies.



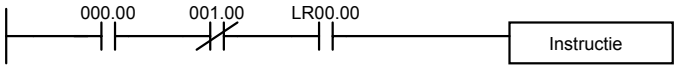

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | Instructie | |
| 00002 | LD NOT | 000.00 |
| 00003 | Instructie | |

Wanneer er maar één conditie in de instructieregel staat dan is de executieconditie voor de instructie aan de rechterkant aan wanneer deze conditie aan is. In het bovenstaande voorbeeld zal voor de LOAD instructie (dit is een normaal open

conditie) de executieconditie aan zijn wanneer 000.00 aan is. Voor de LOAD NOT instructie (dit is een normaal gesloten conditie) zal het aan zijn wanneer 000.00 uit is.

**AND en AND NOT**

Wanneer twee of meer condities in serie zijn geplaatst op dezelfde instructieregel, wordt de eerste met een LOAD of LOAD NOT instructie geprogrammeerd. De rest van de condities worden met AND of AND NOT instructies ingevoerd. Het volgende voorbeeld toont drie instructies die van af de linkerkant gezien een LOAD, een AND NOT en een AND instructie voorstellen. Elk van deze instructies heeft één regel mnemonic code nodig.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND NOT | 001.00 |
| 00002 | AND | LR00.00 |
| 00003 | Instructie | |

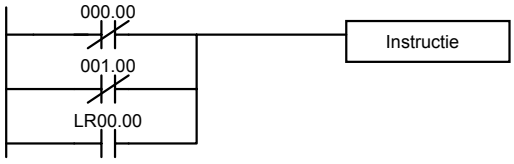
De instructie zal alleen een aan executieconditie hebben wanneer alle drie de condities aan zijn. Dit wil zeggen, wanneer 000.00 aan is, 001.00 uit is en LR00.00 aan is.

AND instructies in serie kunnen individueel beschouwd worden, waarbij elke AND de logische and uitvoert tussen de executieconditie (d.w.z., het totaal van alle condities tot aan dit punt) en de status van het operandbit van de AND instructie. Wanneer beide aan zijn, dan wordt een aan executieconditie aangemaakt voor de volgende instructie. Wanneer een van de twee of beide uit zijn dan is het resultaat ook uit. De executieconditie voor de eerste AND instructie in een serie is de status van de eerste conditie op de instructie regel.

Elke AND NOT instructie in een serie bepaalt de logische AND van de executieconditie en de inverse van het operandbit.

**OR en OR NOT**

Wanneer twee of meer condities op verschillende instructie regels liggen die parallel lopen en vervolgens samenkomen, dan wordt de eerste conditie met een LOAD of LOAD NOT instructie ingevoerd; de overige condities met OR of OR NOT instructies. Het volgende voorbeeld toont drie condities die ingevoerd moeten worden (in volgorde vanaf de bovenste) met een LOAD NOT, een OR NOT en een OR instructie. Wederom heeft elk van deze instructies één regel mnemonic code nodig.



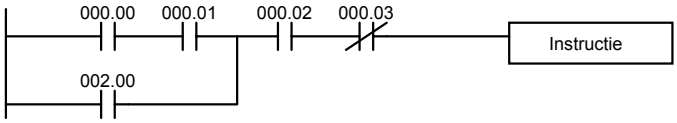

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD NOT | 000.00 |
| 00001 | OR NOT | 001.00 |
| 00002 | OR | LR00.00 |
| 00003 | Instructie | |

De instructie heeft een aan executieconditie wanneer één van de drie condities aan is, dat wil zeggen wanneer 000.00 uit is of 001.00 uit is of wanneer LR00.00 aan is.

OR en OR NOT instructies kunnen individueel beschouwd worden waarbij elke instructie de logische OR uitvoert tussen de executieconditie en de status van het bij de OR instructie horende operandbit. Als één van beide aan is dan wordt een aan executieconditie gegenereerd voor de volgende instructie.

**AND en OR Combineren**

Wanneer AND en OR instructies gecombineerd worden in meer gecompliceerde diagrammen, kunnen ze soms ook individueel beschouwd worden, waarbij elke instructie een logische bewerking uitvoert op de executieconditie en de status van het operandbit. Het volgende is een voorbeeld. Bestudeer dit voorbeeld tot u ervan overtuigd bent dat de mnemonic code dezelfde logica voorstelt als het ladderdiagram.



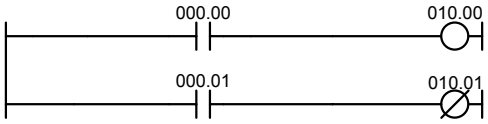

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND | 000.01 |
| 00002 | OR | 002.00 |
| 00003 | AND | 000.02 |
| 00004 | AND NOT | 000.03 |
| 00005 | Instructie | |

Hier wordt een AND berekend tussen de status van 000.00 en die van 000.01 om de executieconditie voor een OR met de status van 002.00 te bepalen. Het resultaat van deze bewerking bepaalt de executieconditie voor een AND met de status van 000.02, welke op zijn beurt de executieconditie bepaalt voor een AND met het inverse (AND NOT) van de status van 000.03.

In meer gecompliceerde diagrammen is het echter noodzakelijk om de juiste opbouw van de logische blokken te bedenken voor een executieconditie bepaald kan worden voor de laatste instructie. Dit is waar de AND LOAD en OR LOAD instructies worden gebruikt. Voor echter deze gecompliceerde diagrammen behandeld gaan worden, worden eerst de instructies beschreven die benodigd zijn om een eenvoudig "input-output" programma te kunnen maken.

**OUTPUT en OUTPUT NOT**

De eenvoudigste manier om de resultaten van gecombineerde executiecondities te bepalen is om het direct vast te leggen met de OUTPUT en OUTPUT NOT instructies. Deze instructies worden gebruikt om de status van het gebruikte operandbit aan te sturen, afhankelijk van de executieconditie. Met de OUTPUT instructie wordt het operandbit aan gezet zolang als de executieconditie aan is. Met de OUTPUT NOT instructie zal het operandbit aan gezet worden zolang de executieconditie uit is en uit gezet worden zolang de executieconditie aan is. Ze verschijnen in het ladderdiagram zoals hieronder getoond. In mnemonic code gebruikt elk van deze instructies één regel.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OUT | 010.00 |

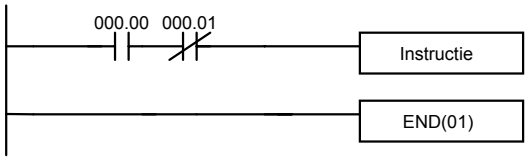
| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.01 |
| 00001 | OUT NOT | 010.01 |

In het bovenstaande voorbeeld zal 010.00 aan zijn zolang als 000.00 aan is en 010.01 zal uit zijn zolang als 000.01 uit is. In dit voorbeeld zijn 000.00 & 000.01 input bits en 010.00 & 010.01 output bits die zijn toegewezen aan de PLC. De signalen die binnen komen door de ingangen 000.00 en 000.01 sturen respectievelijk de output punten 010.00 en 010.01 aan.

De tijd dat een bit aan of uit is kan gemanipuleerd worden door de OUTPUT of OUTPUT NOT instructie te combineren met TIMER instructies. Raadpleeg "Timer - TIM" op pagina 87 voor details.

### *1.2.4 De END instructie*

De laatste instructie die benodigd is om een eenvoudig programma te kunnen completeren is de END instructie. Wanneer de CPU het programma verwerkt, worden alle instructies uitgevoerd tot de eerste END instructie. Hierna zal terug gegaan worden naar het begin van het programma en zal het opnieuw uitgevoerd worden. Alhoewel een END instructie op elk punt in het programma geplaatst kan worden, wat soms gedaan wordt voor debugging, zal geen enkele instructie na de eerste END instructie uitgevoerd worden tot deze END verwijderd wordt. Het nummer dat achter de END instructie staat in de mnemonic code is de functiecode. De functiecode kan gebruikt worden om een functie in te voeren, dit wordt later beschreven. De END instructie heeft geen operands nodig en wordt niet voorafgegaan door een executieconditie op de instructieregel.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND NOT | 000.01 |
| 00002 | Instructie | |
| 00003 | END(01) | |

Als er niet ergens in het programma een END instructie staat zal het programma niet uitgevoerd worden.

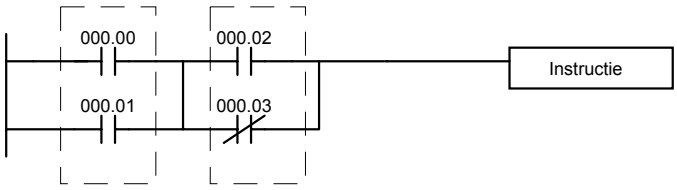
U kent nu alle instructies die nodig zijn om eenvoudige “input-output” programma’s te schrijven. Voor we stoppen met de basis ladderdiagrammen en ons bezig gaan houden met complexere instructies zullen we ons eerst verdiepen in de logische blok instructies (AND LOAD en OR LOAD), die soms ook in eenvoudige programma’s noodzakelijk zijn.

### *1.2.5 Logische blok instructies*

Logisch blok instructies zijn niet verbonden met specifieke condities in het ladderdiagram. Ze beschrijven de relatie tussen logische blokken in een netwerk. De AND LOAD instructie voert een logische AND uit op de executiecondities die twee logische blokken produceren. De OR LOAD instructie voert een logische OR uit tussen de executiecondities die twee logische blokken produceren.

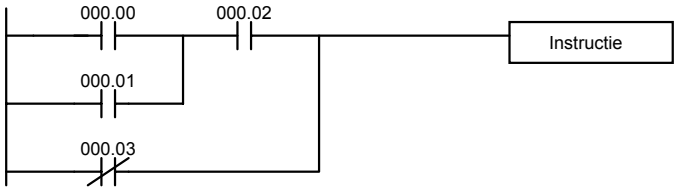
**AND LOAD**

Alhoewel het onderstaande netwerk eenvoudig lijkt, is er een AND LOAD instructie noodzakelijk om het te programmeren.




De twee logische blokken worden aangegeven met de gestreepte lijnen. Bestudering van het voorbeeld toont dat een aan executieconditie wordt geproduceerd wanneer: één van beide condities in het linker logisch blok aan is (dat is, wanneer 000.00 of 000.01 aan is) en wanneer één van beide condities in het rechter logisch blok aan is (dat is, wanneer 000.02 aan is of 000.03 uit is).

Het bovenstaande ladderdiagram kan echter niet omgezet worden naar mnemonic code met alleen AND en OR instructies. Wanneer een AND tussen 000.02 en het resultaat van de OR tussen 000.00 en 000.01 wordt uitgevoerd, raakt de OR NOT tussen 000.02 en 000.03 verloren en zal uitgevoerd worden als een OR NOT tussen alleen 000.03 en het resultaat van de AND tussen 000.02 en de eerste OR. Dit is uitgebeeld in de onderstaande figuur. Een OR functie wordt altijd uitgevoerd met een contact aan de busbar.




Wat hier nodig is, is een manier om beide OR functies onafhankelijk uit te voeren en de resultaten naderhand te combineren. Om dit te realiseren kunnen we de LOAD of LOAD NOT instructie in het midden van een instructieregel toepassen. Wanneer LOAD of LOAD NOT op deze manier wordt uitgevoerd, dan wordt de huidige executieconditie opgeslagen in een speciale buffer en het logische proces opnieuw gestart. Voor het combineren van het resultaat van de huidige executieconditie met dat van een vorige “ongebruikte” executieconditie, kan een AND LOAD of een OR LOAD instructie gebruikt worden. In dit geval refereert "LOAD" naar het laden van de laatste ongebruikte executieconditie. Een

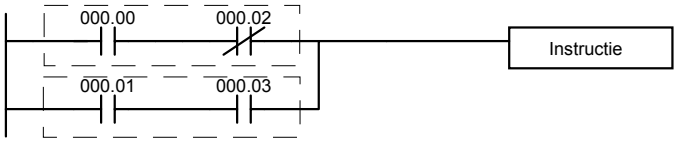
ongebruikte executieconditie wordt aangemaakt voor elke LOAD of LOAD NOT instructie, behalve de eerste, in een netwerk.

Wordt het eerste ladderdiagram geanalyseerd dan is de instructie voor 000.00 een LOAD en voor de conditie eronder een OR instructie tussen de status van 000.00 en die van 000.01. De instructie voor 000.02 is een volgende LOAD en voor de conditie eronder is het een OR NOT instructie, dat is een OR tussen de status van 000.02 en de inverse van de status van 000.03. Om vervolgens de executieconditie voor de instructie aan de rechterkant te bepalen moet de logische AND van de executiecondities geproduceerd door deze twee blokken berekend worden. De AND LOAD instructie doet dit. De mnemonic code van het ladderdiagram is beneden getoond. De AND LOAD instructie heeft geen operands nodig, omdat het berekeningen uitvoert met tevoren bepaalde executiecondities.

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OR | 000.01 |
| 00002 | LD | 000.02 |
| 00003 | OR NOT | 000.03 |
| 00004 | AND LD | |

**OR LOAD**

Het volgende diagram gebruikt een OR LOAD instructie tussen het logische blok boven en het logische blok beneden. Een aan executieconditie zal gegenereerd worden voor de instructie aan de rechterkant wanneer of 000.00 aan is en 000.01 uit is, of wanneer 000.02 en 000.03 beide aan zijn. De werking van de OR LOAD instructie en de mnemonic code ervan is identiek aan die van de AND LOAD instructie, behalve dat de logische OR wordt bepaald tussen de huidige executieconditie en de laatste ongebruikte executieconditie.



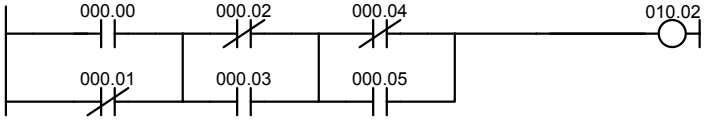

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 00000 |
| 00001 | AND NOT | 00001 |
| 00002 | LD | 00002 |
| 00003 | AND | 00003 |
| 00004 | OR LD | |

Natuurlijk gebruiken sommige netwerken zowel de AND LOAD als de OR LOAD instructies.

**Logische blok instructies in series**

Om een diagram te coderen waar meerdere logische blok instructies in voorkomen, moet het diagram verdeeld worden in logische blokken. Elk blok wordt begonnen door een LOAD instructie voor de eerste conditie te gebruiken en vervolgens wordt AND LOAD of OR LOAD gebruikt om de blokken logisch te combineren. Met zowel AND LOAD als OR LOAD zijn er twee manieren om dit te realiseren. Één manier is om de logische blok instructie na de eerste twee blokken in te voeren en vervolgens na elk volgend blok. De andere manier is om eerst alle blokken die samengevoegd moeten worden te coderen en daarna de logische blok instructies die ze samenvoegen. In dit geval zullen de laatste twee blokken eerst gecombineerd moeten worden en vervolgens elk voorgaande blok, op deze manier terug werkend naar het eerste blok. Alhoewel beide methoden exact hetzelfde resultaat produceren, kan de tweede methode, die waarbij alle logische blok instructies samen geprogrammeerd worden, alleen gebruikt worden wanneer acht of minder blokken gecombineerd moeten worden. Dat is wanneer zeven of minder logische blok instructies gebruikt moeten worden.

Het volgende netwerk gebruikt AND LOAD in mnemonic code omdat drie paar parallelle condities in serie liggen. Beide opties voor het coderen worden getoond.



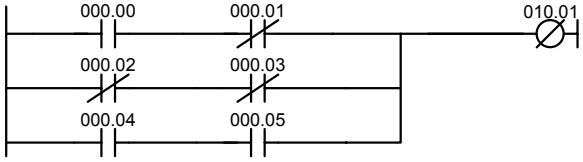

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OR NOT | 000.01 |
| 00002 | LD NOT | 000.02 |

| | | |
|---|---|---|
| 00003 | OR | 000.03 |
| 00004 | AND LD | |
| 00005 | LD NOT | 000.04 |
| 00006 | OR | 000.05 |
| 00007 | AND LD | |
| 00008 | OUT | 010.02 |

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OR NOT | 000.01 |
| 00002 | LD NOT | 000.02 |
| 00003 | OR | 000.03 |
| 00004 | LD NOT | 000.04 |
| 00005 | OR | 000.05 |
| 00006 | AND LD | |
| 00007 | AND LD | |
| 00008 | OUT | 010.02 |

Met de onderste methode kunnen maximaal acht blokken samengevoegd worden. Er is geen limiet aan het aantal blokken dat met de eerste methode gecombineerd kan worden.

Het volgende netwerk gebruikt OR LOAD instructies in mnemonic code aangezien drie paar AND condities parallel aan elkaar liggen.




De eerste conditie van elke reeks wordt begonnen met een LOAD instructie en vervolgens wordt een AND uitgevoerd met de volgende conditie. De eerste twee blokken kunnen eerst gecodeerd worden, gevolgd door een OR LOAD, daarna het laatste blok gevolgd door een volgende OR LOAD; of de drie blokken kunnen eerst gecodeerd worden, gevolgd door twee OR LOAD's. De mnemonic code van beide methoden is hieronder getoond.

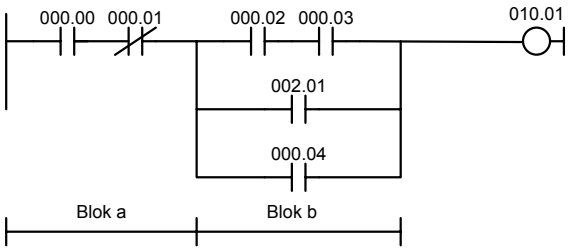
| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND NOT | 000.01 |
| 00002 | LD NOT | 000.02 |
| 00003 | AND NOT | 000.03 |
| 00004 | OR LD | |
| 00005 | LD | 000.04 |
| 00006 | AND | 000.05 |
| 00007 | OR LD | |
| 00008 | OUT NOT | 010.01 |

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND NOT | 000.01 |
| 00002 | LD NOT | 000.02 |
| 00003 | AND NOT | 000.03 |
| 00004 | LD | 000.04 |
| 00005 | AND | 000.05 |
| 00006 | OR LD | |
| 00007 | OR LD | |
| 00008 | OUT NOT | 010.01 |

**AND LOAD en OR LOAD combineren**

Beide codeer methoden die hierboven beschreven zijn kunnen ook gebruikt worden wanneer AND LOAD en OR LOAD gecombineerd gebruikt worden, zolang het aantal blokken niet boven de acht komt.

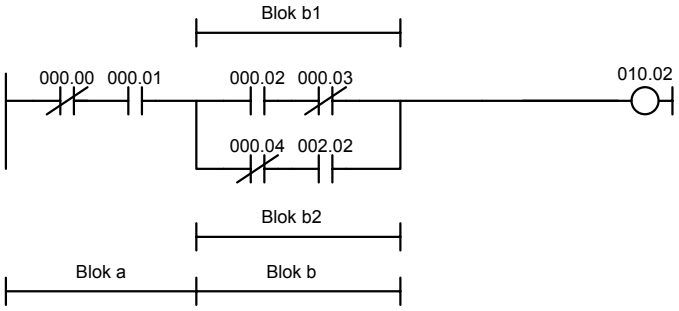
Het volgende diagram bevat twee logische blokken zoals getoond. Het is niet noodzakelijk om blok b te splitsen aangezien het direct met AND en OR gecodeerd kan worden.



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND NOT | 000.01 |
| 00002 | LD | 000.02 |
| 00003 | AND | 000.03 |
| 00004 | OR | 002.01 |
| 00005 | OR | 000.04 |
| 00006 | AND LD | |
| 00007 | OUT | 010.01 |

Alhoewel het volgende netwerk lijkt op het bovenstaande, kan blok b hieronder niet gecodeerd worden zonder het eerst in twee blokken te splitsen die met OR LOAD samen gevoegd worden. In dit voorbeeld worden de drie blokken eerst gecodeerd, vervolgens wordt een OR LOAD gebruikt om de laatste twee blokken te combineren, gevolgd door een AND LOAD om de executieconditie geproduceerd door de OR LOAD met de executieconditie van blok a te combineren.

Wanneer de logische blok instructies samen aan het einde van de logische blokken gecombineerd worden moeten ze zoals hieronder getoond in omgekeerde volgorde ingevoerd worden. D.w.z., de logische blok instructie voor de laatste twee blokken wordt eerst gecodeerd worden, gevolgd door de instructie die de executieconditie van de eerste logische blok instructie en de executieconditie van het logische blok aan het begin (derde vanaf het einde) van het programma combineert.



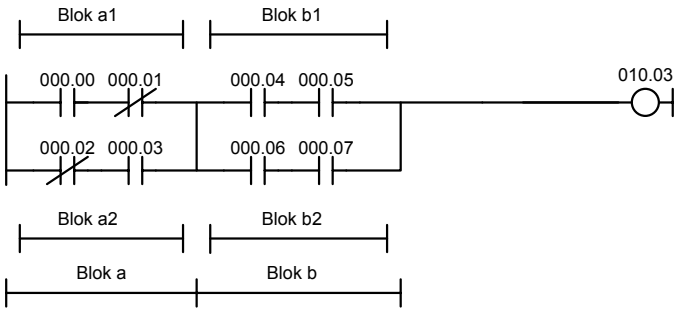

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD NOT | 000.00 |
| 00001 | AND | 000.01 |
| 00002 | LD | 000.02 |
| 00003 | AND NOT | 000.03 |
| 00004 | LD NOT | 000.04 |
| 00005 | AND | 002.02 |
| 00006 | OR LD | |
| 00007 | AND LD | |
| 00008 | OUT | 010.02 |

**Gecompliceerde diagrammen**

Wanneer bepaald moet worden welke logische blok instructies nodig zijn om een netwerk te kunnen coderen, is het soms noodzakelijk om het netwerk in grote blokken onder te verdelen en vervolgens deze grote blokken weer onder te verdelen in logische blokken die zonder logische blok instructies gecodeerd kunnen worden. Deze blokken worden vervolgens gecodeerd door eerst de kleine blokken samen te voegen en vervolgens de grotere. AND LOAD en OR LOAD worden gebruikt om blokken samen te voegen; deze instructies combineren altijd de laatste twee bestaande executiecondities, onafhankelijk of deze executiecondities zijn ontstaan uit een enkele conditie, uit logische blokken of van voorgaande logische blok instructies.

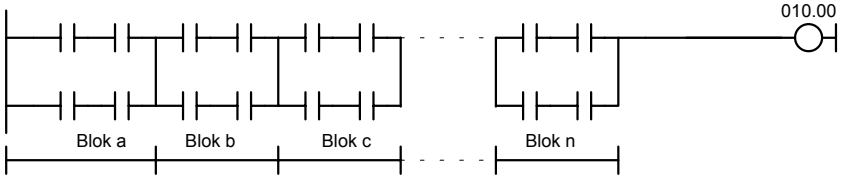
Wanneer er complexe netwerken gemaakt worden, worden blokken van af het begin van het netwerk (links boven) gecodeerd, waarbij indien mogelijk eerst naar beneden wordt gegaan en daarna naar rechts binnen het netwerk. In het algemeen houdt dit in dat wanneer er een keuze is de OR LOAD voor de AND LOAD gecodeerd zal worden.

Het volgende netwerk moet eerst verdeeld worden in twee blokken en elk van deze wordt vervolgens weer onderverdeeld in twee blokken voor het gecodeerd kan worden. Zoals hieronder getoond benodigen de blokken a en b een AND LOAD. Voor de AND LOAD gebruikt kan worden moet echter OR LOAD gebruikt worden om de blokken, boven en onder aan beide kanten, samen te voegen, om dus a1 en a2; b1 en b2 samen te voegen.

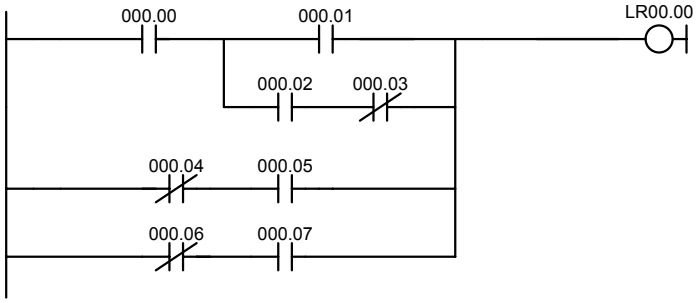



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND NOT | 000.01 |
| 00002 | LD NOT | 000.02 |
| 00003 | AND | 000.03 |
| 00004 | OR LD | |
| 00005 | LD | 000.04 |
| 00006 | AND | 000.05 |
| 00007 | LD | 000.06 |
| 00008 | AND | 000.07 |
| 00009 | OR LD | |
| 00010 | AND LD | |
| 00011 | OUT | 010.03 |

Het volgende type netwerk kan gemakkelijk gecodeerd worden als elk blok in volgorde wordt gecodeerd: Eerst van boven naar beneden, vervolgens van links naar rechts. In het volgende netwerk zullen de blokken a en b samen gevoegd worden door AND LOAD te gebruiken zoals hierboven getoond. Vervolgens zal blok c worden gecodeerd en een tweede AND LOAD zal gebruikt worden om het samen te voegen met de executieconditie van de eerste AND LOAD. Vervolgens wordt blok d gecodeerd en een derde AND LOAD gebruikt voor het samen voegen van de executieconditie van de tweede AND LD met de conditie van blok d, enzovoort tot en met blok n.



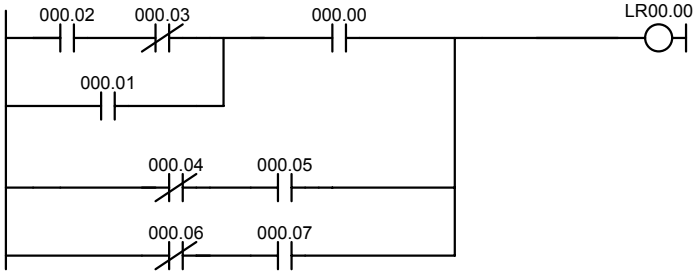

Het volgende netwerk gebruikt een OR LOAD, gevolgd door een AND LOAD om het bovenste deel van de code te programmeren, vervolgens zijn er nog twee OR LOAD's nodig om de code af te maken.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | LD | 000.01 |
| 00002 | LD | 000.02 |
| 00003 | AND NOT | 000.03 |
| 00004 | OR LD | |
| 00005 | AND LD | |
| 00006 | LD NOT | 000.04 |
| 00007 | AND | 000.05 |
| 00008 | OR LD | |
| 00009 | LD NOT | 000.06 |
| 00010 | AND | 000.07 |
| 00011 | OR LD | |
| 00012 | OUT | LR00.00 |

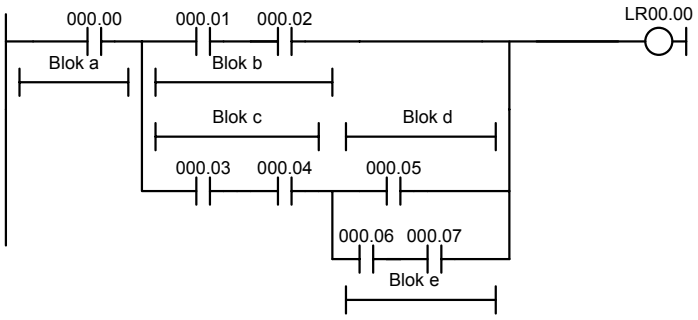
Alhoewel het programma uitgevoerd zal worden zoals het getekend is, kan het netwerk getekend worden zoals hieronder waardoor de eerste OR LOAD en AND

LOAD niet meer noodzakelijk zijn. Hierdoor wordt het programma vereenvoudigd waardoor u ruimte in het programmageheugen bespaart en het programma sneller uitgevoerd zal worden.



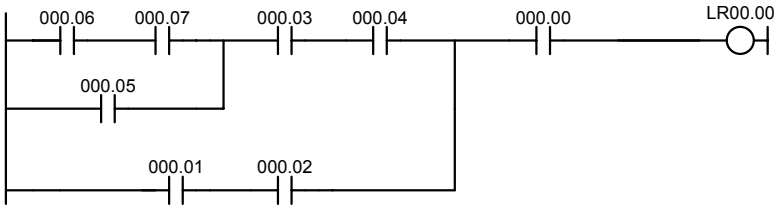

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.02 |
| 00001 | AND NOT | 000.03 |
| 00002 | OR | 000.01 |
| 00003 | AND | 000.00 |
| 00004 | LD NOT | 000.04 |
| 00005 | AND | 000.05 |
| 00006 | OR LD | |
| 00007 | LD NOT | 000.06 |
| 00008 | AND | 000.07 |
| 00009 | OR LD | |
| 00010 | OUT | LR00.00 |

Het volgende netwerk gebruikt vijf blokken, welke hier eerst in volgorde gecodeerd worden voordat OR LOAD en AND LOAD gebruikt worden om ze, vanaf de laatste twee blokken terugwerkend, samen te voegen. De OR LOAD op programma-adres 00008 voegt de blokken d en e samen, de volgende AND LOAD voegt de ontstane executieconditie samen met dat van blok c, etc.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | LD | 000.01 |
| 00002 | AND | 000.02 |
| 00003 | LD | 000.03 |
| 00004 | AND | 000.04 |
| 00005 | LD | 000.05 |
| 00006 | LD | 000.06 |
| 00007 | AND | 000.07 |
| 00008 | OR LD | |
| 00009 | AND LD | |
| 00010 | OR LD | |
| 00011 | AND LD | |
| 00012 | OUT | LR00.00 |

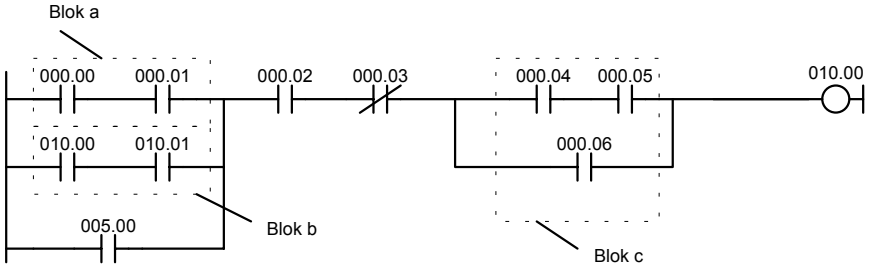
Ook dit netwerk kan hertekend worden om de programmastructuur en codering te vereenvoudigen en om programmageheugen te sparen.



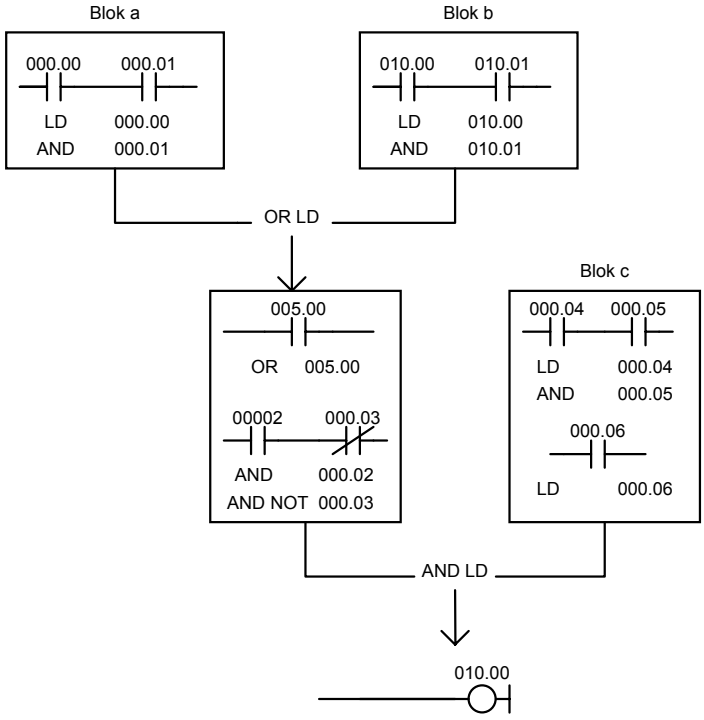

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.06 |
| 00001 | AND | 000.07 |
| 00002 | OR | 000.05 |
| 00003 | AND | 000.03 |

| | | |
|---|---|---|
| 00004 | AND | 000.04 |
| 00005 | LD | 000.01 |
| 00006 | AND | 000.02 |
| 00007 | OR LD | |
| 00008 | AND | 000.00 |
| 00009 | OUT | LR00.00 |

Het volgende en laatste voorbeeld ziet er op het eerste gezicht erg ingewikkeld uit, maar kan gecodeerd worden door gebruik te maken van slechts twee logische blok instructies. Het netwerk ziet er als volgt uit:




De eerste logische blok instructie wordt gebruikt om de executiecondities uit de blokken a en b samen te voegen, de tweede voegt de executieconditie van blok c samen met de executieconditie die ontstaat uit de normaal gesloten conditie die aan 000.03 is toegewezen. De rest van het netwerk kan gecodeerd worden met OR, AND en NOT instructies. De logische flow van het netwerk met de resulterende code is hieronder getoond.

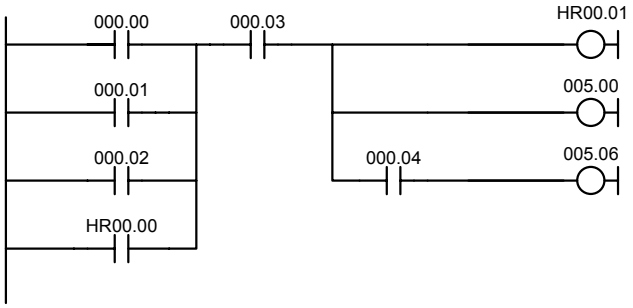



| **Adres** | **Instructie** | **Operands** |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND | 000.01 |
| 00002 | LD | 010.00 |
| 00003 | AND | 010.01 |
| 00004 | OR LD | |
| 00005 | OR | 005.00 |
| 00006 | AND | 000.02 |
| 00007 | AND NOT | 000.03 |
| 00008 | LD | 000.04 |
| 00009 | AND | 000.05 |
| 00010 | OR | 000.06 |
| 00011 | AND LD | |
| 00012 | OUT | 010.00 |

### *1.2.6 Het coderen van meerdere "uitvoerende" instructies*

Wanneer er meer dan één "uitvoerende" instructie geactiveerd moet worden door dezelfde executieconditie, dan worden ze opeenvolgend gecodeerd volgend op de

laatste conditie van de instructie regel. In het volgende voorbeeld, heeft de laatste instructie regel één conditie meer, een AND met 000.04.

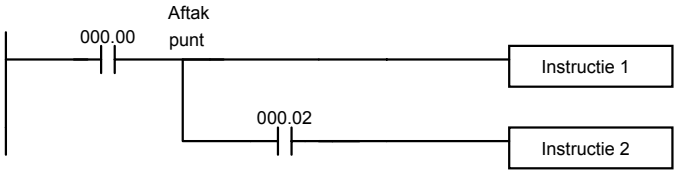



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OR | 000.01 |
| 00002 | OR | 000.02 |
| 00003 | OR | HR00.00 |
| 00004 | AND | 000.03 |
| 00005 | OUT | HR00.01 |
| 00006 | OUT | 005.00 |
| 00007 | AND | 000.04 |
| 00008 | OUT | 005.06 |

## 1.3 Programmeer overwegingen

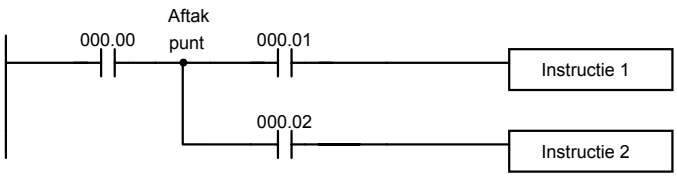
### *1.3.1 Vertakkende instructie regels*

Wanneer een instructieregel vertakt in twee of meer lijnen is het soms noodzakelijk om interlocks of TR bits te gebruiken om de executieconditie op het punt van de vertakking vast te leggen. Dit is nodig omdat de instructieregels uitgevoerd worden tot aan een “right-hand” instructie voordat er teruggegaan wordt naar het punt van de vertakking om de instructies aan de andere takken uit te voeren. Als er een conditie is opgenomen in één van de instructie regels na de aftakking, dan kan de executieconditie veranderd zijn als er weer terug gegaan wordt naar het knooppunt van de vertakking, waardoor juiste programma-uitvoer onmogelijk wordt. Het volgende netwerk illustreert dit. In beide netwerken wordt instructie 1 uitgevoerd voordat teruggesprongen wordt naar het knooppunt van de vertakking en verder wordt gegaan met de aftakking die leidt naar instructie 2.




Netwerk A: Correcte Werking

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | Instructie 1 | |
| 00002 | AND | 000.02 |
| 00003 | Instructie 2 | |




Netwerk B: Incorrecte Werking

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND | 000.01 |
| 00002 | Instructie 1 | |
| 00003 | AND | 000.02 |
| 00004 | Instructie 2 | |

Wanneer, zoals getoond is in netwerk A, de executieconditie die aanwezig was op de aftakking niet veranderd is wanneer er teruggesprongen wordt naar de aftakking (instructies aan de meest rechterzijde van het netwerk beïnvloeden de

executieconditie niet), dan zal de aftaklijn correct uitgevoerd worden en hoeven er geen speciale programmeermaatregelen genomen te worden.

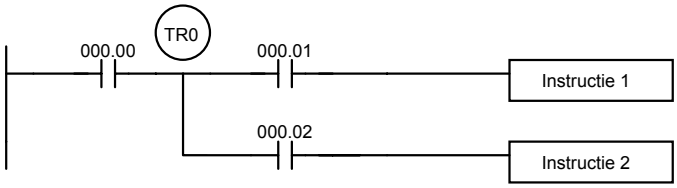
Wanneer, zoals getoond in netwerk B, een conditie is opgenomen tussen het aftakpunt en de laatste instructie op de bovenste instructieregel, dan zal de executieconditie op het aftakpunt en de executieconditie na uitvoer van de bovenste instructieregel soms anders zijn, waardoor het onmogelijk is om een correcte verwerking van de aftakking te verzekeren.

Er zijn twee manieren om programma aftakkingen te programmeren en om de executieconditie vast te houden. De ene is het gebruik van TR bits; de andere is het gebruik van interlocks (IL(02) / IL(03)).

**TR bits**

Het TR gebied voorziet in acht bits, TR 0 tot en met TR 7, die kunnen worden gebruikt om executiecondities tijdelijk op te slaan. Als een TR bit is geplaatst op een aftakpunt, dan wordt de huidige executieconditie opgeslagen op het gekozen TR bit. Wanneer er teruggekeerd wordt naar het aftakpunt, dan kan het TR bit gebruikt worden om de executiestatus, die was opgeslagen toen het aftakpunt de eerste keer werd uitgevoerd, terug te halen voor programma executie.

Het voorgaande netwerk B kan worden geschreven zoals beneden om zeker te zijn van correcte werking. In mnemonic code wordt de executieconditie opgeslagen door op het aftakpunt een TR bit te gebruiken als operand van een OUTPUT instructie. Deze executieconditie wordt vervolgens teruggehaald na de uitvoer van de “right-hand” instructie door hetzelfde TR bit te gebruiken als operand van een LOAD instructie

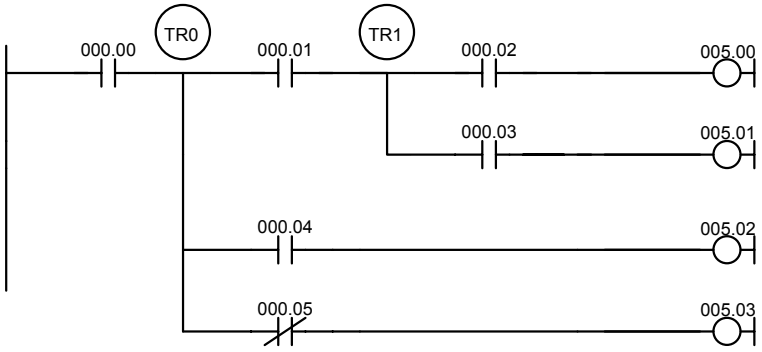



Netwerk B: Gecorrigeerd met een TR bit

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OUT | TR0 |
| 00002 | AND | 000.01 |
| 00003 | Instructie 1 | |
| 00004 | LD | TR0 |
| 00005 | AND | 000.02 |
| 00006 | Instructie 2 | |

Het bovenstaande netwerk voert de volgende actie uit: de status van 000.00 wordt geladen (een LOAD instructie) om de initiële executieconditie te bepalen. Deze executieconditie wordt vervolgens vastgelegd met een OUTPUT instructie op TR0. Hiermee wordt de executieconditie op het aftakpunt vast gelegd. Vervolgens wordt een AND uitgevoerd tussen de executieconditie en de status van 000.01 en wordt instructie 1 dienovereenkomstig uitgevoerd. De executieconditie die was vastgelegd op het aftakpunt wordt vervolgens weer geladen (een LOAD instructie met TR0 als operand), er wordt een AND functie uitgevoerd tussen deze geladen executieconditie en de status van 000.02 en instructie 2 wordt dienovereenkomstig uitgevoerd.

Het volgende voorbeeld toont een applicatie die gebruik maakt van twee TR bits.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OUT | TR0 |

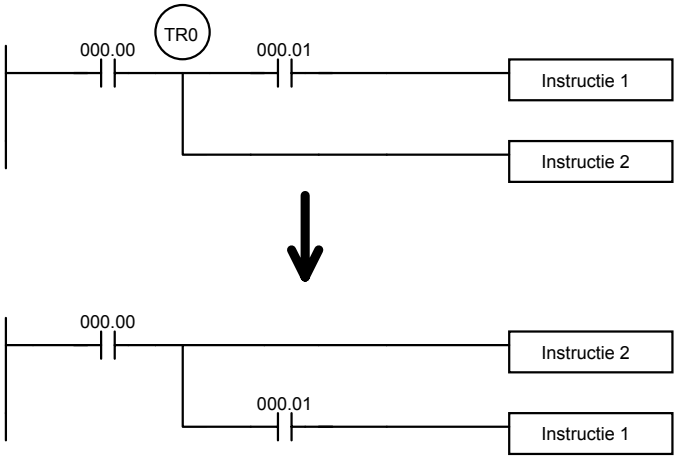
| | | |
|---|---|---|
| 00002 | AND | 000.01 |
| 00003 | OUT | TR1 |
| 00004 | AND | 000.02 |
| 00005 | OUT | 005.00 |
| 00006 | LD | TR1 |
| 00007 | AND | 000.03 |
| 00008 | OUT | 005.01 |
| 00009 | LD | TR0 |
| 00010 | AND | 000.04 |
| 00011 | OUT | 005.02 |
| 00012 | LD | TR0 |
| 00013 | AND NOT | 000.05 |
| 00014 | OUT | 005.03 |

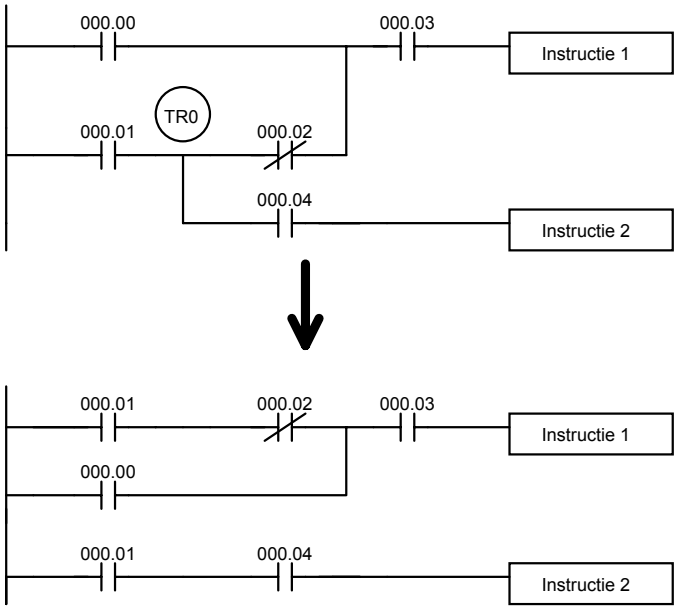
In dit voorbeeld worden TR0 en TR1 gebruikt om de executiecondities op de aftakpunten op te slaan. Na het uitvoeren van OUT 005.00 wordt de executieconditie die opgeslagen is in TR 1 geladen voor een AND met de status van 000.03. De executieconditie die opgeslagen is in TR0 wordt twee keer geladen, de eerste keer voor een AND met de status van 000.04 en de tweede keer voor een AND met de inverse van de status van 000.05.

TR bits kunnen zo vaak gebruikt worden als nodig is, zolang hetzelfde TR bit niet meer dan één keer gebruikt wordt in een instructie blok. In dit geval begint een nieuw instructie blok elke keer wanneer de programma uitvoering terugkeert naar de busbar. Als het, in een enkel instructie block, noodzakelijk is om meer dan acht verschillende aftakkingen te programmeren kunnen interlocks (die hierna uitgelegd worden) gebruikt worden. In deze situatie kan het ook aan te raden zijn om het programma te vereenvoudigen.

Wees voorzichtig, wanneer u een ladderdiagram tekent, om geen TR bits te gebruiken tenzij dit noodzakelijk is. Vaak kan het aantal instructies dat nodig is om een programma te schrijven drastisch verminderd en het programma zelf duidelijker worden door het netwerk dusdanig te tekenen dat er geen (of zo weinig mogelijk) TR bits noodzakelijk zijn. In de onderstaande voorbeelden gebruiken de netwerken geen TR relais en minder code. In het eerste voorbeeld wordt dit gerealiseerd door de onderdelen van het netwerk anders te plaatsen. Bij het tweede voorbeeld gebeurt dit door de tweede output in een eigen netwerk te plaatsen en door er een aparte LOAD instructie voor te programmeren om de juiste executieconditie te creëren.

**Opmerking** Alhoewel het vereenvoudigen van programma's altijd van belang is, is soms ook de volgorde van uitvoering van belang. Bijvoorbeeld, een MOVE instructie kan noodzakelijk zijn voor de uitvoering van een BINARY ADD instructie om de juiste data in het gebruikte operandwoord te plaatsen. Deze instructie zal dan ook voor de BINARY ADD moeten blijven staan. Overweeg altijd eerst de volgorde van uitvoering voor het programma vereenvoudigd wordt.





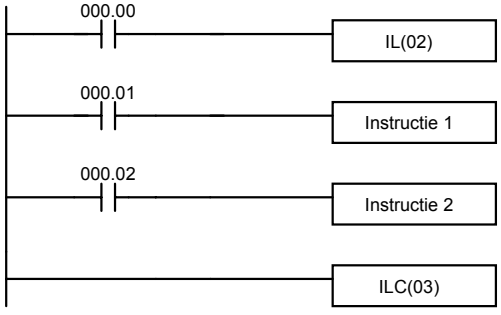

**Opmerking** TR bits worden alleen gebruikt wanneer er geprogrammeerd wordt in mnemonic code. In een ladderdiagram moet echter ook rekening worden gehouden met het aantal aftakpunten dat TR bits nodig heeft en de methoden om het aantal instructies dat nodig is om een netwerk te programmeren te verminderen. Wanneer een netwerk meer dan acht TR relais of meer dan acht AND LD of OR LD instructies gebruikt geeft SYSWIN een foutmelding.

**Interlocks**

Het probleem van de opslag van executiecondities op aftakkingen kan ook worden opgelost door de interlock (IL(O2)) en interlock clear (ILC(03)) instructies te gebruiken om het aftakpunt compleet te elimineren en toch een specifieke executieconditie een groep instructies aan te laten sturen. De interlock en interlock clear instructies worden altijd in combinatie gebruikt.

Wanneer een interlock instructie voor een sectie van een ladderdiagram programma wordt geplaatst, dan zal de executieconditie voor de interlock instructie de uitvoering van alle instructies tot aan de volgende interlock clear instructie beheren. Als de executieconditie voor de interlock instructie uit is, dan zullen alle “right-hand” instructies tot aan de volgende interlock clear instructie uitgevoerd worden met een uit conditie en zo de gehele sectie ladderdiagram resetten. Het effect dat dit heeft op bepaalde instructies is beschreven in hoofdstuk “interlocks - il(02) en ilc(03)” op pagina 82.

Netwerk B uit het hoofdstuk “vertakkende instructie regels” kan ook uitgevoerd worden met een interlock. In dit geval zullen de condities die leiden naar het aftakpunt geplaatst worden in de instructieregel voor de interlock instructie, alle regels vanaf het aftakpunt worden geschreven als aparte instructieregels en een nieuwe instructieregel wordt toegevoegd voor de interlock clear instructie. Er zijn geen condities toegestaan in de instructieregel voor de interlock clear. Merk dat zowel de interlock als de interlock clear geen operands heeft.

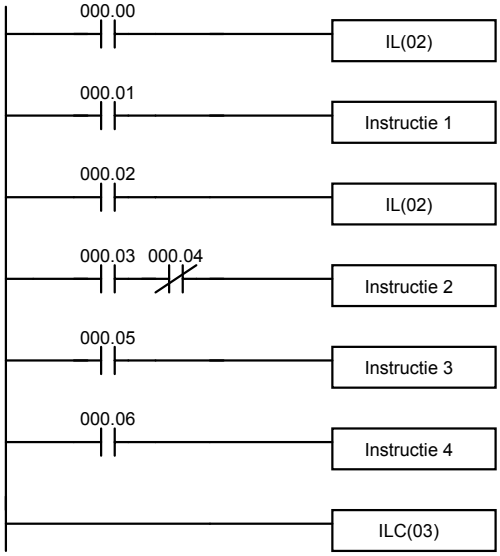



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | IL(02) | |
| 00002 | LD | 000.01 |
| 00003 | Instructie 1 | |
| 00004 | LD | 000.02 |
| 00005 | Instructie 2 | |

00006 ILC(03)

Als 000.00 aan is in de gereviseerde versie van netwerk B hierboven, dan zullen de statussen van 000.01 en 000.02 de executieconditie voor de instructies 1 en 2 bepalen. Wanneer 000.00 aan is zal dit hetzelfde resultaat geven als een AND met de status van deze bits. Als 000.00 uit is, dan zal de interlock instructie een uit executieconditie genereren voor de instructies 1 en 2 en de uitvoer van het programma zal doorgaan met de instructieregel die volgt op de interlock clear instructie.

Zoals getoond in het volgende diagram kan meer dan één interlock instructie gebruikt worden binnen een instructie blok: elke interlock is echter effectief tot de eerst volgende interlock clear instructie.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | IL(02) | |
| 00002 | LD | 000.01 |
| 00003 | Instructie 1 | |
| 00004 | LD | 000.02 |
| 00005 | IL(02) | |
| 00006 | LD | 000.03 |
| 00007 | AND NOT | 000.04 |
| 00008 | Instructie 2 | |
| 00009 | LD | 000.05 |
| 00010 | Instructie 3 | |
| 00011 | LD | 000.06 |
| 00012 | Instructie 4 | |
| 00013 | ILC(03) | |

Als 000.00 in het bovenstaande diagram uit is (d.w.z., als de executieconditie voor de eerste INTERLOCK instructie uit is), worden de instructies 1 t/m 4 uitgevoerd met uit executiecondities en zal de programma-uitvoer verder gaan met de instructie na de INTERLOCK CLEAR instructie. Als 000.00 aan is, dan zal de status van 000.01 geladen worden als executieconditie voor instructie 1 en vervolgens zal de status van 000.02 geladen worden om de executieconditie voor de tweede INTERLOCK instructie te bepalen. Als 000.02 uit is dan zullen de instructies 2 tot en met 4 uitgevoerd worden met uit executiecondities. Als 000.02 aan is, dan zullen 000.03, 000.05 en 000.06 de eerste executieconditie in de nieuwe instructieregels bepalen.

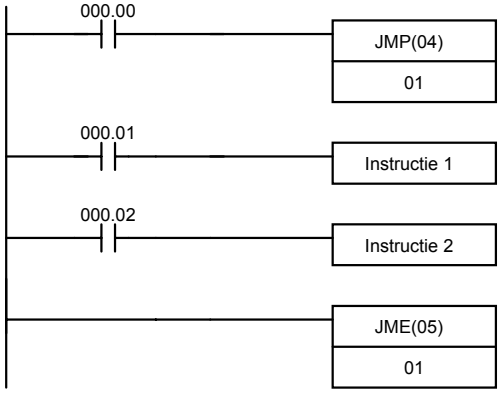
### *1.3.2 Springen*

Een specifieke sectie van een programma kan worden overgeslagen, afhankelijk van een hiervoor gebruikte executieconditie. Alhoewel het gelijk is aan wat er gebeurt wanneer de executieconditie voor een INTERLOCK instructie uit is, behouden met sprongen de operands van alle instructies hun status. Sprongen kunnen daarom gebruikt worden om apparatuur te bedienen die een aanhoudende aansturing nodig hebben, bijvoorbeeld pneumatiek and hydrauliek, terwijl interlocks gebruikt kunnen worden om apparatuur te bedienen die geen aanhoudende aansturing nodig hebben, bijvoorbeeld elektronische instrumenten.

Sprongen worden gecreëerd met de JMP(04) en JME(05) instructies. Wanneer de executieconditie voor een JMP instructie aan is, dan wordt het programma normaal uitgevoerd, alsof de sprong niet bestaat. Is de executieconditie voor de JMP instructie uit, dan verplaatst de programma uitvoering zich direct naar de JME instructie zonder dat de status van iets tussen de JMP en JME instructie verandert.

Alle JMP en JME instructies krijgen sprongnummers toegewezen die liggen tussen 00 en 49. Er zijn twee typen sprongen. Het gebruikte sprongnummer bepaalt het type van de sprong.

Een sprong met het sprongnummer 01 t/m 49 kan maar één keer gedefinieerd worden. D.w.z., elk van deze nummers mag één keer gebruikt worden in een JMP instructie en één keer gebruikt worden in een JME instructie. Wanneer een JMP instructie waaraan één van deze nummers is toegewezen wordt uitgevoerd, dan verplaatst de programma uitvoer zich direct naar de JME instructie met hetzelfde nummer, alsof het programma ertussen niet bestaat. Het netwerk B van het TR bit en interlock voorbeeld kan hertekend worden, zoals hieronder getoond, met een sprong. Alhoewel 01 gebruikt is als het sprong nummer, kan elk nummer tussen de 01 en 49 gebruikt worden zolang het niet gebruikt wordt in een ander deel van het programma. JMP en JME gebruiken geen andere operand en JME heeft nooit condities in de instructieregel ervoor.




Netwerk B: gecorrigeerd met een sprong

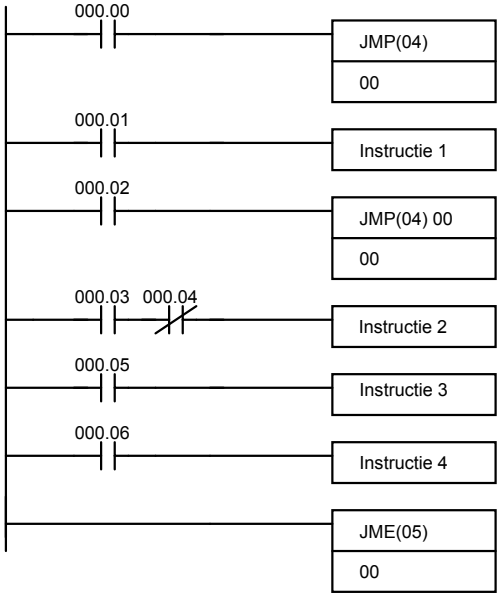
| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | JMP(04) | 01 |
| 00002 | LD | 000.01 |
| 00003 | Instructie 1 | |
| 00004 | LD | 000.02 |
| 00005 | Instructie 2 | |
| 00006 | JME(05) | 01 |

Deze versie van netwerk B zal een kortere executietijd hebben wanneer 000.00 uit is dan de andere versies.

Het andere type sprong wordt gecreëerd met een sprongnummer 00. Net zoveel sprongen als gewenst kunnen gecreëerd worden door sprongnummer 00 te gebruiken. JMP instructies met nummer 00 kunnen opeenvolgend gebruikt worden zonder dat er een JME tussen gebruikt wordt. Het is zelfs mogelijk om alle JMP 00 instructies naar dezelfde JME 00 te laten springen, dus slechts één JME 00 instructie is benodigd voor alle JMP 00 instructies in het programma. Wanneer 00 wordt gebruikt als sprongnummer voor een JMP instructie, dan wordt de programma uitvoer vervolgd bij de instructie die volgt op de JME instructie met sprongnummer 00. Alhoewel, zoals bij alle sprongen, geen statussen veranderd worden en geen instructies uitgevoerd worden tussen de JMP 00 en JME 00 instructies, zal het programma zoeken naar de volgende JME 00 instructie, waardoor een enigszins langere executie tijd wordt gecreëerd.

De uitvoering van programma's die meerdere JMP 00 instructies bevatten voor een JME 00 instructie is gelijk aan dat van het voorbeeld met meerdere interlock instructies. Het volgende voorbeeld is hetzelfde als het voorbeeld dat gebruikt is bij het interlock voorbeeld hierboven, het is alleen hertekend met sprongen. De uitvoering van dit diagram zal verschillen van het voorbeeld hierboven. In het vorige diagram zouden de interlocks bepaalde delen van het programma resetten.

Sprongen daarentegen veranderen geen enkele status tussen de JMP en JME instructies.



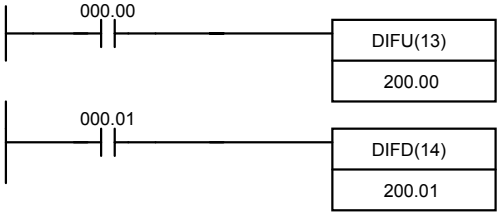

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | JMP(04) | 00 |
| 00002 | LD | 000.01 |
| 00003 | Instructie 1 | |
| 00004 | LD | 000.02 |
| 00005 | JMP(04) | 00 |
| 00006 | LD | 000.03 |
| 00007 | AND NOT | 000.04 |
| 00008 | Instructie 2 | |
| 00009 | LD | 000.05 |
| 00010 | Instructie 3 | |
| 00011 | LD | 000.06 |
| 00012 | Instructie 4 | |
| 00013 | JME(05) | 00 |

## 1.4 Bit statussen aansturen

Er zijn in het algemeen vijf instructies die gebruikt kunnen worden om individuele bits aan te sturen. Dit zijn de OUT, OUT NOT, DIFU, DIFD en KEEP instructies. Al deze instructies verschijnen als de laatste instructie in een instructieregel en gebruiken een bitadres als operand. Alhoewel details gegeven worden in de sectie “bitcontrol instructies” worden deze instructies, behalve de OUT en OUT NOT die al geïntroduceerd zijn, hier beschreven vanwege hun belangrijke functie in de meeste programma’s. Deze instructies kunnen gebruikt worden om outputbits in het IR gebied aan en uit te sturen, om signalen te geven naar externe apparatuur, maar ze kunnen ook gebruikt worden om andere bits in het IR gebied of andere bits in andere datagebieden in de PLC aan te sturen.

### *1.4.1 DIFFERENTIATE UP en DIFFERENTIATE DOWN*

DIFU en DIFD instructies worden gebruikt om het operandbit aan te sturen voor één scan op de op- of neergaande flank van de executieconditie. De DIFU instructie zet het operandbit aan voor één scan nadat de executieconditie ervoor van uit naar aan gaat (opgaande flank); De DIFD instructie zet het operandbit aan voor één scan nadat de executieconditie ervoor van aan naar uit gaat (neergaande flank).



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | DIFU(13) | 200.00 |

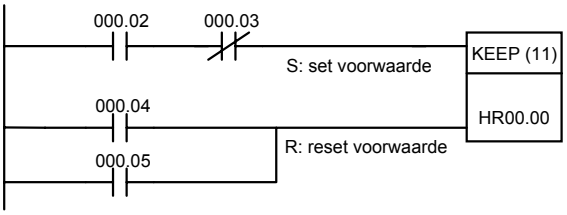
| Adres | Instructie | Operands |
|---|---|---|
| 00002 | LD | 000.01 |
| 00003 | DIFD(14) | 200.01 |

In dit voorbeeld zal 200.00 voor één scan aan gaan wanneer 000.00 aan gaat. De volgende keer dat DIFU(13) 200.00 uitgevoerd wordt gaat 200.00 uit, onafhankelijk van de status van 000.00. Bit 200.00 zal pas weer door de DIFU(13) instructie aangestuurd kunnen worden wanneer 000.00 eerst uit is geweest. Met de DIFD instructie zal 200.01 aan gezet worden voor één scan nadat 000.01 uit gaat (200.01 zal tot dan uit zijn) en zal de volgende keer dat DIFD(14) 200.01 uitgevoerd wordt uit gezet worden.

### *1.4.2 KEEP*

De KEEP instructie wordt gebruikt om de status van het operandbit vast te houden, afhankelijk van twee executiecondities. Om dit te realiseren wordt de KEEP instructie aangestuurd door twee instructieregels. Wanneer de executieconditie aan het einde van de eerste instructieregel aan is, wordt het operandbit van de KEEP instructie aan gezet. Wanneer de executieconditie aan het einde van de tweede instructieregel aan is wordt het operandbit van de KEEP instructie uit gezet. Het operandbit van de KEEP instructie zal zijn aan of uit status handhaven als de executiecondities van beide instructieregels laag zijn, zelfs wanneer deze in een interlock (tussen IL en ILC) wordt gebruikt.

In het volgende voorbeeld wordt HR00.00 aangezet als 000.02 aan is en 000.03 uit is. HR00.00 zal dan dezelfde status handhaven tot 000.04 of 000.05 aan gaat. Bij KEEP zullen, zoals bij alle instructies die meer dan één instructieregel nodig hebben, de instructieregels gecodeerd worden voor de instructie die ze aansturen.



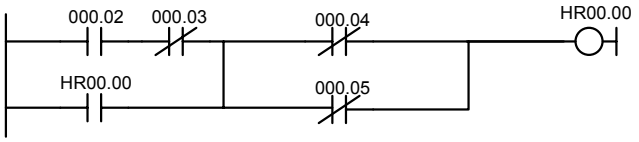

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.02 |
| 00001 | AND NOT | 000.03 |
| 00002 | LD | 000.04 |
| 00003 | OR | 000.05 |
| 00004 | KEEP(11) | HR00.00 |

### *1.4.3 Zelfhandhavende bits*

Alhoewel de KEEP instructie gebruikt kan worden om zelfhandhavende bits (houd schakelingen) te programmeren, is het soms noodzakelijk om zelfhandhavende bits te creëren op een andere manier, bijvoorbeeld zodat ze uitgezet kunnen worden in een interlock in het programma.

Om een zelfhandhavend bit te creëren, zal het operandbit van de OUT instructie aan het einde van de instructieregel als conditie voor dezelfde OUT instructie in een OR verbinding in de instructieregel opgenomen moeten worden. Hierdoor zal het operandbit van de OUT instructie zijn aan of uit status handhaven tot er een verandering optreed in de andere bits in de instructieregel. Op zijn minst moet één andere conditie gebruikt worden vlak voor de OUT instructie om als reset te fungeren. Zonder deze reset zou er geen manier zijn om het operandbit van de OUT instructie laag te maken.

Het diagram hierboven voor de KEEP instructie kan herschreven worden zoals hieronder getoond is. Het enige verschil in deze diagrammen zou hun werking in een interlock zijn als de executieconditie voor de interlock instructie uit is. Zoals in het diagram met de KEEP instructie worden ook hier twee resetbits gebruikt. HR00.00 kan dus uitgezet worden door 000.04 en 000.05.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.02 |
| 00001 | AND NOT | 000.03 |
| 00002 | OR | HR00.00 |
| 00003 | LD NOT | 000.04 |
| 00004 | OR NOT | 000.05 |
| 00005 | AND LD | |
| 00006 | OUT | HR00.00 |

## 1.5 Werkbits (interne relais)

Om tijdens het programmeren condities zo te combineren dat ze direct de juiste executiecondities genereren is vaak behoorlijk ingewikkeld. Deze moeilijkheden zijn te overbruggen door bepaalde bits te gebruiken om andere instructies indirect aan te sturen. Dit kan gerealiseerd worden door werkbits te gebruiken. Meestal worden vele woorden gebruikt voor dit doel. Deze woorden worden vaak werkwoorden genoemd.

Werkbits sturen niets aan buiten de PLC en worden ook nergens direct door aangestuurd. Het zijn bits die geselecteerd kunnen worden door de programmeur om zoals hierboven beschreven te programmeren. I/O bits en andere bits met een functie kunnen niet gebruikt worden als werkbits. Alle bits in het IR gebied die niet gebruikt worden om I/O aan te sturen, alle bits in het HR en LR geheugen en sommige ongebruikte bits in het AR geheugen kunnen gebruikt worden als werkbits. Hou altijd een bestand bij waarin u noteert waarvoor en hoe u belangrijke werkbits gebruikt. Dit helpt bij het plannen, schrijven en debuggen van een programma.

### *1.5.1 Werkbit toepassingen*

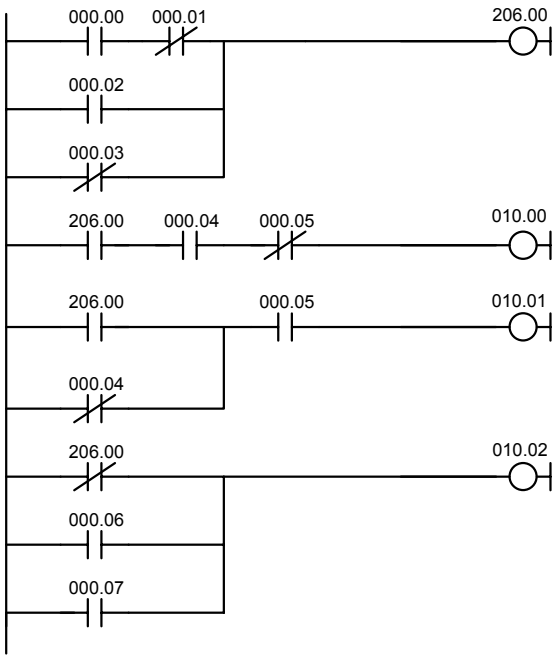
Zodra er moeilijkheden ontstaan bij het programmeren van een actie moet er rekening mee worden gehouden dat de toepassing van werkbits noodzakelijk is. Daarnaast kunnen werkbits ook gebruikt worden om een programma te vereenvoudigen.

Werkbits worden vaak gebruikt met de OUT, OUT NOT, DIFU, DIFD en KEEP instructies. Het werkbit dat wordt gebruikt als operand bij één van deze instructies kan later gebruikt worden in een conditie om te bepalen wanneer andere instructies uitgevoerd moeten worden. Werkbits kunnen ook gebruikt worden bij andere instructies, bijvoorbeeld bij de shift register instructie (SFT(10)).

Begrijpen van het gebruik van werkbits is essentieel voor effectief programmeren.

### *1.5.2 Reduceren van complexe condities*

Werkbits kunnen gebruikt worden om programma's te vereenvoudigen wanneer een bepaalde combinatie van condities vaker wordt gebruikt in combinatie met andere condities en instructies. In het volgende voorbeeld worden 000.00, 000.01, 000.02 en 000.03 gecombineerd tot een logisch blok waarvan de resulterende executieconditie wordt opgeslagen als de status van 246.00. 246.00 wordt vervolgens gebruikt met diverse andere condities om uitgangscondities te genereren voor 001.00, 001.01 en 001.02, om bijvoorbeeld de uitgangen die hieraan zijn toegewezen aan of uit te sturen.

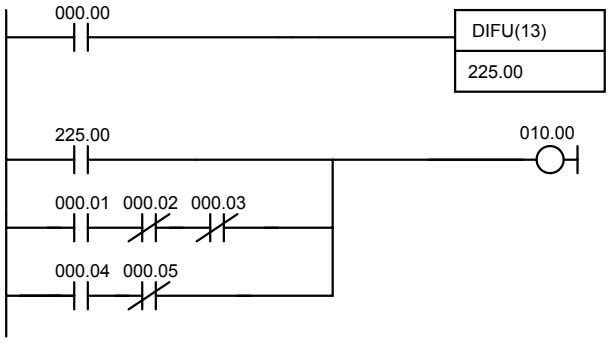



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND NOT | 000.01 |
| 00002 | OR | 000.02 |
| 00003 | OR NOT | 000.03 |
| 00004 | OUT | 206.00 |
| 00005 | LD | 206.00 |
| 00006 | AND | 000.04 |
| 00007 | AND NOT | 000.05 |
| 00008 | OUT | 001.00 |
| 00009 | LD | 206.00 |
| 00010 | OR NOT | 000.04 |
| 00011 | AND | 000.05 |
| 00012 | OUT | 001.01 |
| 00013 | LD NOT | 206.00 |
| 00014 | OR | 000.06 |
| 00015 | OR | 000.07 |
| 00016 | OUT | 001.02 |

### *1.5.3 Gedifferentieerde condities*

Werkbits kunnen ook gebruikt worden wanneer een differentiatie van een conditie gebruikt moet worden om een instructie uit te voeren. In dit voorbeeld moet 010.00 aan blijven zolang als 000.01 aan is en zowel 000.02 als 000.03 uit zijn, of zo lang als 000.04 aan is en 000.05 uit is. Het bit moet voor één scan aangezet worden als 000.00 aan gaat, tenzij een van de voorgaande condities het continu aan houdt.

Deze actie kan eenvoudig geprogrammeerd worden door 225.00 te gebruiken als een werkbit voor de operand van de DIFFERENTIATE UP instructie (DIFU(13)). Als 000.00 aan gaat, zal 225.00 voor één scan aan gezet worden en de volgende scan uit door de DIFU(13) instructie. Wanneer de andere condities die 010.00 aansturen het niet aan houden dan zal het werkbit 225.00 het bit 010.00 voor alleen één scan aan houden. (Dit is op een uitgang natuurlijk niet te zien).




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 00.000 |

| | | |
|---|---|---|
| 00001 | DIFU(13) | 225.00 |
| 00002 | LD | 225.00 |
| 00003 | LD | 000.01 |
| 00004 | AND NOT | 000.02 |
| 00005 | AND NOT | 000.03 |
| 00006 | OR LD | |
| 00007 | LD | 000.04 |
| 00008 | AND NOT | 000.05 |
| 00009 | OR LD | |
| 00010 | OUT | 010.00 |

## 1.6 Programmeer voorzorgsmaatregelen

Het aantal condities dat gebruikt kan worden in serie of parallel is ongelimiteerd, zolang het geheugen van de PLC maar niet overschreden kan worden. Programmeer daarom zoveel condities als nodig zijn om duidelijke netwerken te maken. Alhoewel met instructieregels complexe netwerken geschreven kunnen worden mogen er zich op de verticale lijnen, tussen instructieregels, in het diagram geen condities bevinden. Diagram A hieronder bijvoorbeeld is niet toegestaan en moet zoals diagram B getekend worden. De mnemonic code wordt alleen getoond voor diagram B, het coderen van diagram A is namelijk onmogelijk.

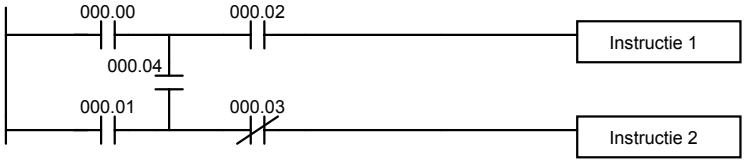



Diagram A

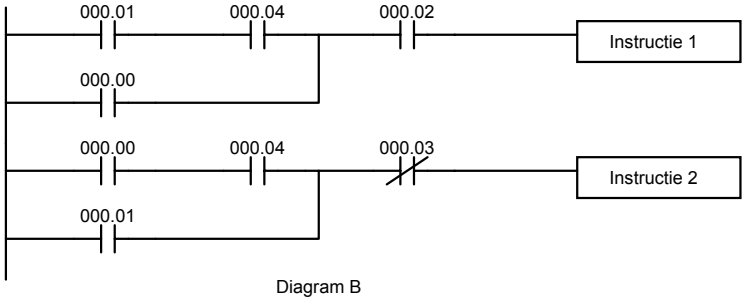
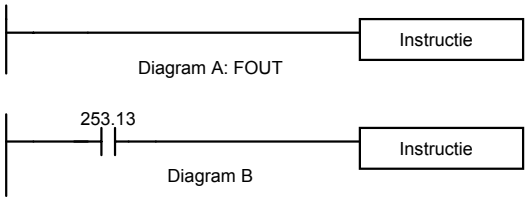



Diagram B

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.01 |
| 00001 | AND | 000.04 |
| 00002 | OR | 000.00 |
| 00003 | AND | 000.02 |
| 00004 | Instructie 1 | |
| 00005 | LD | 000.00 |
| 00006 | AND | 000.04 |
| 00007 | OR | 000.01 |
| 00008 | AND NOT | 000.03 |
| 00009 | Instructie 2 | |

Het aantal maal dat een specifiek bit toegewezen kan worden aan een conditie is niet gelimiteerd, u kunt ze dus zo vaak als noodzakelijk programmeren om uw programma te vereenvoudigen. Gecompliceerde netwerken ontstaan vaak door het zo kort mogelijk schrijven van programma's of pogingen om bits een zo min mogelijk aantal keren te gebruiken.

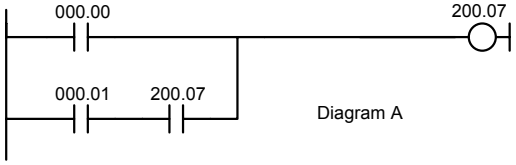
Behalve voor de instructies waarbij condities niet toegestaan zijn (bijvoorbeeld de ILC en JME, zie hieronder), moet elke instructieregel minimaal één conditie bevatten om de executieconditie voor de "uitvoerende" instructie te bepalen. Diagram A, hieronder, moet hertekend worden als diagram B. Als een instructie continu uitgevoerd moet worden, bijvoorbeeld een uitgang die altijd hoog moet zijn als het programma verwerkt wordt, kan de altijd aan vlag (Always ON) uit het SR gebied gebruikt worden.



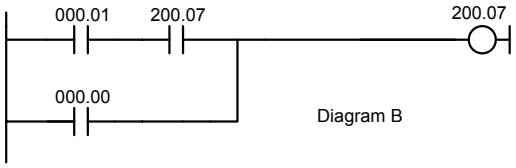

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 253.13 |
| 00001 | Instructie | |

Er zijn een paar uitzonderingen op deze regel, bijvoorbeeld de ILC, JME en STEP instructies. Elk van deze instructies wordt gebruikt als tweede instructie van een paar en wordt aangestuurd door de executieconditie van de eerste instructie van het paar. In de instructieregels naar deze instructies moeten ook geen condities opgenomen worden. Raadpleeg "Instructieset" op pagina 71 voor details.

Wanneer u ladderdiagrammen aan het tekenen bent is het van belang om het aantal instructies dat nodig is om het te coderen in gedachten te houden. In diagram A hieronder is een OR LOAD instructie nodig om de twee instructieregels samen te voegen. Dit kan voorkomen worden door het te hertekenen zoals in dagram B. Hierdoor zijn er geen AND LOAD of OR LOAD instructies nodig. Raadpleeg "AND LOAD en OR LOAD" op pagina 77 voor meer details.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | LD | 000.01 |
| 00002 | AND | 200.07 |
| 00003 | OR LD | |
| 00004 | OUT | 200.07 |




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.01 |
| 00001 | AND | 200.07 |
| 00002 | OR | 000.00 |
| 00003 | OUT | 200.07 |

## 1.7 Programma uitvoer

Wanneer de programma uitvoer wordt gestart, scant de CPU het programma van de eerste naar de laatste regel, waarbij alle condities stuk voor stuk uitgevoerd worden. Het is belangrijk dat instructies in de juiste volgorde geplaatst worden zodat bijvoorbeeld de gewenste data naar een woord verplaatst wordt voordat dit woord wordt gebruikt als operand door een instructie. Onthoud dat een instructieregel uitgevoerd wordt tot aan de afsluitende "uitvoerende" instructie voordat een aftakkende instructieregel van deze eerste regel wordt uitgevoerd tot aan de "uitvoerende" instructie. De aftakkingen worden van boven naar beneden uitgevoerd.

Het uitvoeren van het programma is één van de taken die de CPU uit moet voeren tijdens een scan. De uiteindelijk cyclustijd van het programma wordt ook nog door andere factoren bepaald.

# 2 CPM1(A) PC Setup

De PC Setup bevat diverse parameters die de werking van de CPM1(A) bepalen. Voor maximaal gebruik van de functionaliteit van de CPM1(A), bij gebruik van interrupts en communicatiefuncties, kan de PC Setup "op maat" ingesteld worden afhankelijk van de taak die uitgevoerd moet worden.

Bij het verzenden van de CPM1(A) worden alle instellingen op default gezet zodat de CPM1(A) direct gebruikt kan worden zonder dat er instellingen aangepast moeten worden. Het is echter aan te raden om deze instellingen te controleren voordat u begint met programmeren.

**Default waarden**

De default waarden voor de PC Setup is 0000 voor alle woorden. De default waarden kunnen ingesteld worden door bit 252.10 aan te zetten.

| **Voorzichtig** | Wanneer het datamemory (DM) wordt gewist met een programmeerapparaat, dan worden de settings in de PC Setup ingesteld op hun default waarde. |
|---|---|

**PC Setup veranderen**

Veranderingen in de PC Setup instellingen zijn werkzaam of beïnvloeden de werking van de CPM1(A) op verschillende momenten, afhankelijk van de veranderde instellingen.

DM6600 t/m DM6614: Alleen werkzaam wanneer de CPM1(A) powersupply aangezet wordt.

DM6615 t/m DM6644: Alleen werkzaam wanneer de het uitvoeren van het programma begint.

DM6645 t/m DM6655: Direct werkzaam op elk moment dat de CPM1(A) aan staat.

**Opmerking** Veranderingen in de PC Setup zijn alleen effectief op de hierboven aangegeven momenten. Wees er zeker van dat u de juiste maatregelen neemt voor u de veranderingen in de PC Setup aanbrengt en doorgaat met uw werkzaamheden.

Alhoewel de PC Setup is opgeslagen in DM6600 tot en met DM6655, kunnen de instellingen alleen gemaakt worden met een programmeerapparaat (bijvoorbeeld SYSWIN, handprogrammeerapparaat of SYSTOOLS). DM6600 t/m DM6644 kunnen alleen in de PROGRAM mode aangepast worden. DM6645 t/m DM6655 kunnen ingesteld worden in PROGRAM of MONITOR mode.

**Opmerking** De PC Setup kan in het programma uitgelezen en niet beschreven worden. Schrijven kan alleen met een programmeerapparaat gebeuren.

Als een PC Setup instelling verkeerd is, dan wordt een niet fatale fout (errorcode 9B) gegenereerd op het moment dat de CPM1(A) deze instelling leest. Tegelijkertijd wordt het juiste bit tussen AR24.00 en AR24.02 aan gezet. De foute instelling zal door de CPM1(A) worden gelezen als default.

**PC Setup inhoud**

De PC Setup is verdeeld in drie categorieën:

**1, 2, 3...**

1. Instellingen van de basis CPM1(A) werking en I/O processen
2. Instellingen van interrupts
3. Instellingen van communicatie

Deze sectie verklaard de instellingen naar deze classificatie. De onderstaande tabel toont de instellingen op volgorde van DM adres.

| Woord | Bit | Functie |
|---|---|---|
| **Startup processing (DM6600 t/m DM6614)**<br>Tijdens de transfer van deze data moet de PLC in program mode staan en direct na de transfer moet de PLC uit/aan gezet worden. | | |
| DM6600 | 00 t/m 07 | Startup mode (alleen actief als de 08 tot en met 15 op 02 ingesteld staan).<br>00: PROGRAM; 01: MONITOR; 02 : RUN |
| | 08 t/m 15 | Startup mode keuze<br>00: Programmeerapparaat schakelaar<br>01: Vervolg in de mode die gebruikt werd voordat de spanning uitgezet werd<br>02: Instelling in 00 t/m 07 |
| DM6601 | 00 t/m 07 | Gereserveerd (zet op 00) |
| | 08 t/m 11 | *IOM Hold Bit (252.12) Status*<br>0: Reset; 1: Handhaven |

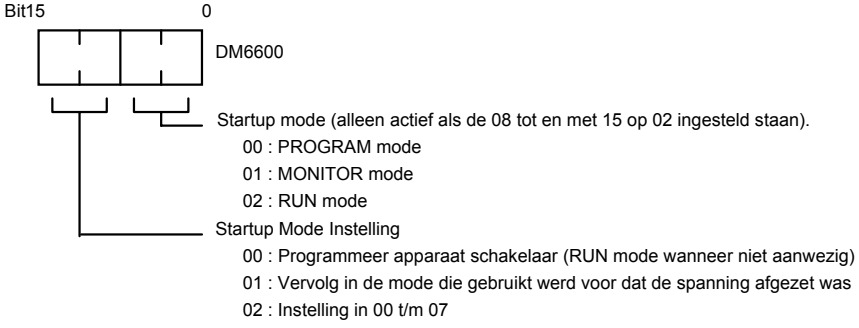
| Woord | Bit | Functie |
|---|---|---|
| | 12 t/m 15 | *Forced Status Hold Bit (252.11) Status*<br>0: Reset; 1: Handhaven |
| DM6602 | 00 t/m 03 | Programma geheugen write protect<br>0: Programma geheugen niet tegen overschrijven beveiligd<br>1: Programma geheugen tegen overschrijven beveiligd |
| | 04 t/m 07 | Taal van het handprogrammeerapparaat<br>0: Engels, 1: Japans |
| | 08 t/m 11 | Veranderen van de uitbreidingsinstructies<br>0: Veranderen is niet mogelijk<br>1: Veranderen is mogelijk |
| | 12 t/m 15 | Gereserveerd |
| DM6603 t/m DM6614 | 00 t/m 15 | Gereserveerd |
| **Puls uitgang en cyclustijd instellingen (DM6615 t/m DM6619)**<br>Tijdens de transfer van deze data moet de PLC in program mode staan. | | |
| DM6615 t/m DM6616 | 00 t/m 15 | Gereserveerd |
| DM6617 | 00 t/m 07 | Servicingtijd voor de periferie poort (alleen actief wanneer de bits 08 t/m 15 op 01 staan)<br>00 t/m 99 (BCD); Percentage van de cyclustijd gebruikt om de periferie poort te servicen |
| | 08 t/m 15 | Periferie poort servicing instelling activeren<br>00: Stel geen servicetijd in<br>01: Gebruik de tijd in 00 t/m 07 |
| DM6618 | 00 t/m 07 | Cyclus monitor tijd (actief wanneer de bits 08 t/m 15 ingesteld staan op 01, 02 of 03)<br>00 t/m 99 (BCD): Instelling (zie 08 t/m 15) |
| | 08 t/m 15 | Cyclus monitor activeren (Instelling in 00 t/m 07 x unit; 99s max.)<br>00: 120ms (instelling in bits 00 t/m 07 niet actief)<br>01: Instelling unit: 10ms<br>02: Instelling unit: 100ms<br>03: Instelling unit: 1s |
| DM6619 | 00 t/m 15 | Cyclustijd<br>0000: Variabel (geen minimum)<br>0001 t/m 9999 (BCD): Minimum tijd in ms |
| **Interrupt processing (DM6620 t/m DM6639)**<br>Tijdens de transfer van deze data moet de PLC in program mode staan. | | |
| DM6620 | 00 t/m 03 | Ingangstijdvertraging voor 000.00 t/m 000.02<br>00: 8ms; 01: 1ms; 02: 2ms; 03: 4ms; 05: 16ms; 06: 32ms; 07: 64ms; 08: 128ms |
| | 04 t/m 07 | Ingangstijdvertraging voor 000.03 en 000.04 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| | 08 t/m 11 | Ingangstijdvertraging voor 000.05 en 000.06 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| | 12 t/m 15 | Ingangstijdvertraging voor 000.07 t/m 000.11 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| DM6621 | 00 t/m 07 | Ingangstijdvertraging voor 001 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| | 08 t/m 15 | Ingangstijdvertraging voor 002 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| DM6622 | 00 t/m 07 | Ingangstijdvertraging voor 003 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| | 08 t/m 15 | Ingangstijdvertraging voor 004 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| DM6623 | 00 t/m 07 | Ingangstijdvertraging voor 005 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| | 08 t/m 15 | Ingangstijdvertraging voor 006 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| DM6624 | 00 t/m 07 | Ingangstijdvertraging voor 007 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| | 08 t/m 15 | Ingangstijdvertraging voor 008 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| DM6625 | 00 t/m 07 | Ingangstijdvertraging voor 009 (Zelfde instelling als de bits 00 t/m 03 van DM6620) |
| | 08 t/m 15 | Gereserveerd |
| DM6626 t/m DM6627 | 00 t/m 15 | Gereserveerd |
| DM6628 | 00 t/m 03 | Interrupt activeren voor 000.03 (0: Normale input; 1: Interrupt input, 2: Quick respons) |
| | 04 t/m 07 | Interrupt activeren voor 000.04 (0: Normale input; 1: Interrupt input, 2: Quick respons) |
| | 08 t/m 11 | Interrupt activeren voor 000.05 (0: Normale input; 1: Interrupt input, 2: Quick respons) |
| | 11 t/m 15 | Interrupt activeren voor 000.06 (0: Normale input; 1: Interrupt input, 2: Quick respons) |
| DM6629 t/m DM6639 | 00 t/m 14 | Gereserveerd |
| **Highspeed counter instellingen (DM6640 t/m DM6644)**<br>Tijdens de transfer van deze data moet de PLC in program mode staan. | | |
| DM6640 t/m DM6641 | 00 t/m 15 | Gereserveerd |
| DM6642 | 00 t/m 03 | Highspeed counter 0 mode<br>0: Up/down counter mode, 4: Incrementele counter mode |
| | 04 t/m 07 | Highspeed counter 0 reset mode<br>0: Z fase en software reset, 1: Alleen software reset |

| Woord | Bit | Functie |
|---|---|---|
| | 08 t/m 15 | Highspeed counter 0 activeren<br>00: Gebruik de counter niet, 01: gebruik de counter zoals ingesteld op 00 t/m 07 |
| DM6643 t/m DM6644 | 00 t/m 15 | Gereserveerd |
| **Periferie poort instellingen**<br>De volgende instellingen zijn direct actief na transfer naar de PLC. | | |
| DM6645 t/m DM6649 | 00 t/m 15 | Gereserveerd |
| DM6650 | 00 t/m 07 | Poort instellingen<br>00: Standaard (1 start bit, 7 data bits, even pariteit, 2 stop bits, 9.600 bps)<br>01: instellingen in DM6651 |
| | 08 t/m 11 | Link gebied voor en 1:1 Link op de periferie poort<br>0: LR00 t/m LR15 |
| | 12 t/m 15 | Communicatie mode<br>0: Hostlink; 2: 1:1 link slave, 3: 1:1 link master, 4: NT link |
| DM6651 | 00 t/m 07 | Baudrate<br>00: 1.2K, 01: 2.4K, 02:4.8K, 03: 9.6K, 04: 19.2K |
| | | Frame formaat<br>Waarde Start Lengte Stop Pariteit<br>00: 1 bit 7 bits 1 bit even<br>01: 1 bit 7 bits 1 bit oneven<br>02: 1 bit 7 bits 1 bit geen<br>03: 1 bit 7 bits 2 bit even<br>04: 1 bit 7 bits 2 bit oneven<br>05: 1 bit 7 bits 2 bit geen<br>06: 1 bit 8 bits 1 bit even<br>07: 1 bit 8 bits 1 bit oneven<br>08: 1 bit 8 bits 1 bit geen<br>09: 1 bit 8 bits 2 bit even<br>10: 1 bit 8 bits 2 bit oneven<br>11: 1 bit 8 bits 2 bit geen |
| DM6652 | 00 t/m 15 | Transmissie vertraging (Hostlink)<br>0000 t/m 9999: in ms |
| DM6653 | 00 t/m 07 | Node nummer (Host link) |
| | 08 t/m 15 | Gereserveerd |
| DM6654 | 00 t/m 15 | Gereserveerd |
| **Errorlog instellingen (DM6655)**<br>De volgende instellingen zijn direct actief na transfer naar de PLC. | | |
| DM6655 | 00 t/m 03 | Stijl<br>0: Schuif door nadat 10 records zijn opgeslagen (FIFO)<br>1: Sla alleen de eerste 10 records op<br>2 t/m F: Sla geen records op |
| | 04 t/m 07 | Gereserveerd |
| | 08 t/m 11 | Cyclus tijd monitor activeren<br>0: Detecteer lange cyclussen als niet fatale errors<br>1: Detecteer te lange cyclussen niet |
| | 12 t/m 15 | Gereserveerd |

## 2.1 Basis CPM1(A) werking en I/O afhandeling

**Startup mode (DM6600)**

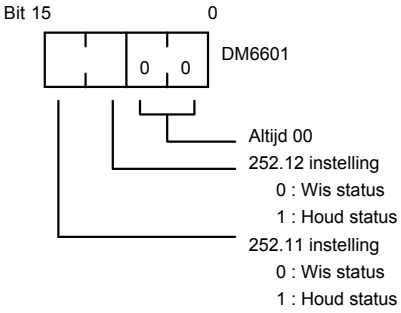
De mode waarin de CPM1(A) zal starten als de spanning aangezet wordt, kan volgens de hieronder getoonde methode ingesteld worden.




**Default:** Programming Console Mode Selector

**Hold bit status (DM6601)**

Maak de hieronder getoonde instellingen om te bepalen of, wanneer de powersupply aangezet wordt, het *Forced Status Hold Bit* (252.11) en/of het *IOM Hold Bit* ( 25212) de status zullen handhaven die actief was toen de spanning werd uitgezet of dat de vorige status gewist wordt.



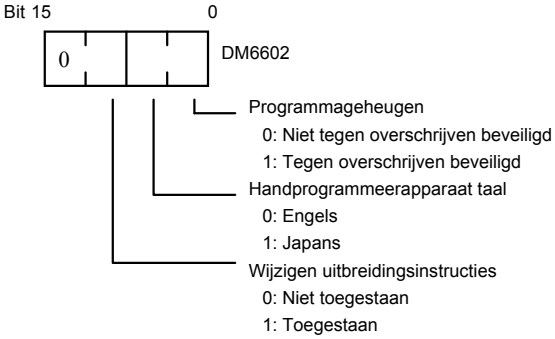

**Default:** Beide wissen

Het *Forced Status Hold Bit* (252.11) bepaalt of de geforceerde set/reset statussen worden vastgehouden bij het veranderen van PROGRAM mode naar MONITOR mode.

Het *IOM Hold Bit* (252.12) bepaalt of de status van IR bits en LR bits wordt vastgehouden wanneer de werking van de CPM1(A) wordt gestart en gestopt.

**Programma geheugen tegen overschijven beschermen (DM6602)**

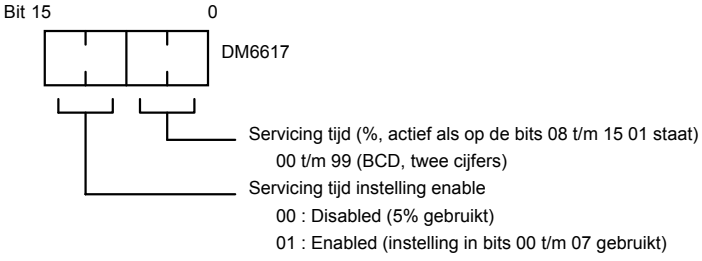
Op DM6602 kan worden ingesteld of het programmageheugen van de CPM1(A) mag worden overschreven. Tevens kan hier de taal van het handprogrammeerapparaat worden ingesteld en of er wijzigingen mogen worden aangebracht in de uitbreidingsinstructies.




**Default:** Programma overschrijfbaar, taal is Engels en uitbreidingsinstructies zijn niet te wijzigen.

**Periferie poort servicing tijden (DM6617)**

De volgende instelling wordt gebruikt om het percentage van de cyclustijd te bepalen dat wordt gebruikt om de periferie poort te servicen.



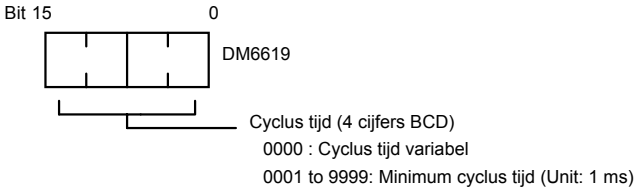

**Default:** 5% van de cyclus tijd

**Voorbeeld**

Als DM6617 wordt ingesteld op 0110, dan zal de periferie poort in 10% van de cyclustijd geserviced worden. Wanneer DM6617 wordt ingesteld op 0115, dan zal de periferie poort in 15% van de cyclustijd geserviced worden. De servicingtijd is minimaal 0,34ms. De hele servicingtijd wordt alleen gebruikt wanneer dit noodzakelijk is.

**Cyclustijd (DM6619)**

Maak de instellingen die hieronder getoond zijn om de cyclustijd te normaliseren en variaties in de I/O responsetijd te elimineren door een minimum cyclustijd in te stellen.

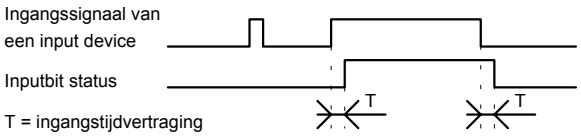



**Default:** Cyclus tijd variabel

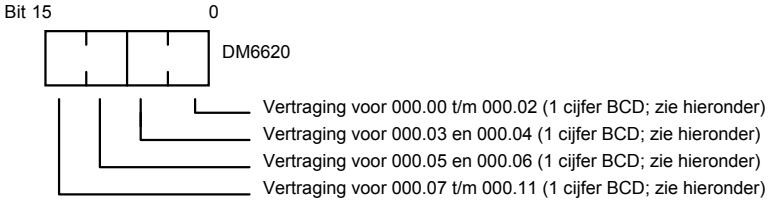
Als de actuele cyclustijd korter is dan de minimum ingestelde cyclustijd, dan zal de uitvoering van het programma wachten tot de minimum tijd is verstreken. Wanneer de actuele cyclustijd langer is dan de minimum cyclustijd, dan zal de uitvoering doorgaan en de minimum cyclustijd overschreden worden. AR24.05 wordt hoog wanneer de minimum cyclustijd, de vorige scan, is overschreden.

**Ingangstijdvertraging (DM6620 t/m DM6625)**

Maak de hieronder getoonde instellingen om de tijd in te stellen tussen het aan/uit gaan van de actuele ingangen van de inputunit en het updaten van het bij deze ingang horende inputbit. Verander deze instelling wanneer u de ingangstijd-vertraging wilt verlengen of verkorten om respectievelijk een stabieler ingangs-signaal of een snellere response op een ingang te krijgen.

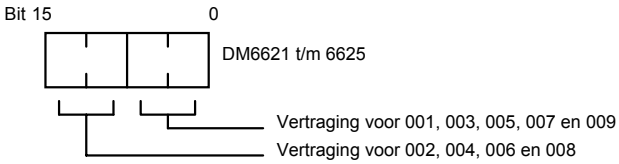



**Ingangstijdvertraging voor input woord 000**




**Default:** 8 ms voor elke ingang

**Ingangstijdvertraging voor input woord 001 t/m 009**




**Default:** 8 ms voor elke ingang

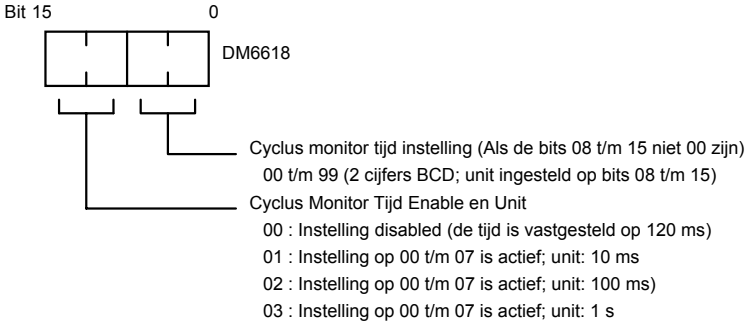
De negen mogelijke instellingen voor de ingangstijdvertraging worden hieronder getoond.

| | | | | |
|---|---|---|---|---|
| 00 : 8ms | 01 : 1ms | 02 : 2ms | 03 : 4ms | 04 : 8ms |
| 05 : 16ms | 06 : 32ms | 07 : 64ms | 08 : 128ms | |

**Errorlog instellingen**

Maak de hieronder getoonde instellingen om errors te detecteren en op te slaan in de errorlog.

**Cyclus monitor tijd (DM6618)**




**Default:** 120ms

De cyclus monitor tijd wordt gebruikt om te controleren op extreem lange cyclustijden, zoals bijvoorbeeld kan gebeuren als het programma in een oneindige loop terecht komt. Als de cyclustijd de cyclus monitor instelling overschrijdt, wordt een fatale error (FALS 9F) gegenereerd.

**1, 2, 3...**

1. De meeteenheden die gebruikt wordt voor de maximum en huidige cyclustijden, zoals opgeslagen in AR26 en AR27 veranderen afhankelijk van de units die ingesteld zijn voor de cyclus monitor tijd zoals hieronder getoond.
   Bits 08 t/m 15 ingesteld op 00 of 01: 0,1ms
   Bits 08 t/m 15 ingesteld op 02: 1ms
   Bits 08 t/m 15 ingesteld op 03: 10ms
2. Zelfs als de cyclus tijd 1s of langer duurt, zal de cyclustijd zoals uitgelezen met een programmeerapparaat niet boven de 999,9ms komen. De correcte maximale en huidige cyclustijden zullen in AR26 en AR27 opgeslagen worden.

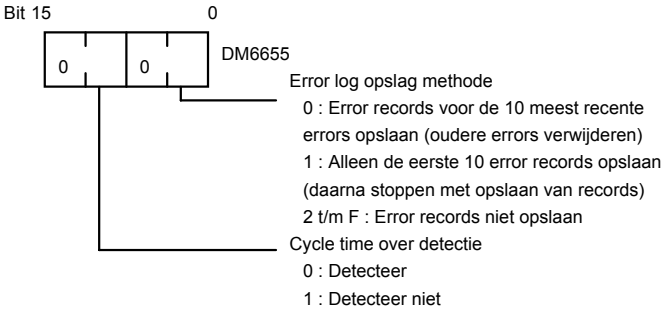
**Voorbeeld**

Als 0230 is ingesteld op DM6618, zal een FALS 9F error niet optreden tot de cyclustijd groter is dan 3s. Als de actuele cyclustijd 2.59s is, zal de inhoud van AR27 2590 (ms) zijn en de cyclustijd zoals uitgelezen met een programmeerapparaat 999.9ms.

Een "cycle time over" error (niet fatale) wordt gegenereerd als de cyclustijd boven de 100ms komt, tenzij de detectie van te lange cyclustijden is gedeactiveerd met de instelling hiervoor op DM6655. Cycle time overrun errors zijn niet fatale errors.

**Errordetectie en errorlog werking (DM6655)**

Maak de hieronder getoonde instellingen om te bepalen of een niet fatale error gegenereerd moet worden wanneer de cyclustijd boven de 100ms komt en de manier waarop records opgeslagen moeten worden in de errorlog als ze optreden.




**Default:** Cycle time over errors worden gedetecteerd en de 10 tien meest recente error records worden opgeslagen.

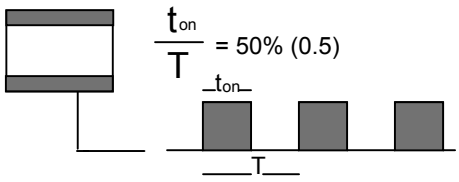
## 2.2 CPM1A pulsuitgang functie instellen en gebruik

Deze sectie verklaart de instellingen en methoden voor het gebruik van de CPM1A pulsuitgang functie. Alle CPM1A PLC's kunnen standaard pulsen uitsturen op een uitgang. De standaard pulsuitgangen hebben een duty-ratio ($t_{on}$/T) van 50%. De frequentie die uitgestuurd kan worden ligt tussen de 20 en 2000Hz.

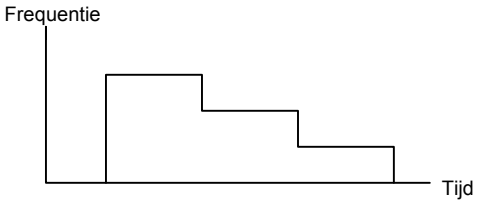
**Opmerking**

Alleen de CPM1A PLC typen beschikken over de pulsuitgang functie. De CPM1 PLC typen beschikken niet over deze functie.

Standaard kunnen pulsen uitgestuurd worden op een gespecificeerde uitgang door de SPED(—) instructie. De pulsen kunnen vanuit één uitgang tegelijk uitgestuurd worden. Het volgende diagram toont de pulsen die uitgestuurd worden op een uitgang van CPM1A. De duty-ratio van de puls uitsturing is 50% en de frequentie kan ingesteld worden van 20Hz t/m 2kHz.




Wanneer pulsen worden uitgestuurd op een uitgang kan de frequentie in stappen veranderd worden door de SPED(—) instructie opnieuw uit te voeren met verschillende frequenties, zoals in het volgende diagram getoond wordt.



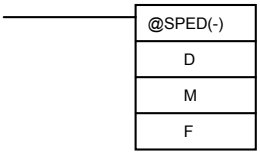

Er zijn twee manieren op de puls uitsturing te stoppen.

**1, 2, 3...**

1. Na het uitvoeren van SPED(—), zal de puls uitsturing stoppen als een INI(—) wordt uitgevoerd met C=003 of wanneer SPED(—) opnieuw wordt uitgevoerd waarbij de frequentie ingesteld is op 0.
2. Het totale aantal pulsen dat uitgestuurd moet worden kan worden ingesteld met PULS(—) voor de executie van SPED(—). In dit geval moet SPED(—) uitgevoerd worden in de independent mode. De puls uitsturing stopt automatisch wanneer het aantal pulsen, ingesteld met PULS(—), uitgestuurd is.

**Continue puls output**

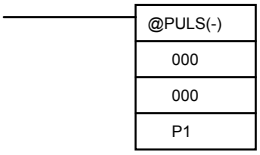
Pulsen worden uitgestuurd op de gespecificeerde uitgang als SPED(—) wordt uitgevoerd. Stel de gekozen uitgang in met het nummer ervan 00 t/m 01 (D=000 t/m 001) en de frequentie van 20Hz t/m 2000Hz (F=0002 t/m 0200). Zet de mode op continuous mode (M=001).



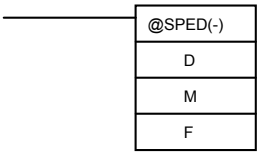

De puls uitsturing kan gestopt worden door INI(—) uit te voeren met C=003 of door SPED(—) opnieuw uit te voeren waarbij de frequentie ingesteld is op 0. De frequentie kan veranderd worden door SPED(—) opnieuw uit te voeren met een andere frequentie instelling.

**Aantal pulsen instellen**

Het totale aantal pulsen dat uitgestuurd wordt kan worden ingesteld met PULS(—) voor het uitvoeren van SPED(—) in de independent mode. De pulsuitgang zal automatisch stoppen als het aantal pulsen, ingesteld met PULS(—), is uitgestuurd.




PULS(—) stelt het 8-cijferig aantal pulsen in met P1+1, P1. Het aantal pulsen kan worden ingesteld van 00000001 t/m 16777215. Het aantal pulsen dat ingesteld is met PULS(—) wordt uitgestuurd als SPED(—) wordt uitgevoerd in de independent mode. Het aantal pulsen kan niet worden veranderd tot alle pulsen zijn uitgestuurd. Het is echter wel mogelijk om de puls uitsturing te stoppen.




Wanneer SPED(—) wordt uitgevoerd, zullen pulsen uitgestuurd worden op de gespecificeerde uitgang (D=000 t/m 001: bit 00 t/m 01) met de gespecificeerde frequentie (F=0002 t/m 0200: 20Hz t/m 2000Hz). Zet de mode op independent mode (M=001) om het aantal pulsen uit te sturen dat met PULS(—) is ingesteld. De frequentie kan veranderd worden door SPED(—) opnieuw uit te voeren met een andere frequentie instelling.

**Frequentie veranderen**

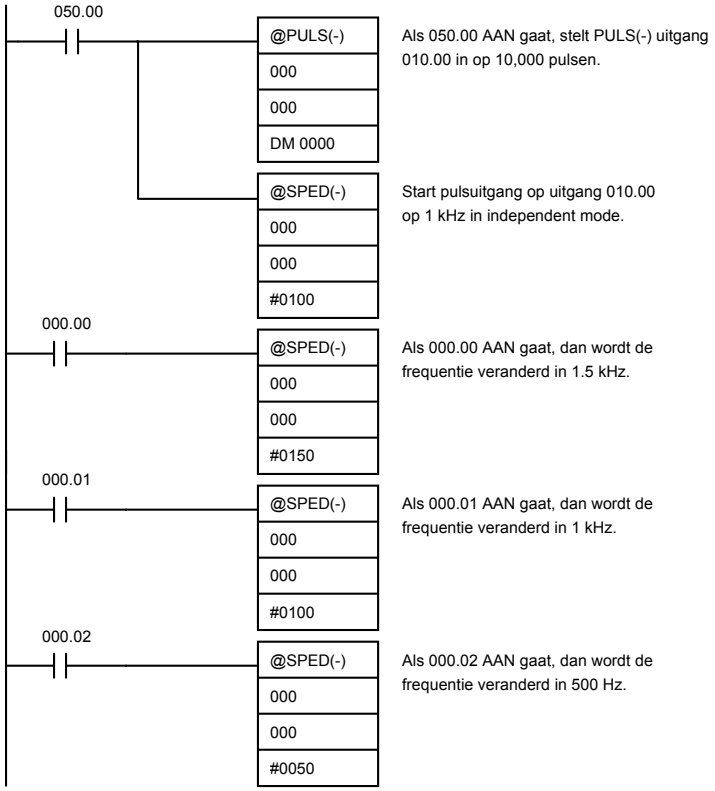
De frequentie van de pulsuitgang kan veranderd worden door SPED(—) opnieuw uit te voeren met een andere frequentie instelling. Gebruik dezelfde uitgang (P) en mode (M) instellingen die gebruikt zijn om de pulsuitgang te starten. De nieuwe frequentie kan elke frequentie zijn tussen de 20Hz t/m 2000Hz in units van 10Hz (F=0002 t/m 0200).

**Voorbeeld 1: pulsuitgang met PULS(-) en SPED(-)**

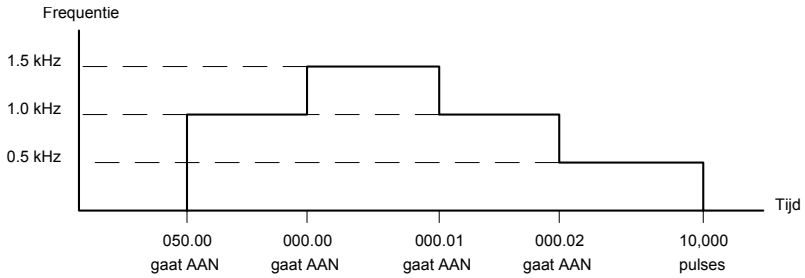
Het volgende voorbeeld toont PULS(-) en SPED(-), gebruikt om puls uitsturing op uitgang 010.00 realiseren. Het aantal pulsen gespecificeerd met PULS(-) (10,000)

**starten**

worden uitgestuurd als de frequentie wordt veranderd door het uitvoeren van SPED(-) met verschillende frequentie instellingen.




Het volgende diagram toont de pulsuitgang frequentie als het programma wordt uitgevoerd.

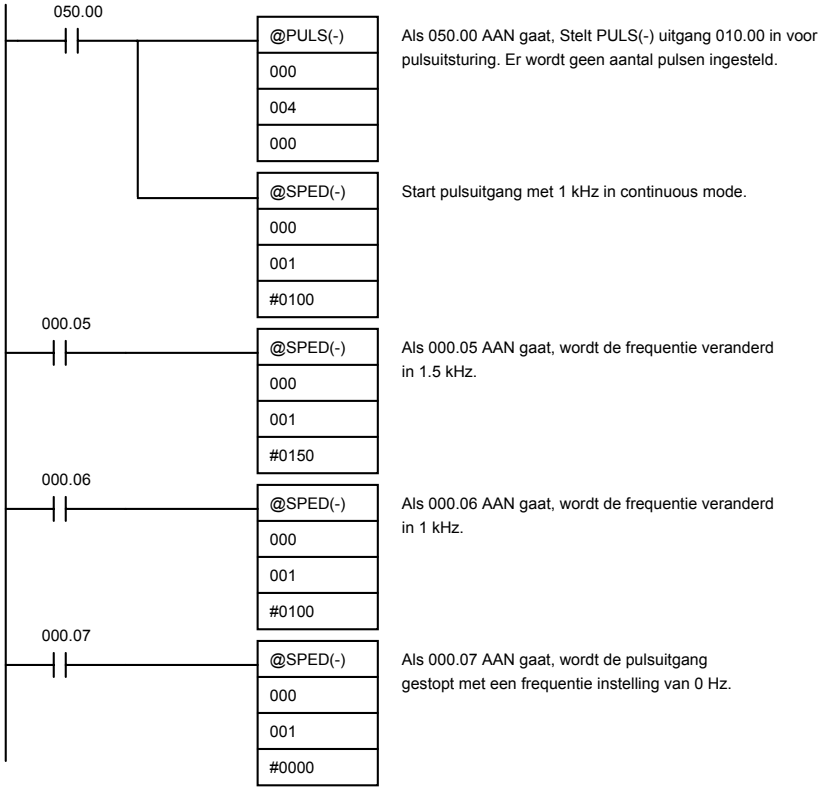



| **Let op!** | Wees er zeker van dat de uitgestuurde frequentie boven de minimum frequentie van het aangestuurde systeem ligt. |
|---|---|

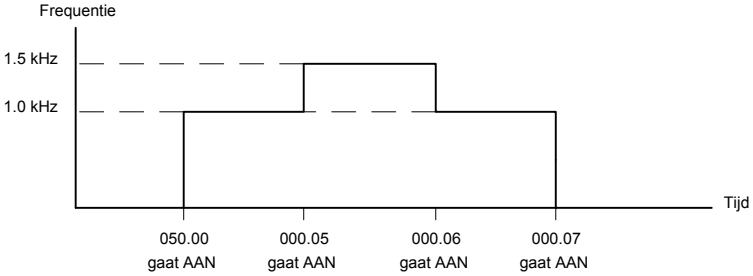
**Opmerking** Speed control timing kan erg nauwkeurig zijn wanneer de frequentie veranderingen worden uitgevoerd als interrupt processen.

**Voorbeeld 2: stoppen van pulsuitgang met SPED(-)**

Het volgende voorbeeld toont PULS(-) en SPED(-), gebruikt om de pulsuitgang op uitgang 010.00 aan te sturen. De frequentie wordt veranderd door het uitvoeren van SPED(-) met verschillende frequentie instellingen en uiteindelijk gestopt met een frequentie instelling van 0.




Het volgende diagram toont de pulsuitgang frequentie van poort 1 als het programma wordt uitgevoerd.




| | |
|---|---|
| **Voorzichtig** | Wees er zeker van dat de uitgestuurde frequentie boven de minimum frequentie van het aangestuurde systeem ligt. |

## 2.3 Instellen en gebruik van de CPM1(A) interrupt functies

Deze sectie verklaart de instellingen en methodes voor gebruik van de CPM1(A)'s interrupt functies.

### *2.3.1 Interrupt typen*

De CPM1(A) heeft drie typen interrupts.

**Input interrupts** Input interrupts worden uitgevoerd wanneer een signaal van een extern apparaat één van de CPU ingangen 000.03 t/m 000.06 aan zet.

**Interval timer interrupts:** Interval timer interrupts worden uitgevoerd door een interval timer met een precisie van 0,1 ms.

**Highspeed counter interrupts:** Highspeed counter interrupt wordt uitgevoerd afhankelijk van de huidige waarde (actuele waarde) van de ingebouwde highspeed counter. Alle CPM1(A) CPU's beschikken over highspeed counter 0, die pulsen telt via de CPU ingangen 000.00 t/m 000.02. Phase differential pulsen tot 2.5 kHz of één fase pulsen tot 5 kHz kunnen worden geteld.

**Interrupt afhandeling** Wanneer een interrupt wordt gegenereerd wordt de gespecificeerde interrupt-routine uitgevoerd. Interrupts hebben de volgende prioriteitsvolgorde (Input interrupt 0 heeft de hoogste prioriteit en highspeed counter interrupt 0 heeft de laagste).

Input interrupt 0 > Input interrupt 1 > Input interrupt 2 > Input interrupt 3> Interval timer interrupts > highspeed counter interrupt

Wanneer een interrupt met een hogere prioriteit wordt ontvangen gedurende interrupt uitvoer, dan zal het huidige proces worden gestopt en de nieuw ontvangen interrupt uitgevoerd worden. Nadat deze routine in zijn geheel is uitgevoerd zal de uitvoer van de vorige interrupt hervat worden.

Wanneer een interrupt met een lagere of gelijke prioriteit wordt ontvangen tijdens interrupt uitvoer, dan wordt de nieuw ontvangen interrupt direct uitgevoerd nadat de routine die op dat moment uitgevoerd wordt geheel uitgevoerd is.

**Voorzorgen**

Wanneer u met interrupts gaat werken moet u de volgende voorzorgen treffen:

**1, 2, 3...**

1. In een interrupt programma kan een nieuwe interrupt gedefinieerd worden. Het is ook mogelijk om een interrupt te wissen in een interrupt programma.
2. Een ander interrupt programma kan niet in een interrupt programma geschreven worden.
3. In een interrupt programma kan geen subroutine geschreven worden. Schrijf geen SBN(92), subroutine definitie, instructie in een interrupt programma.
4. Een interrupt programma kan niet worden geschreven in een subroutine. Schrijf geen interrupt programma tussen een SBN(92) en RET(93) instructie.

Ingangen die zijn gebruikt als interrupt ingang kunnen niet als normale ingang worden gebruikt.

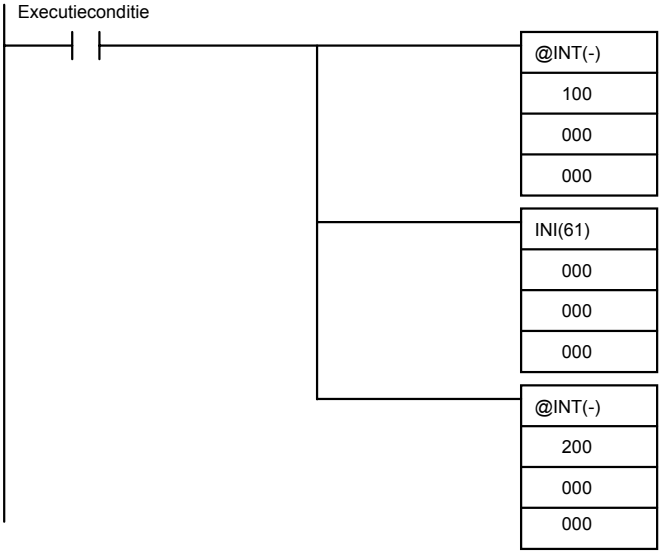
**Highspeed counter instructies en interrupts**

De volgende instructies kunnen niet worden uitgevoerd in een interruptsubroutine als een instructie die de highspeed counter aanstuurt wordt uitgevoerd in het hoofdprogramma: (255.03 gaat aan).
INI(—), PRV(—), CTBL(—)

De volgende methodes kunnen gebruikt worden om deze beperking te omzeilen:

**Methode 1**

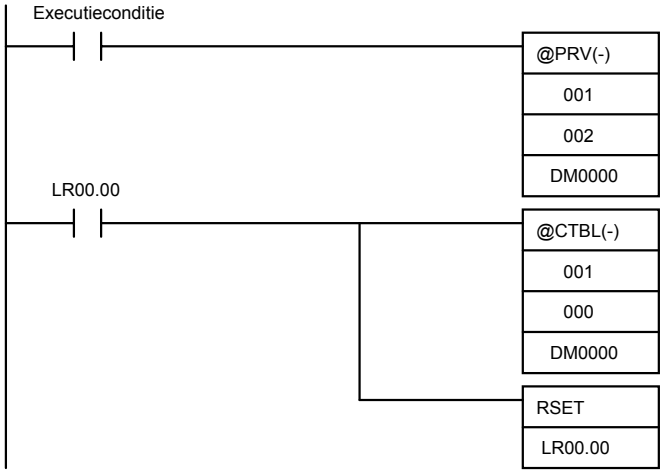
Alle interruptuitvoer kan gemaskeerd worden terwijl de instructie wordt uitgevoerd.



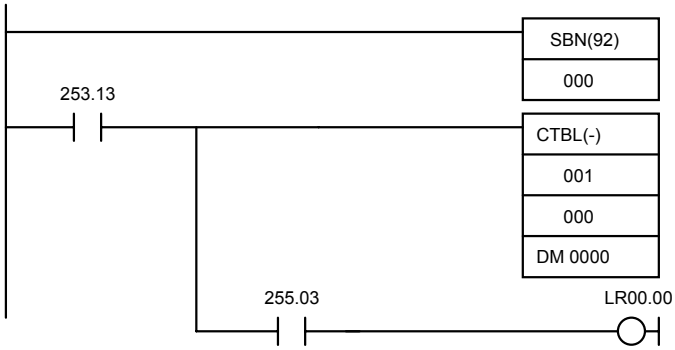

**Methode 2**

Voer de instructie opnieuw uit in het hoofdprogramma.

Dit is de programmasectie uit het hoofd programma:




Dit is de programmasectie uit de interrupt subroutine:




**Opmerking**

1. Net zoals bij normale subroutines worden interruptroutines gedefinieerd door gebruik te maken van SBN(92) en RET(93) instructies aan het einde van het hoofdprogramma.
2. Wanneer een interruptroutine is gedefinieerd dan zal een "no SBS error" worden gegenereerd tijdens een programmacheck, maar de uitvoer zal normaal verlopen. Controleer als deze error voorkomt alle normale subroutines om er zeker van te zijn dat in het programma een SBS(91) is geprogrammeerd om deze aan te roepen voor u verder gaat.

### *2.3.2 Input interrupts*

De CPM1(A)-10 heeft twee interrupt ingangen (000.03 en 000.04), terwijl de CPM1(A)-20, 30 en 40 vier interrupt ingangen hebben (000.03 t/m 000.06).

**Uitvoer**

Er zijn twee modes voor uitvoer van Input interrupts. De eerste is de Input Interrupt Mode, waarin de interrupt wordt uitgevoerd als respons op een extern signaal (ingang van de PLC). De tweede is de Counter Mode, waarin de signalen van het externe apparaat op hoge snelheid worden geteld en een interrupt wordt gegenereerd na elke bepaalde hoeveelheid signalen.

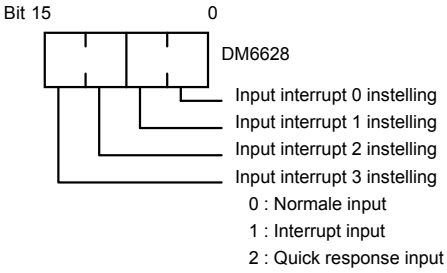
In de Input Interrupt Mode kunnen signalen met een lengte van 300μs of meer gedetecteerd worden. In de Counter Mode kunnen signalen tot 1kHz geteld worden.

**PC Setup parameters**

Wanneer ingang 000.03 t/m 000.06 worden gebruikt als interrupt ingangen moeten deze eerst in de PC Setup geactiveerd worden. Voor het programma uitgevoerd kan worden moet eerst de volgende instelling, in PROGRAM mode, veranderd worden in de PC Setup.

**Interrupt input instellingen (DM6628)**

Als deze instellingen niet gemaakt zijn kunnen de interrupts niet in het programma gebruikt worden.




**Default:** Alle ingangen normaal

**Interrupt subroutines**

Interrupts van ingang 000.03 t/m 000.06 krijgen de interrupt nummer 0 t/m 3 toegewezen en activeren de subroutines 0 t/m 3. Wanneer een interrupt niet wordt gebruikt kunnen de subroutines 0 t/m 3 gewoon in het PLC programma gebruikt worden.

| PLC Model | Interrupt ingang | Interrupt nummer | Responsetijd | | interrupt subroutine |
|---|---|---|---|---|---|
| | | | Interrupt mode | Counter mode | |
| CPM1(A)-10 | 000.03<br>000.04 | 00<br>01 | 0,3 ms max tijd tot het interrupt programma wordt geacti-veerd | 1 kHz | 000<br>001 |
| CPM1(A)-20/30/40 | 000.03<br>000.04<br>000.05<br>000.06 | 00<br>01<br>02<br>03 | | | 000<br>001<br>002<br>003 |

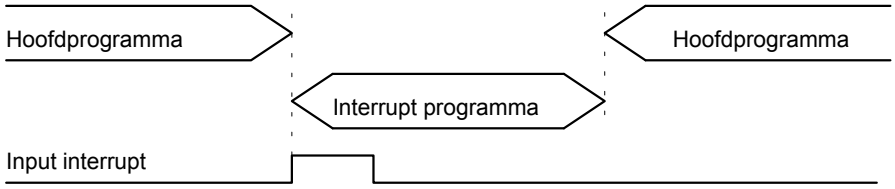
**Input refreshing**

Wanneer input refreshing niet gebruikt wordt, dan is de status van ingangen onbetrouwbaar in een interrupt routine. Afhankelijk van de ingestelde ingangstijdvertraging kunnen ingangen, zelfs als de ingangen gerefreshed wordt, niet aan gaan. Dit geldt ook voor de status van de ingang die de interrupt activeerde.

Bijvoorbeeld, in interrupt subroutine 000 zal de status van ingang 000.03 uit zijn als de interrupt geactiveerd wordt. In dit geval is het aan te raden om 253.13 (altijd hoog) te gebruiken in plaats van ingang 000.03.

**Input Interrupt Mode**

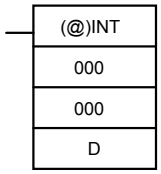
Wanneer een input interrupt signaal wordt ontvangen, dan wordt het hoofdprogramma onderbroken en het interrupt programma direct uitgevoerd, onafhankelijk van het punt binnen de cyclus waar de interrupt binnenkomt. Het interrupt signaal moet minimaal 200µs aanwezig zijn om gedetecteerd te kunnen worden.




Gebruik de volgende instructies om Input Interrupts te programmeren in de Input Interrupt Mode.

**Maskeren van interrupts**

Met de INT(—) instructie is het mogelijk om Input Interrupt maskers te zetten of te wissen zoals benodigd is.




Maak de instellingen met de bits 00 t/m 03 van D,
deze corresponden met de input interrupts 0 to 3.
0 : Masker gewist (Input interrupt is toegestaan)
1 : Masker gezet (Input interrupt niet toegestaan)

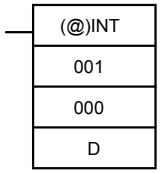
Tijdens het opstarten van programma bij de CPM1(A) worden alle Input Interrupts gemaskeerd.

**Wissen van gemaskeerde interrupts**

Als het bij een Input Interrupt behorende bit aan gaat terwijl de interrupt gemaskeerd is, dan zal deze interrupt worden opgeslagen in het geheugen en zal deze uitgevoerd worden zodra het masker gewist wordt. Wanneer een opgeslagen Input Interrupt niet moet worden uitgevoerd als het masker gewist wordt, dan moet deze interrupt uit het geheugen gewist worden.

Alleen één interrupt signaal wordt in het geheugen opgeslagen per interrupt nummer.

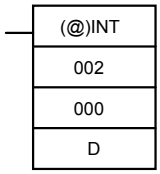
Met de INT(—) instructie is het mogelijk om een Input Interrupt uit het geheugen te wissen.




Als de bits 0 t/m 3 uit D, die corresponderen met input interrupts 0 t/m 3 op "1" zijn ingesteld, dan worden de input interrupts gewist uit het geheugen.
0 : Input interrupt bewaard.
1 : Input interrupt gewist.

**Uitlezen van de masker status**

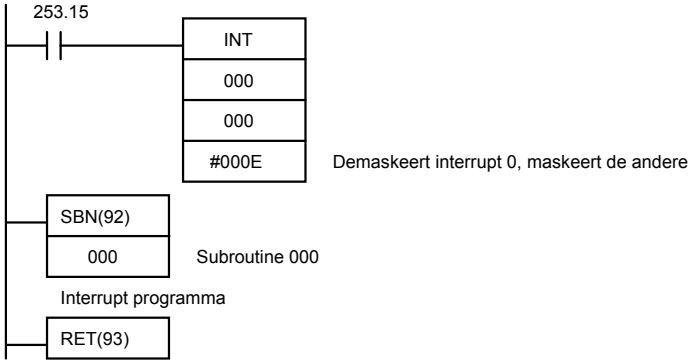
Met de INT(—) instructie kan de status van het Input Interrupt masker uitgelezen worden.




De status van de meest rechtse bits van de in woord D opgeslagen data (0 t/m 3) data tonen de masker mask status.
0 : Masker gewist (Input interrupt toegestaan)
1 : Masker gezet (Input interrupt niet toegestaan)

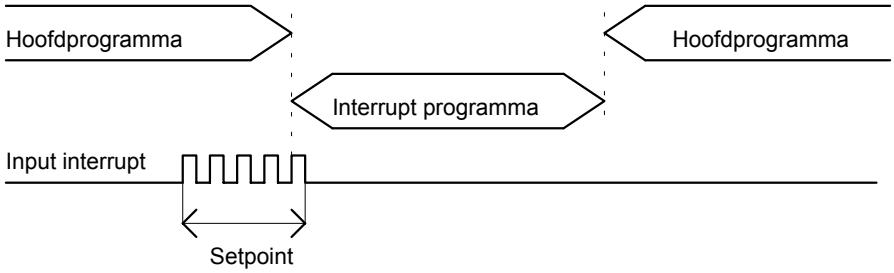
**Voorbeeld**

Wanneer ingang 000.03 (Interrupt nr 0) aan gaat, dan zal de programma uitvoer voor het interrupt programma in subroutine nummer 0 direct worden gestart. DM6628 moet voor dit voorbeeld worden ingesteld op 0001.




**Counter mode**

Externe signalen worden geteld en een interrupt wordt gegenereerd wanneer de tel waarde een opgegeven setpoint bereikt. Wanneer een interrupt wordt geactiveerd, dan wordt het hoofdprogramma onderbroken en het interrupt programma direct uitgevoerd, onafhankelijk van het punt binnen de cyclus waar de interrupt binnenkomt. Externe signalen met een frequentie tot 1kHz worden geteld.




Gebruik de volgende stappen om Input Interrupts te programmeren die de Counter Mode gebruiken.

**Opmerking** De SR woorden die in de Counter Mode gebruikt worden (240 t/m 243) bevatten allemaal binaire (hexadecimale) data, geen BCD.

**1, 2, 3...**

1. Schrijf de ingestelde waarde voor de counter werking op de bij interrupt 0 t/m 3 horende SR woorden. De ingestelde waarden kunnen worden opgegeven tussen 0000 en FFFF (0 t/m 65535). Een waarde van 0000 zal de teller werking uitzetten tot een nieuwe waarde wordt ingesteld en stap 2, hieronder getoond, wordt herhaald.

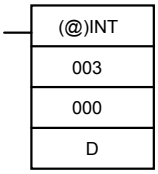
**Opmerking** De onderstaande SR woorden worden gewist tijden het opstarten van de PLC en moeten vanuit het programma geschreven worden.

Het maximale input signaal dat geteld kan worden is 1 kHz.

| Interrupt | Woord |
|---|---|
| Input interrupt 0 | 240 |
| Input interrupt 1 | 241 |
| Input interrupt 2 | 242 |
| Input interrupt 3 | 243 |

Als de Counter mode niet wordt gebruikt, kunnen deze SR bits gebruikt worden als werkbits.

2. Met de INT(—) instructie kan de Counter Mode ingestelde waarde gerefreshed worden en de interrupts geactiveerd.




Als de bits 0 t/m 3 uit D, die corresponderen met input interrupts 0 to 3, zijn ingesteld op "0", dan wordt de ingestelde waarde gerefreshed en zijn interrupts toegestaan.
0 : Counter mode ingestelde waarde gerefreshed en masker gewist.
1 : Niets gebeurd (Zet de bits voor alle interrupts die niet veranderen op "1").

De Input Interrupt waarvoor de ingestelde waarde wordt gerefreshed wordt geactiveerd in de Counter Mode. Wanneer de counter de ingestelde waarde bereikt, wordt een interrupt gegenereerd, de counter wordt gereset en zal weer vanaf de ingestelde waarde terug tellen. Dit gaat door tot de counter werking wordt gestopt.

**1, 2, 3...**

1. Als de INT(—) instructie wordt gebruikt tijdens het tellen, dan wordt de actuele waarde terug gezet op de ingestelde waarde. U moet daarom de gedifferentieerde variant van deze instructie gebruiken (@INT), anders zal er mogelijk nooit een interrupt plaats vinden.
2. De ingestelde waarde wordt geactiveerd als de INT(—) instructie wordt uitgevoerd. De ingestelde waarde kan dus niet veranderd worden door simpelweg de inhoud van 240 t/m 243 te veranderen. Als de inhoud van deze registers is veranderd moet de ingestelde waarde gerefreshed worden door de INT instructie opnieuw uit te voeren.

Counter Interrupts kunnen op dezelfde manier gemaskeerd worden als in de Input Interrupt Mode, worden echter de maskers op deze manier gewist, dan zal de Counter Mode niet gehandhaafd worden en de Input interrupt Mode zal geactiveerd worden. Interrupt signalen die ontvangen zijn voor gemaskeerde interrupts kunnen gewist worden op dezelfde manier als bij de Input Interrupt mode.

**Counter actuele waarde in counter mode**

Wanneer input interrupts worden gebruikt in counter mode, dan wordt de actuele waarde van de counter opgeslagen in het SR woord dat hoort bij de Input Interrupt 0 t/m 3. De waardes zijn 0000 t/m FFFE (0 t/m 65534), de uitgelezen waarde is gelijk aan de counter actuele waarde min één.

| Interrupt | Woord |
|---|---|
| Input interrupt 0 | 244 |
| Input interrupt 1 | 245 |
| Input interrupt 2 | 246 |
| Input interrupt 3 | 247 |

**Voorbeeld**

De actuele waarde voor een interrupt wiens ingestelde waarde 000A is zal getoond worden als een actuele waarde van 0009 direct nadat INT(—) is uitgevoerd.

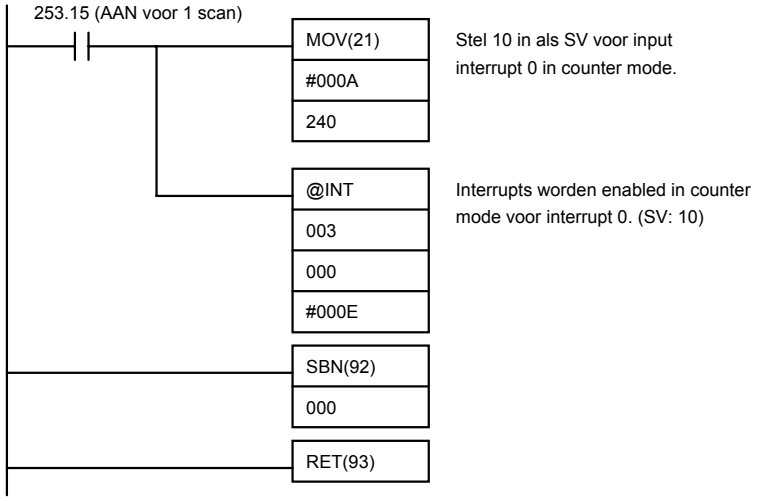
**Opmerking**

Zelfs als de Input Interrupts niet gebruikt worden in Counter Mode, zijn deze SR bits niet te gebruiken als werkbits.

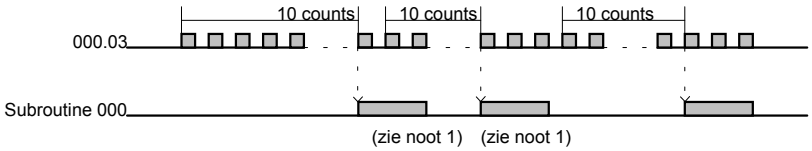
**Voorbeeld**

In dit voorbeeld wordt Input Interrupt 0 gebruikt in de Counter Mode. Controleer voor het uitvoeren van het programma de PC Setup. Op ingang 000.03 moeten 10 pulsen worden ingevoerd, waarna een interrupt wordt gegenereerd.

PC Setup: DM6628: 0001 (000.03 is gebruikt als Interrupt) De default instellingen zijn gebruikt voor alle andere PC Setup parameters.




Wanneer het programma wordt uitgevoerd, dan zal de werking zijn zoals in het onderstaande diagram is getoond.




**Noot**

1. De counter zal doorgaan met werken, zelfs als de interruptroutine verwerkt wordt.

### *2.3.3 Alle interrupts maskeren*

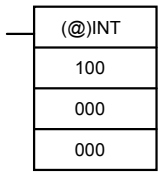
Alle interrupts, inclusief Input Interrupts, Interval Timer Interrupts en highspeed counter interrupts, kunnen als groep ge(de)maskeerd worden met de INT(—) instructie. Deze methode is een toevoeging op het maskeren van individuele soorten interrupts. Verder zal het wissen van de maskeringen voor alle interrupts niet de maskeringen van de individuele interrupts wissen, maar ze herstellen naar de gemaskeerde conditie die bestond voordat INT(—) werd uitgevoerd om ze als groep te maskeren.

Gebruik INT(—) niet om interrupts als groep te maskeren, tenzij dit nodig is om tijdelijk alle interrupts te maskeren. Gebruik INT(—) instructies in dit geval altijd in paren, waarbij de eerste INT(—) instructie de interrupts maskeert en de tweede deze bewerking ongedaan maakt.

INT(—) kan niet gebruikt worden om alle interrupts te (de)maskeren vanuit een interrupt routines.

**Interrupts maskeren**

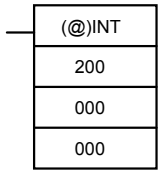
Gebruik de INT(—) instructie om alle interrupts te deactiveren.




Als een interrupt wordt gegenereerd wanneer interrupts gemaskeerd zijn, dan zal interrupt processing niet worden uitgevoerd maar de gegenereerde interrupt zal worden opgeslagen voor de input, interval timer en highspeed counter interrupts. De interrupts worden vervolgens geserviced zodra deze gedemaskeerd worden.

**Interrupts demaskeren**

Gebruik de INT(—) instructie zoals hieronder getoond om interrupts te demaskeren.



### *2.3.4 Interval timer interrupts*

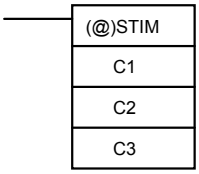
Snelle, nauwkeurig timer interrupts kunnen worden gegenereerd door gebruik te maken van interval timers. De CPM1(A) beschikt over één interval timer.

**Modes**

Er zijn twee modes waarin de interval timer kan werken. De one-shot Mode, waarin één interrupt wordt uitgevoerd als de tijd verstreken is en de scheduled Interrupt Mode waarin de interrupt wordt herhaald op een vast interval.

**Opstarten in de one-shot mode**

Gebruik de STIM (—) instructie om de interval timer in de one-shot mode te starten.




C1: Interval timer nummer (000).
C2: Timer ingestelde waarde (eerste woord adres)
C3: Subroutine nr. (4 cijfers BCD): 0000 t/m 0255

$C_2$: Timer ingestelde waarde (4 cijfers BCD): 0000 t/m 9999

$C_2 + 1$: Timer tijdsinterval (4 cijfers BCD; unit: 0,1ms): 0005 t/m 0320 (0,5ms t/m 32ms).

Elke keer als het interval gespecificeerd in $C_2 + 1$ afloopt, zal de timer de actuele waarde met één verlagen. Wanneer de actuele waarde 0 bereikt, zal de opgegeven subroutine éénmaal worden aangeroepen en de timer stoppen.

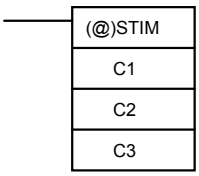
De tijd die verloopt tussen het starten van de STIM instructie en de afloop er van kan als volgt worden berekend:

(Inhoud van woord $C_2$) x (Inhoud van woord $C_2 + 1$) x 0,1ms =(0,5 t/m 319968,0ms)

Als een constante is ingesteld op $C_2$, dan zal de ingestelde waarde van de timer die waarde nemen en het interval zal 10 zijn (1ms; de ingestelde waarde wordt uitgedrukt in ms)

**Opstarten in de scheduled interrupt mode**

Gebruik de STIM(—) instructie om de interval timer in de scheduled interrupt mode te starten.




C1: Interval timer nr (003)
C2: Timer ingestelde waarde (eerste woord)
C3: Subroutine nr. (4 cijfers BCD): 0000 t/m 0255

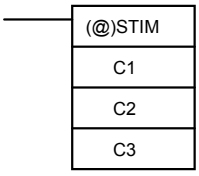
$C_2$: Timer ingestelde waarde (4 cijfers BCD): 0000 t/m 9999

$C_2 + 1$: Timer tijdsinterval (4 cijfers BCD; unit: 0,1ms): 0005 t/m 0320 (0,5ms t/m 32ms).

De betekenis van de instellingen blijft hetzelfde als bij de one-shot mode, maar in de scheduled interrupt mode zal de actuele waarde van de timer teruggezet worden op de ingestelde waarde en zal de timer opnieuw gestart worden nadat de interrupt subroutine is uitgevoerd. In de scheduled interrupt mode worden interrupts repeterend uitgevoerd op een vast interval tot de uitvoer wordt gestopt.

**Uitlezen van de timers verlopen tijd**

Gebruik de STIM(—) instructie om de verlopen tijd van de timer uit te lezen.




C1: Interval timer nr. (006)
C2: Eerste woord van parameter 1
C3: Parameter 2

$C_2$: Aantal keer dat de waarde van de timer is verlaagd (4 cijfers BCD)

$C_2 + 1$: Timer tijdsinterval (4 cijfers BCD; unit: 0.1 ms)

$C_3$: De verlopen tijd sinds de vorige verlaging van de timer waarde (4 cijfers BCD; unit: 0.1 ms)

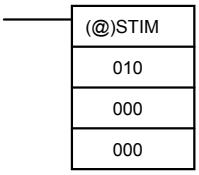
De tijd die verlopen is sinds de interval timer is gestart kan als volgt berekend worden:

{(Inhoud van woord C2) x (Inhoud van woord C2 + 1) + (Inhoud van woord C3)} x 0.1 ms

Als de gespecificeerde interval timer is gestopt dan wordt "0000" opgeslagen.

**Stoppen van timers**

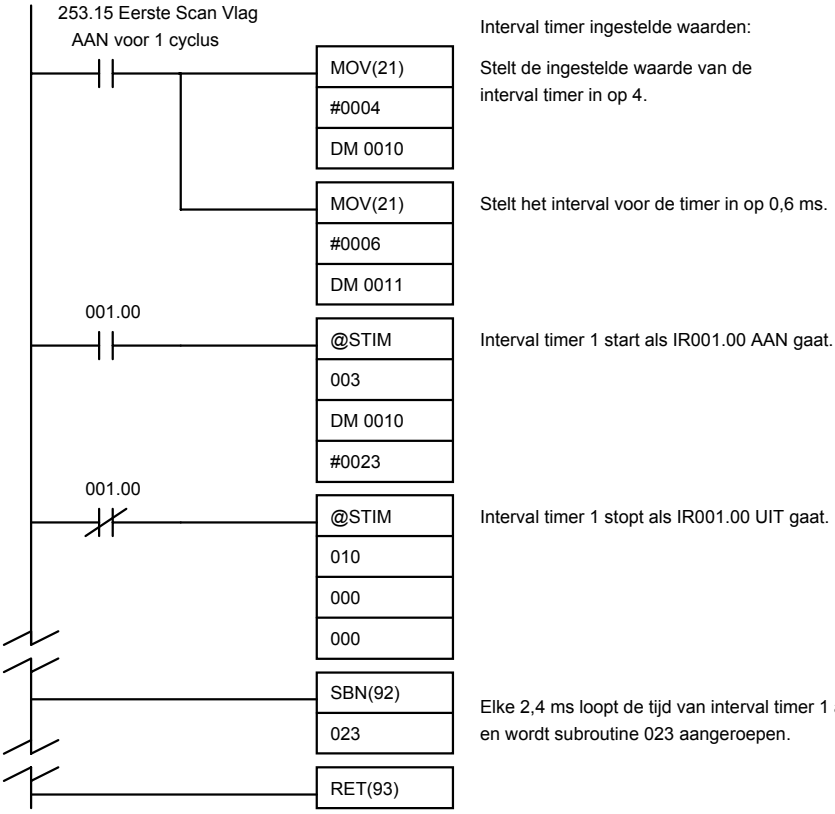
Gebruik de STIM(—) instructie om de interval timer te stoppen.



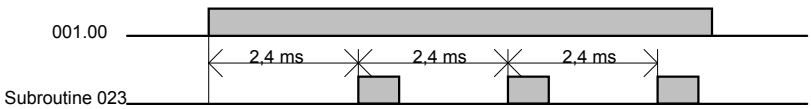

Na uitvoer van de STIM instructie zoals hierboven getoond zal de interval timer stoppen.

**Voorbeeld**

In dit voorbeeld wordt een interrupt om de 2,4ms (0,6ms x 4) aangeroepen door middel van een interval timer. Zet voor het uitvoeren van het programma alle parameters in de PC Setup op default. (De ingangen worden niet gerefreshed voor de uitvoer van de interrupt subroutine.)

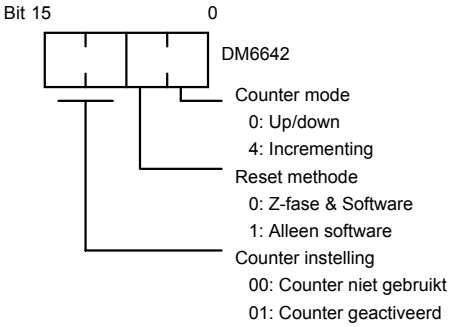



Wanneer het programma wordt uitgevoerd dan wordt subroutine 023 elke 2,4ms aangeroepen zolang 001.00 aan is.




## *2.3.5 Highspeed counter interrupts*

Pulsen van een encoder die aangesloten is op de ingangen 000.00 t/m 000.02 van de CPM1(A) CPU kunnen met een hoge snelheid geteld worden. Het is mogelijk om, afhankelijk van de getelde waarde, interrupts uit te voeren. Voor de counter gebruikt kan worden moet deze in de PC Setup geactiveerd worden.




**Default:** Counter is niet in gebruik

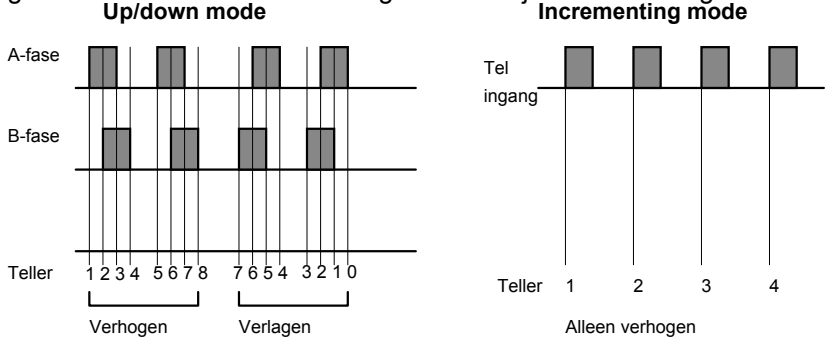
Veranderingen in de instelling in DM6642 zijn alleen effectief na het opnieuw starten van de uitvoer van het programma (uitzetten van de voedingsspanning of de PLC in program mode zetten).

**Signaal typen & tel modes**

Twee typen signalen kunnen worden ingelezen van een puls encoder. De tel mode die gebruikt wordt voor highspeed counter 0 is afhankelijk van het signaal type.

**Up/down mode**: Een phase-difference 4X twee fase signaal (A-fase en B-fase) en een Z-fase signaal worden gebruikt om te tellen. De getelde waarde wordt verhoogd of verlaagd afhankelijk van het fase verschil tussen de twee fase (A en B) signalen.

**Incrementing mode:** Één tel signaal (puls) en een teller reset signaal worden gebruikt. De teller wordt verhoogd afhankelijk van het tel signaal.




**Opmerking**

Één van de verderop beschreven methoden moet gebruikt worden om de counter te resetten wanneer deze herstart wordt. De counter wordt automatisch gereset wanneer de uitvoer van het programma wordt gestart of gestopt.

De volgende signaalovergangen worden gezien als voorwaartse (optellende) pulsen: A-fase opgaande flank gevolgd door B-fase opgaande flank gevolgd door A-fase neergaande flank gevolgd B-fase neergaande flank. De volgende signaal-overgangen worden gezien als teruggaande (verlagende) pulsen: B-fase opgaande flank gevolgd door A-fase opgaande flank gevolgd door B-fase neergaande flank gevolgd A-fase neergaande flank.

Het tel bereik loopt van -32767 tot 32767 voor de Up/Down Mode en van 0 tot 65535 voor de Incrementing Mode. Pulsen kunnen worden geteld met een frequentie tot 2.5 kHz in Up/Down Mode en tot 5.0 kHz in Incrementing Mode.

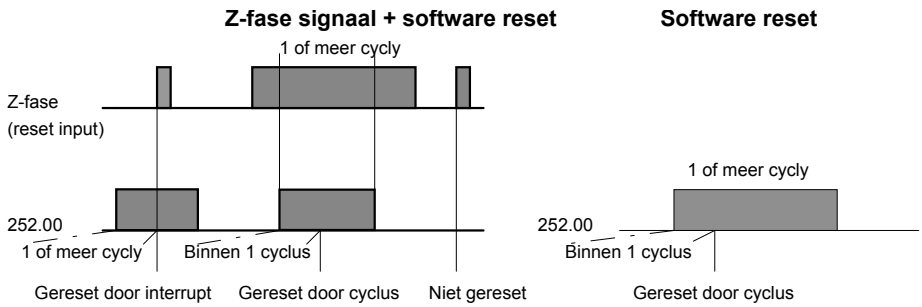
De Up/Down Mode gebruikt altijd een 4X phase-difference signaal. Het aantal getelde pulsen voor elke encoder omwenteling is 4 maal de resolutie van de encoder. Selecteer de encoder gebaseerd op de telbare bereiken.

**Reset methoden**

Één van de twee methoden hieronder beschreven moet worden gebruikt om de actuele waarde van de counter te resetten. (D.w.z. instellen op 0).

**Z-fase signaal + software reset:** De actuele waarde wordt gereset wanneer het Z-fase signaal (reset input) en het *Highspeed Counter 0 Resetbit* (252.00) beide aan zijn.

**Software reset:** De actuele waarde wordt gereset wanneer het *Highspeed Counter 0 Resetbit* (252.00) aan wordt gezet.




**Opmerking** De status van het *Highspeed Counter 0 Resetbit* (252.00) wordt één keer per cyclus, tijdens de I/O refresh, doorgegeven aan de highspeed counter. Om de counter betrouwbaar te resetten is het dus noodzakelijk om het minimaal één cyclus aan te houden.

De "Z" in "Z-fase" is een afkorting voor "Zero" (Engels voor nul). Het is een signaal dat de encoder één keer rond is gedraaid.

**Highspeed counter interrupt** Voor highspeed counter interrupts wordt een vergelijkingstabel gebruikt in plaats van het “Tel tot” principe. De teller controle kan worden uitgevoerd op één van de twee hieronder beschreven methoden. In de vergelijkingstabel worden de met de actuele waarde te vergelijken condities en interruptroutine combinaties opgeslagen.

**Doel waarde:** Een maximum van 16 vergelijkingscondities (waarde en de telrichting) en interruptroutine combinaties worden opgeslagen in de vergelijkingstabel. Wanneer de counter actuele waarde en de telrichting gelijk zijn aan de vergelijkingscondities, dan wordt de gespecificeerde interruptroutine uitgevoerd.
**Bereik vergelijken:** Acht vergelijkingscondities (hoge en lage limieten) en interruptroutine combinaties worden opgeslagen in de vergelijkingstabel. Wanneer de actuele waarde groter of gelijk is aan de lage limiet en kleiner of gelijk aan de hoge limiet dan wordt de gespecificeerde interruptroutine uitgevoerd.

**Bedrading** Hieronder worden de aansluitingen van de puls encoder op de input terminal van de CPU gegeven, afhankelijk van de telmode.

| Terminal nr. | Up/Down Mode | Incrementing Mode |
|---|---|---|
| 000.00 | Encoder A-fase | Puls count input |
| 000.01 | Encoder B-fase | --- |
| 000.02 | Encoder Z-fase | Reset input |

Wanneer alleen de software reset wordt gebruikt dan kan terminal 000.02 worden gebruikt als normale ingang. In de Incrementing Mode kan terminal 000.01 worden gebruikt als normale ingang.

**Programmeren** Gebruik de volgende stappen om de highspeed counter te programmeren.

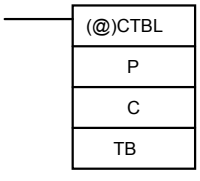
De highspeed counter begint met tellen nadat de juiste instellingen in de PC Setup zijn gemaakt. Vergelijkingen zullen echter niet worden uitgevoerd met de vergelijkingstabel en interrupts worden niet uitgevoerd tot de CTBL(—) instructie wordt uitgevoerd.

De highspeed counter wordt ingesteld op "0" als de voedingsspanning aan wordt gezet en wanneer de uitvoer van het programma wordt gestart en gestopt.

De huidige waarde van de highspeed counter kan worden uitgelezen op 248 en 249.

**Beheersen van highspeed counter interrupts**

1. Gebruik de CTBL (—) instructie om de vergelijkingstabel in de CPM1(A) op te slaan en om de vergelijkingen te starten.




P: Poort specificatie (0 = Highspeed counter 0)
C: (3 cijfers BCD)
000 : Doel tabel ingesteld en vergelijken gestart
001 : Bereik tabel ingesteld en vergelijken gestart
002 : Doel tabel alleen ingesteld
003 : Bereik tabel alleen ingesteld
TB : Begin woord van de vergelijkings tabel

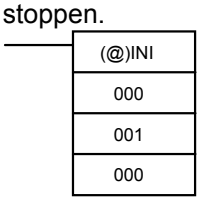
Als C is ingesteld op 000, dan worden de vergelijkingen uitgevoerd in de doel mode; Als C 001 is dan worden ze uitgevoerd in de bereik vergelijking mode. De vergelijkingstabel wordt door de instructie opgeslagen en wanneer deze bewerking uitgevoerd is, zal het vergelijken gestart worden. Als de

vergelijkingen worden uitgevoerd zullen highspeed interrupts worden uitgevoerd zoals dit in de vergelijkingstabel is opgeslagen. Zie voor meer details "Registreer vergelijkingstabel - CTBL(—)" op pagina 97.

**Opmerking** In de bereik vergelijking mode worden de resultaten van de vergelijking normaal opgeslagen in AR11.00 t/m AR11.07.

Als C is ingesteld op 002, dan worden de vergelijkingen uitgevoerd in de doel mode als C 003 is dan worden ze uitgevoerd in de bereik vergelijking mode. Bij beide instellingen wordt de vergelijkingstabel opgeslagen, maar de vergelijkingen zullen niet gestart worden. De INI(—) instructie moet in dit geval gebruikt worden om de vergelijkingen te starten.

2. Voer de INI (—) instructie uit zoals hieronder getoond om de vergelijkingen te stoppen.




Zet de tweede operand op “000” om de vergelijkingen opnieuw te starten en voer de INI(—) instructie uit.

Zodra een tabel is opgeslagen dan wordt deze in de CPM1(A) bewaard zolang de CPM1(A) het programma uitvoert en er geen andere tabel wordt opgeslagen.

**Uitlezen van de huidige waarde**

Er zijn twee manieren om de actuele waarde uit te lezen. De eerste is om deze direct uit 248 en 249 te lezen, de tweede is door gebruik te maken van de PRV(—) instructie.

**Uitlezen van 248 en 249**

De actuele waarde van de highspeed counter is opgeslagen in 248 en 249 zoals hieronder getoond. Het meest linkse cijfer van 249 zal “F” bevatten als het getal negatief is.

| Meest linker 4 cijfers | Meest rechter 4 cijfers |
|---|---|
| 249 | 248 |

Up/Down Mode : F0032767 (-32767) t/m 00032767

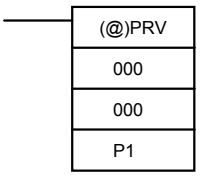
Incrementing mode: 00000000 t/m 00065535

**Opmerking** Deze woorden worden één keer per cyclus gerefreshed, tijdens de I/O refresh, Zodat er een verschil kan bestaan tussen de uitgelezen actuele waarde en de huidige actuele waarde.

Wanneer de highspeed counter niet wordt gebruikt kunnen deze woorden (bits) worden gebruikt als werkbits.

**De PRV(—) instructie gebruiken**

Lees de actuele waarde van de highspeed counter uit door de PRV(—) instructie te gebruiken.




P1: Eerste woord van de actuele waarde

De actuele waarde van de highspeed counter is opgeslagen zoals hieronder getoond. Het meest linkse cijfer zal “F” bevatten als het getal negatief is.

| Meest linker 4 cijfers | Meest rechter 4 cijfers |
|---|---|
| P1+1 | P1 |

Up/Down Mode : F0032767 (-32767) t/m 00032767

Incrementing mode: 00000000 t/m 00065535

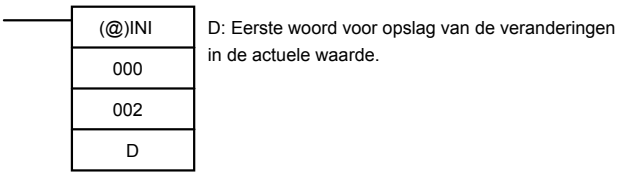
De actuele waarde wordt uitgelezen op het moment waarop de PRV(—) instructie wordt uitgevoerd.

**Actuele waarde veranderen**

Er zijn twee manieren om de actuele waarde van de highspeed counter te veranderen. De eerste manier is door deze te resetten door één van de reset methodes te gebruiken. In dit geval wordt de huidige waarde ingesteld op 0. De tweede manier is door gebruik te maken van de INI(—) instructie.

De methode die de INI(—) instructie gebruikt wordt hier uitgelegd. Refereer naar het begin van deze verklaring voor een verklaring van de reset methodes voor highspeed counter 0.

Verander de huidige waarde van de counter door de INI(—) instructie zoals hieronder getoond te gebruiken.




| Meest linker 4 cijfers | Meest rechter 4 cijfers |
|---|---|
| D+1 | D |

Up/Down Mode : F0032767 (-32767) t/m 00032767

Incrementing mode: 00000000 t/m 00065535

Zet op het meest linker cijfer van D+1 een F om een negatief getal in te kunnen stellen.

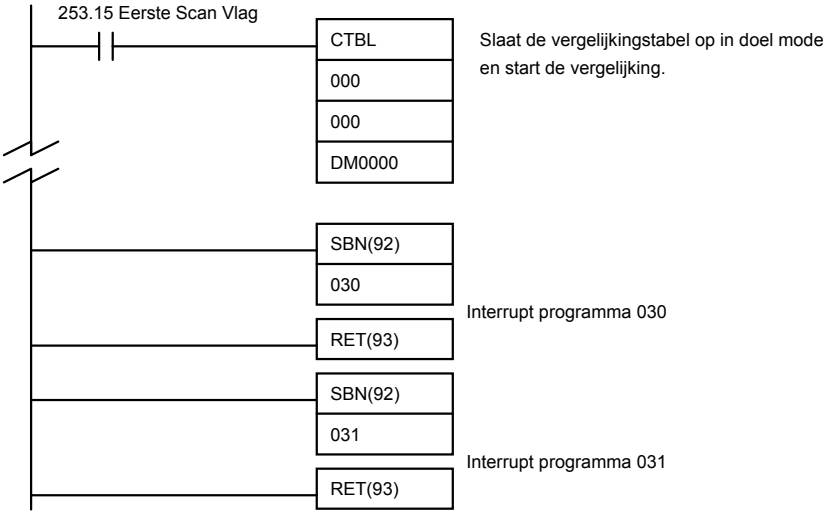
**Voorbeeld**

Dit voorbeeld toont een programma waarin de highspeed counter in de Incrementing Mode wordt gebruikt. Vergelijkingen worden uitgevoerd in de doel mode en afhankelijk van de huidige waarde van de counter zullen de verschillende subroutines uitgevoerd worden. Stel de PC Setup als volgt in voor het uitvoeren van het programma:

DM6642: 0114 (Highspeed counter gebruikt met software reset en in de Incrementing Mode). Gebruik voor alle ander PC Setup instellingen de default waarden.

Als aanvulling moet de volgende data opgeslagen worden voor de vergelijkingstabel:

| | | |
|---|---|---|
| DM0000 | 0002 | Aantal vergelijkingscondities: 2 |
| DM0001 | 1000 | Doel waarde 1:1000 |
| DM0002 | 0000 | |
| DM0003 | 0030 | Vergelijking 1 interruptroutine nr.: 030 |
| DM0004 | 2000 | Doel waarde 1:2000 |
| DM0005 | 0000 | |
| DM0006 | 0031 | Vergelijking 2 interruptroutine nr.: 031 |




## *2.3.6 Highspeed counter overflows / underflows*

Als het toegestane telbereik voor de highspeed counter wordt onder- of overschreden en een underflow of overflow status zal optreden zal de actuele waarde van de counter blijven staan op 0FFF FFFF voor overflows en FFFF FFFF voor underflows tot de overflow/underflow status wordt gewist door de counter te resetten. De toegestane telbereiken zijn als volgt:

Up/Down Mode: F003 2767 t/m 0003 2767
Incrementing Mode: 0000 0000 t/m 0006 5535

**1, 2, 3...**

1. De waarden die hierboven gegeven worden zijn theoretisch waarbij wordt aangenomen dat de PLC een aannemelijk korte cyclus tijd heeft. Actueel zullen waarden voorkomen die één cyclus voor de overflow/underflow error voorkwam aanwezig waren.
2. Het zesde en zevende cijfer van de highspeed counters actuele waarde zijn normaal gesproken 00, maar kunnen gebruikt worden als “Overflow/Underflow vlaggen" door te detecteren of deze bits uit of aan zijn. Hiermee kan gedetecteerd worden of waarden buiten de toegestane bereiken zijn gekomen.

De highspeed counter kan gereset worden op één van de hiervoor besproken methoden of automatisch door de programma uitvoer te herstarten. De highspeed counter en de gerelateerde functies zullen niet normaal functioneren tot de underflow/overflow status is gewist. De werking tijdens een overflow/underflow status is als volgt:

- ⇨ Werking van de vergelijkingstabel stopt.
- ⇨ De vergelijkingstabel wordt niet gewist.
- ⇨ Interruptroutines van de highspeed counter worden niet uitgevoerd.
- ⇨ CTBL(—) kan alleen gebruikt worden om een nieuwe vergelijkingstabel te registreren. Als een poging wordt gedaan om de werking van de vergelijkingstabel te starten dan zal deze niet gestart en ook niet geregistreerd worden.
- ⇨ INI(—) kan niet gebruikt worden om de werking van de vergelijkingstabel te starten of te stoppen of om de actuele waarde aan te passen.
- ⇨ PRV(—) zal alleen 0FFF FFFF of FFFF FFFF uitlezen als actuele waarde.

**Herstellen**

Gebruik de volgende procedure om een overflow/underflow status te herstellen.

**Met een geregistreerde vergelijkingstabel**

1. Reset de counter.
2. Stel indien nodig de actuele waarde in met PRV(—).
3. Stel indien nodig de vergelijkingstabel in met CTBL(—).
4. Start de vergelijkingstabel met INI(—).

**Zonder geregistreerde vergelijkingstabel**

1. Reset de counter.
2. Stel indien nodig de actuele waarde in met PRV(—).
3. Stel de vergelijkingstabel in en start de werking van deze met CTBL(—) en INI(—).

**Opmerking**

De bereik vergelijkingsvlaggen in AR11 blijven hun waarde handhaven na het herstellen. De interruptroutine voor een interruptconditie die direct na het herstel ontstaat zal niet worden uitgevoerd als dezelfde interruptconditie al was ontstaan voor de overflow/underflow status optrad. Als de uitvoer van de interruptroutine noodzakelijk is moet AR11 gewist worden voor u de overflow/underflow status hersteld.

**Werking reset**

Wanneer de highspeed counter wordt gereset, dan wordt de actuele waarde ingesteld op 0. Het tellen zal beginnen van 0 en de vergelijkingstabel, uitvoeringsstatus en uitvoeringsresultaten zullen hun status/waarde behouden.

**Opstart counter status**

Wanneer de highspeed counter wordt gestart, dan wordt de counter mode in de PC Setup gelezen en de actuele waarde ingesteld op 0, underflow/overflow statussen worden gewist, de geregistreerde vergelijkingstabel en de executie status ervan wordt gewist (Bereik vergelijkingsresultaten worden altijd gewist wanneer de werking wordt begonnen of wanneer de vergelijkingstabel wordt geregistreerd).

**Gestopte counter status**

Wanneer de highspeed counter wordt gestopt wordt de actuele waarde vastgehouden, de vergelijkingstabel en de executie status wordt gewist en de bereik vergelijkingsresultaten worden vastgehouden.

## 2.4 CPM1(A) communicatie functies

De volgende typen communicatie kunnen worden uitgevoerd door periferie poort van de CPM1(A).

Hostlink communicatie met een computer of terminal.

One-to-one link communicatie met een andere CPM1(A), CQM1 of C200HS/E/G/X PLC.
NT-link naar een OMRON NT terminal. Deze communicatievorm naar OMRON terminals werkt aanzienlijk sneller dan Hostlink.

**Opmerking** Met de CPM1(A)-CIF01 of CPM1(A)-CIF11 kan de periferie poort worden omgezet naar RS232C of RS422.

Deze sectie verklaart de benodigde instellingen in de PC Setup en methodes om deze communicatie te gebruiken.

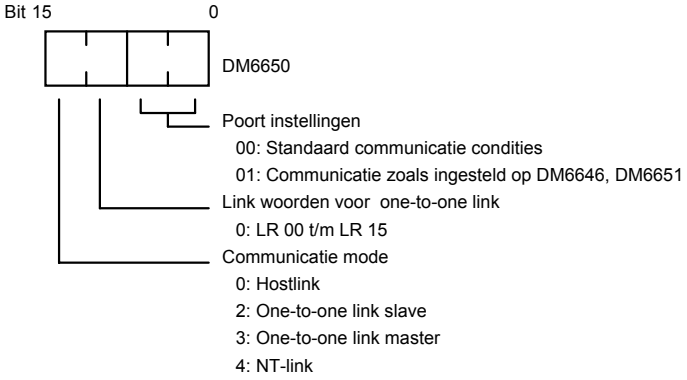
### *2.4.1 Communicatie PC Setup*

De PC Setup parameters in DM6650 t/m DM6654 worden gebruikt voor instellingen voor de communicatie poort.

**Opmerking** Als de dipswitch op de CPM1(A)-CIF01 op host wordt gezet dan worden de PC Setup communicatie parameters genegeerd en worden de onderstaande instellingen gebruikt voor de periferie poort.

| | |
|---|---|
| Mode: | Hostlink |
| Node nummer: | 00 |
| Start bits: | 1 bit |
| Data lengte: | 7 bits |
| Stop bits: | 2 bits |
| Pariteit: | Even |
| Baudrate: | 9.600 bps (2.400bps & 1 stopbit bij CPU's van voor juli '95) |
| Transmission delay: | Geen |

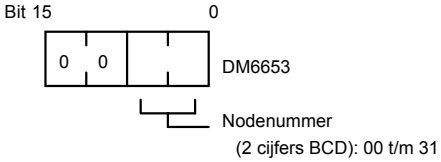
De instellingen in DM6650 bepalen de belangrijkste communicatie parameters, zoals in het volgende diagram getoond wordt.




**Default:** Hostlink met standaard parameters

**One-to-one link** Om een 1:1 link te gebruiken zijn de enige instellingen de communicatiemode en de linkwoorden. Stel de communicatiemode voor één van de PLC's in op de 1:1 master en de andere op de 1:1 slave. Stel vervolgens de linkwoorden in bij de PLC die ingesteld is als master. De instelling op de bits 08 t/m 11 is alleen geldig bij de master van de 1:1 link. (Let op!, 1:1 link is alleen te gebruiken in combinatie met de CPM1(A)-CIF01).

**Hostlink nodenummer** Een nodenummer moet ingesteld worden voor Hostlink communicatie om het verschil aan te kunnen geven tussen nodes wanneer er meer dan 1 node in de communicatie is opgenomen. Deze instelling is alleen nodig voor Hostlink communicatie. Om Hostlink communicatie te kunnen gebruiken moet Hostlink worden ingesteld als communicatiemode en moeten de communicatie parameters worden ingesteld.




**Default:** 00

Stel het nodenummer in op 00, tenzij er meerdere nodes in één netwerk zijn opgenomen.

### *2.4.2 Hostlink communicatie*

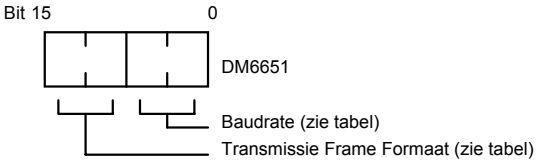
Selecteer Hostlink communicatie en stel vervolgens de communicatie parameters in zoals hieronder beschreven. Zorg ervoor dat de communicatie instellingen van de CPM1(A) exact overeen komen met die van het apparaat waarmee gecommuniceerd moet worden.

**Standaard communicatie**

Als de volgende instellingen correct zijn voor de communicatie zet dan de twee rechter cijfers van DM6650 op 00. Door deze instelling worden de ingevoerde parameters op DM6651 genegeerd.

| | |
|---|---|
| Start bits: | 1 bit |
| Data lengte: | 7 bits |
| Stop bits: | 2 bits |
| Pariteit: | Even |
| Baudrate: | 9.600 bps |

**Communicatie instellen**




**Default:** Standaard communicatie condities.

**Transmissie frame formaat**

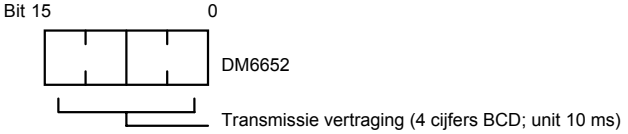
| Instelling | Start bit | Data lengte | Stop bits | Pariteit |
|---|---|---|---|---|
| 00 | 1 | 7 | 1 | Even |
| 01 | 1 | 7 | 1 | Oneven |
| 02 | 1 | 7 | 1 | None |
| 03 | 1 | 7 | 2 | Even |
| 04 | 1 | 7 | 2 | Oneven |
| 05 | 1 | 7 | 2 | None |
| 06 | 1 | 8 | 1 | Even |
| 07 | 1 | 8 | 1 | Oneven |
| 08 | 1 | 8 | 1 | None |
| 09 | 1 | 8 | 2 | Even |
| 10 | 1 | 8 | 2 | Oneven |
| 11 | 1 | 8 | 2 | None |

**Baudrate**

| Instelling | Baudrate |
|---|---|
| 00 | 1.200 bps |
| 01 | 2.400 bps |
| 02 | 4.800 bps |
| 03 | 9.600 bps |
| 04 | 19.200 bps |

**Transmissie vertraging**

Afhankelijk van het apparaat dat aangesloten is op de periferie poort, kan het nodig zijn om een transmissie vertraging in te stellen. Wanneer dit het geval is, stel dan de transmissie vertraging in om de toegestane tijd te regelen. De transmissie vertraging is de tijd die de PLC zal wachten met het versturen van een volgend byte nadat er een is verzonden.




**Default:** Geen vertraging

**Hostlink communicatie**

Hostlink communicatie is ontwikkeld door OMRON om één of meer PLC's met een computer te kunnen laten communiceren via een RS-232C kabel waarbij de computer data uit de PLC kan lezen of ernaar kan schrijven. Normaal stuurt de computer een commando naar de PLC en stuurt de PLC automatisch een

response terug. De communicatie wordt dus uitgevoerd zonder dat de PLC er actief bij is betrokken. De PLC kan ook communicatie opbouwen naar de computer indien dit noodzakelijk is. In dit geval wordt gesproken over event driven communicatie.

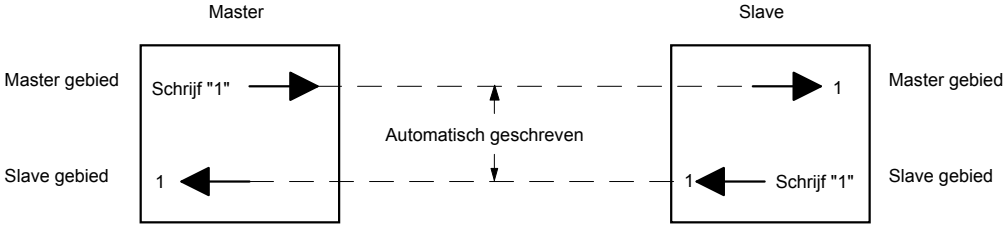
Over het algemeen zijn er twee manieren om Hostlink communicatie te implementeren. De ene manier is gebaseerd op C-mode commando's en de andere op FINS (CV-mode) commando's. De CPM1(A) ondersteunt alleen de C-mode commando's.

### *2.4.3 One-to-one link communicatie*

Als twee CPM1(A)'s op 1:1 basis worden gekoppeld via de periferie poort, dan kunnen ze het LR geheugen gezamenlijk delen. Wanneer twee PLC's one-to-one worden gekoppeld, dan zal de een als master en de ander als slave fungeren.

**Opmerking** Bij de CQM1 en C200HS/E/G/X kan de periferie poort niet worden gebruikt voor een 1:1 link. Bij deze CPU's dient de RS232C poort gebruikt te worden. Bij de CPM1(A) kan een 1:1 link alleen gebruikt worden in combinatie met een CPM1(A)-CIF01.

**One-to-one link** Een one-to-one link staat twee PLC's toe om data gemeenschappelijk te delen in het LR gebied. Zoals in het onderstaande diagram getoond, zal data die geschreven wordt in een woord in het LR gebied van de ene PLC automatisch naar hetzelfde woord in het LR gebied van de andere PLC worden geschreven. Elke PLC krijgt een aantal vaste woorden toegewezen waarop geschreven kan worden en een aantal vaste woorden waar de andere PLC op mag schrijven.




De woorden die door de PLC's gebruikt worden staan in de onderstaande tabel. Deze zijn afhankelijk van de in de PC Setup ingestelde link woorden voor de master.

| DM6645 instelling | LR00 t/m LR63 | LR00 t/m LR31 | LR00 t/m LR15 |
|---|---|---|---|
| Master woorden | LR00 t/m LR31 | LR00 t/m LR15 | LR00 t/m LR07 |
| Slave woorden | LR32 t/m LR63 | LR16 t/m LR31 | LR08 t/m LR15 |

**Opmerking** De CPM1(A) ondersteund alleen de optie LR00 t/m LR15 voor de grootte van het LR gebied. Word een CPM1(A) gebruikt in combinatie met een CQM1 of C200HS/E/X/G, dan moet bij deze PLC's dus ook voor deze instelling worden gekozen.

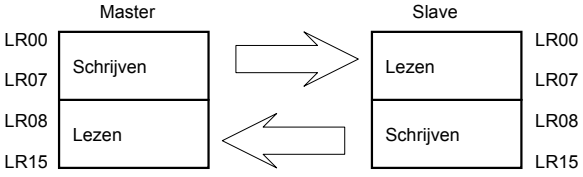
**Communicatie procedure** Als de instellingen voor de master en de slave correct zijn gemaakt, dan wordt de one-to-one link automatisch gestart door simpelweg de spanning op beide PLC's te zetten. De 1:1 link werkt onafhankelijk van de mode van de PLC.

**Voorbeeld** Dit voorbeeld toont een programma dat de statussen van de ene CPU doorgeeft aan de andere. Dit gebeurt door middel van een one-to-one link die ingesteld staat op de RS-232C poort. Voor het programma uitgevoerd kan worden moeten de volgende instellingen in de PC Setup gemaakt worden.

Master: DM6645: 3000 (one-to-one link master; LR 00 t/m LR 15 wordt gebruikt)

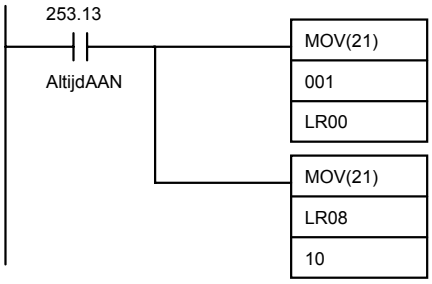
Slave: DM6645: 2000 (one-to-one link slave)

Voor de rest van de parameters in de PC Setup wordt aangenomen dat ze op de standaard waarde staan ingesteld. De woorden die voor de one-to-one link worden gebruikt zijn hieronder getoond.

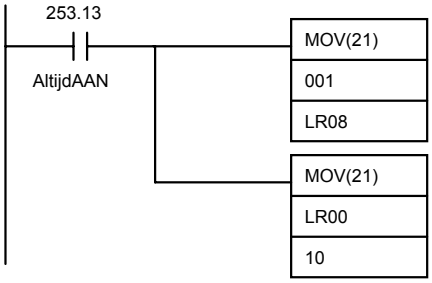


Wanneer het programma in zowel de master als de slave wordt uitgevoerd, zal de status van woord 001 van elke unit op woord 10 van de andere unit getoond worden. Woord 001 is een ingangswoord en 10 is een uitgangswoord.

**Master**



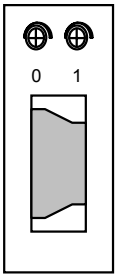

**Slave**




## 2.5 Analoge instellingen

In de CPM1(A) verplaatst de analoge instellingen functie automatisch de instellingen van de CPU's potentiometers naar woord 250 en 251. Deze functie is handig wanneer ingestelde waarden in het programma voorkomen die tijdens bedrijf veranderd moeten worden. Deze ingestelde waarden kunnen aangepast worden door aan de potentiometers op de CPU te draaien.

De volgende figuur geeft de IR woorden aan die bij de potentiometers horen. De potentiometers zijn ondergebracht onder hetzelfde luikje op de CPU waar zich de periferiepoort bevindt.




Het adres voor schakelaar 0 is woord 250
Het adres voor schakelaar 1 is woord 251

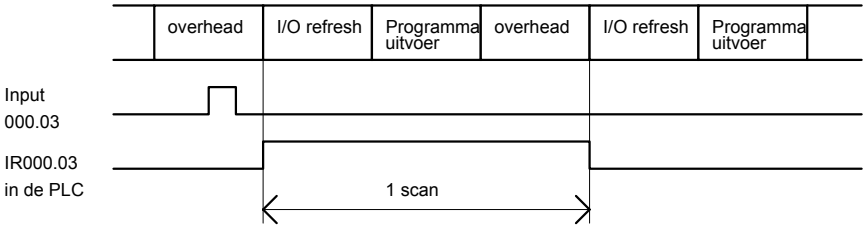
De instellingen van de potentiometers worden opgeslagen in BCD en kunnen variëren van 0000 t/m 0200. Gebruik een precisieschroevendraaier om de instellingen aan te passen. Draai de schakelaars met de klok mee om de waarde te verhogen.

**Opmerking** De CPM1(A) vernieuwt zolang de CPU aan staat de woorden 250 t/m 251 voortdurend met de waarde die ingesteld is op de potentiometers. Overschrijf deze woorden niet met het programma of periferie.

## 2.6 Quick response ingangen

De CPM1(A)-10CDR PLC’s beschikken over twee quick response ingangen en de CPM1(A)-20CDR en 30CDR over vier quick response ingangen. Voor de quick response ingangen en de interrupt ingangen worden dezelfde PLC ingangen gebruikt.

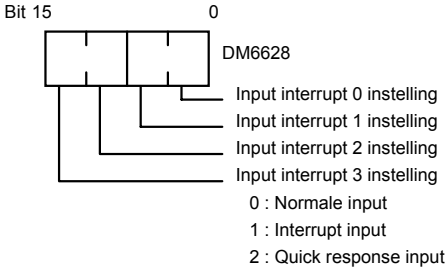
Quick response ingangen gebruiken een interne buffer, zodat ingangssignalen korter dan één cyclus gedetecteerd kunnen worden. Signalen met een pulsbreedte tot 0,2ms kunnen met deze functie gedetecteerd worden, onafhankelijk van hun timing ten opzichte van de cyclus van de PLC.




| PLC model | Input bits | Min input puls breedte |
|---|---|---|
| CPM1(A)-10CDR | 000.03 & 000.04 | 0,2 ms |
| CPM1(A)-20CDR/30CDR | 000.03 t/m 000.06 | |

**Instellen**

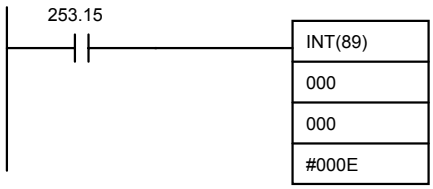
De ingangen 000.03 t/m 000.06 (000.03 & 000.04 voor CPM1(A)-10CDR) kunnen worden ingesteld als quick response ingangen op DM6628 zoals in de volgende figuur getoond is.




**Default:** Alle ingangen normaal

**Programma voorbeeld**

Het onderstaande programma kan gebruikt worden om ingang 000.03 als quick response ingang te gebruiken. DM6628 moet worden ingesteld op 0002, de rest van de PC Setup wordt ingesteld op de default waarde.



# 3 Geheugengebieden

Verschillende typen data zijn nodig om een besturing effectief en correct te kunnen laten werken. Om met deze verschillende typen data overweg te kunnen is de PLC voorzien van verschillende geheugengebieden voor data opslag, waarin elk gebied een andere functie heeft. De gebieden die gebruikt worden in het programma worden datagebieden genoemd. Het andere geheugengebied is het programmageheugen, waar het programma zelf in wordt opgeslagen. Deze gebieden worden hier individueel besproken waarbij alle informatie die nodig is om het te kunnen gebruiken wordt gegeven. Daarom wordt het SR en TR geheugen hier ook besproken, alhoewel dit strikt gezien geen geheugengebieden zijn.

## 3.1 Introductie

Details waaronder de naam, het acroniem en de functie van elke gebied worden in de volgende tabel samengevat. Op de laatste twee na zijn dit datagebieden. Data en geheugengebieden worden normaal gesproken benoemd met hun acroniem.

| Gebied | Acroniem | Functie |
|---|---|---|
| Interne relais | IR | Gebruikt om I/O punten aan te sturen en om tijdelijke data in op te slaan. |
| Speciale relais | SR | Bevat systeem klokpulsen, vlaggen, bits met speciale functies en status informatie. |
| Auxiliary relais | AR | Bevat vlaggen en bits voor speciale functies. Dit gebied houdt de status vast na spanningsuitval. |
| Datageheugen | DM | Gebruikt voor interne dataopslag en manipulatie. Dit gebied houdt de status vast na spanningsuitval. |
| Holding relais | HR | Gebruikt om data (bits) in op te slaan waarvan de status vastgehouden moet worden als de spanning van de PLC wordt afgezet. |
| Timer/Counter | TIM / CNT (TC) | Gebruikt om timers en counters te definiëren en de timer- en countervlaggen te bereiken. Wanneer dit gebied wordt uitgelezen of bewerkt met instructies op woordniveau wordt de actuele waarde van een timer of counter uitgelezen. Wordt het gebruikt als bitoperand dan wordt de completionvlag van een timer of counter uitgelezen. De completionvlag is hoog als een timer of counter is afgelopen. |
| Link relais | LR | Beschikbaar als werkbits. Bij diverse soorten OMRON netwerken wordt dit gebied gebruikt als gemeenschappelijk geheugen. |
| Temporary relais | TR | Gebruikt om tijdelijk executieconditities in op te slaan. Deze bits kunnen alleen met de LOAD (LD) en Output (OUT) instructies aangestuurd worden. Bij het programmeren in een ladderdiagram mogen ze niet gebruikt worden. Het opslaan van de status van een executieconditie is nodig tijdens het programmeren van bepaalde aftakkingen in ladderdiagrammen. |
| Programma geheugen | UM | Bevat het programma dat wordt uitgevoerd door de CPU. |

**Werkbits en woorden**

Wanneer sommige bits en woorden in bepaalde datagebieden niet gebruikt worden voor hun eigenlijke doel, kunnen ze vrij in het programma gebruikt worden om statussen in op te slaan. Woorden en bits die op deze manier gebruikt worden, worden werkwoorden en werkbits (helpwoorden, helpbits) genoemd. Bijna alle ongebruikte bits kunnen worden gebruikt als werkbits. Deze bits worden verderop in dit hoofdstuk gebied voor gebied uitgelegd. Het daadwerkelijk toepassen van werkbits en werkwoorden wordt uitgelegd in de sectie “Programma uitvoer” op pagina 30.

**Vlaggen en controlebits**

Sommige datagebieden bevatten vlaggen en/of controlebits. Vlaggen zijn bits die automatisch aan- en uitgezet worden om een bepaalde status van de PLC of instructie weer te geven. Alhoewel sommige vlaggen door de gebruiker aan- en uitgezet kunnen worden, zijn de meeste vlaggen read only. De gebruiker mag alleen de status van de vlag uitlezen, deze vlaggen zijn niet direct aan te sturen.

Controlebits zijn bits die door de gebruiker aan- en uitgezet kunnen worden om bepaalde aspecten binnen de werking van de PLC of een instructie te beïnvloeden. Elk bit dat een naam heeft gekregen, met de naam bit in plaats van vlag, is een controlebit. Bijvoorbeeld restartbits zijn controlebits.

# 3.2 Geheugengebieden voor de CPM1(A)

Deze sectie beschrijft de structuur van de CPM1(A)'s geheugengebieden en verklaart hoe deze te gebruiken. Een overzicht van de verschillende geheugengebieden voor de CPM1(A) kan in de sectie hierover worden gevonden

## *3.2.1 Geheugengebied functie*

**Geheugengebied structuur** De volgende geheugengebieden kunnen gebruikt worden bij de CPM1(A).

| Datagebied | | Grootte | Woorden | Bits | Functie |
|---|---|---|---|---|---|
| IR gebied[1] | Input gebied | 160 | 000 t/m 009 | 000.00 t/m 009.15 | Deze bits kunnen worden toegewezen aan externe I/O. |
| | Output gebied | | 010 t/m 019 | 010.00 t/m 019.15 | |
| | Werk gebieden | 512 bits min. | 200 t/m 231 | 200.00 t/m 231.15 | Werkbits hebben geen speciale functie en kunnen vrij gebruikt worden in het programma |
| SR gebied | | 384 bits | 232 t/m 255 | 232.00 t/m 255.07 | Deze bits hebben specifieke functies. Het zijn een aantal vlaggen en control bits |
| TR gebied | | 8 bits | --- | TR0 t/m TR7 | Deze bits worden gebruikt om tijdelijke statussen van programma vertakkingen in op te slaan |
| HR gebied[2] | | 320 bits | HR00 t/m HR19 | HR00.00 t/m HR19.15 | Deze bits kunnen gebruikt worden om bits op te slaan waarvan de status behouden moet worden als de spanning van de CPU wordt afgezet |
| AR gebied[2] | | 256 bits | AR00 t/m AR15 | AR00.00 t/m AR15.15 | Deze bits hebben specifieke functies. Het zijn een aantal vlaggen en control bits |
| LR gebied[1] | | 256 bits | LR00 t/m LR16 | LR00.00 t/m LR16.15 | Gebruikt door de 1:1 datalink met de periferie poort |
| Timer/Counter gebied[3] | | 128 bits | TC000 t/m TC127 (timer/counter nummers) | | Dezelfde nummers worden gebruikt voor timers en voor counters. TC000 wordt gebruikt door de interval timer. |
| DM gebied[2] | Read/Write[2] | 1.000 woorden | DM0000 t/m DM0999, | --- | DM gebied data kan alleen worden geadresseerd in woord georiënteerde instructies. De hier opgeslagen waarden behouden hun waarde bij spanningsuitval. |
| | Read-only[4] | 456 woorden | DM6144 t/m DM6599 | --- | Kan niet overschreven worden door het programma |
| | Error historie gebied[4] | 22 woorden | DM1000 t/m DM1022 | --- | Gebruikt om de tijd van voorkomen en de errorcode weg te zetten van errors die zijn voorgekomen |
| | PC Setup[4] | 56 woorden | DM6600 t/m DM6655 | --- | Gebruikt om diverse instellingen van de PLC in op te slaan |
| Gebruiker programma (user program) gebied (UM gebied) | | 2.048 woorden | ---- | | Gebruikt om het programma in op te slaan. Gebufferd bij spanningsuitval. |

**Noot**

1. IR en LR bits die niet gebruikt worden voor hun toegewezen functie kunnen gebruikt worden als werkbits.
2. De inhoud van het HR, AR, CNT en read/write DM geheugen wordt door een condensator van back-up spanning voorzien. Bij 25ºC zal deze condensator het geheugen voor 20 dagen vasthouden. Raadpleeg de installatie handleiding van de CPM1(A) voor informatie over het effect van de omgevingstemperatuur op de back-up tijd van de condensator.
3. Wanneer een actuele waarde geadresseerd moet worden, wordt het TC nummer gebruikt als woord data, wanneer completionvlaggen geadresseerd moeten worden wordt het nummer gebruikt als bit data.
4. Data in DM1000 t/m DM1022 & DM6144 t/m DM6655 kan niet worden overschreven door het PLC programma. Het is echter wel mogelijk om deze data met SYSWIN of een handprogrammeerapparaat te overschrijven.

**Opmerking** De woorden 020 t/m 199 uit het IR gebied zijn bij de CPM1(A) niet aanwezig. Deze kunnen dus ook niet gebruikt worden in het programma.

Een uitgebreidere uitleg van de verschillende geheugengebieden volg verderop in dit hoofdstuk.

## 3.3 Toewijzen van I/O bits

De volgende tabel toont de bits in het IR geheugen die aan de I/O van een CPM1 CPU en uitbreidingsunit worden toegewezen. Op een CPM1 CPU kan 1 uitbreiding worden aangesloten. Dit kan een CPM1 uitbreidingsunit zijn, maar ook een CPM1A uitbreidingsunit mag op een CPM1 worden aangesloten. Alle CPM1 CPU's zijn uit te breiden

| Aantal I/O op de CPU | | 10 | | 20 | | 30 | | 40 | |
|---|---|---|---|---|---|---|---|---|---|
| **CPU aansluitingen** | **Ingangen** | 6pts<br>000.00 t/m 000.05 | | 12pts<br>000.00 t/m 000.11 | | 18pts<br>000.00 t/m 000.11<br>001.00 t/m 001.05 | | --- | |
| | **Uitgangen** | 4pts<br>010.00 t/m 010.03 | | 8pts<br>010.00 t/m 010.07 | | 12pts<br>010.00 t/m 010.07<br>011.00 t/m 011.03 | | --- | |
| **Uitbreidings-unit** | **Ingangen** | 12pts<br>001.00 t/m 001.11 | | 12pts<br>001.00 t/m 001.11 | | 12pts<br>002.00 t/m 002.11 | | --- | |
| | **Uitgangen** | 8pts<br>011.00 t/m 011.07 | | 8pts<br>011.00 t/m 011.07 | | 8pts<br>012.00 t/m 012.07 | | --- | |
| **Voedingsspanning** | | AC | DC | AC | DC | AC | DC | --- | --- |
| **Typenummer** | **Relais uitgangen** | CPM1-10CDR-A | CPM1-10CDR-D | CPM1-20CDR-A | CPM1-20CDR-D | CPM1-30CDR-A | CPM1-30CDR-D | --- | --- |

De volgende tabel toont de bits in het IR geheugen die aan de I/O van een CPM1A CPU en uitbreidingsunit worden toegewezen. Op een CPM1 CPU kan 1 uitbreiding worden aangesloten. Dit kan een CPM1 uitbreidingsunit zijn, maar ook een CPM1A uitbreidingsunit mag op een CPM1 worden aangesloten. Alleen de CPM1A CPU's met 30 en 40 I/O zijn uit te breiden.

| Aantal I/O op de CPU | | 10 | 20 | 30 | 40 |
|---|---|---|---|---|---|
| **CPU aansluitingen** | **Ingangen** | 6pts<br>000.00 t/m 000.05 | 12pts<br>000.00 t/m 000.11 | 18pts<br>000.00 t/m 000.11<br>001.00 t/m 001.05 | 24pts<br>000.00 t/m 000.11<br>001.00 t/m 001.11 |
| | **Uitgangen** | 4pts<br>010.00 t/m 010.03 | 8pts<br>010.00 t/m 010.07 | 12pts<br>010.00 t/m 010.07<br>011.00 t/m 011.03 | 16pts<br>010.00 t/m 010.07<br>011.00 t/m 011.07 |
| **Uitbreidings-unit** | **Ingangen** | --- | --- | 12pts<br>002.00 t/m 002.11 | 12pts<br>002.00 t/m 002.11 |
| | **Uitgangen** | --- | --- | 8pts<br>012.00 t/m 012.07 | 8pts<br>012.00 t/m 012.07 |
| | **Ingangen** | --- | --- | 12pts<br>003.00 t/m 003.11 | 12pts<br>003.00 t/m 003.11 |
| | **Uitgangen** | --- | --- | 8pts<br>013.00 t/m 013.07 | 8pts<br>013.00 t/m 013.07 |
| | **Ingangen** | --- | --- | 12pts<br>004.00 t/m 004.11 | 12pts<br>004.00 t/m 004.11 |
| | **Uitgangen** | --- | --- | 8pts<br>014.00 t/m 014.07 | 8pts<br>014.00 t/m 014.07 |
| **Voedingsspanning** | | DC | DC | DC | DC |
| **Typenummer** | **PNP uit** | CPM1A-10CDT1-D | CPM1A-20CDT1-D | CPM1A-30CDT1-D | CPM1A-30CDT1-D |
| | **NPN uit** | CPM1A-10CDT-D | CPM1A-20CDT-D | CPM1A-30CDT-D | CPM1A-30CDT-D |

## 3.4 Datagebied structuur

Wanneer een datagebied benoemd wordt, moet het acroniem voor dat gebied altijd ingevoerd worden, behalve voor het IR en SR gebied. Alhoewel de acroniemen voor IR en SR in deze tekst soms genoemd worden, zijn ze niet nodig tijdens invoer van het programma. SYSWIN zal zelfs een foutmelding geven wanneer de acroniemen SR en IR worden ingevoerd. Van elk datagebied dat in deze tekst genoemd wordt zonder acroniem kunt u aannemen dat het in het SR of IR gebied ligt. Omdat IR en SR adressen opeenvolgend in het geheugen geplaatst zijn, zijn de woord en bitadressen voldoende om het onderscheid tussen deze twee gebieden te maken.

De actuele locatie van data binnen een gebied (behalve voor het TC gebied) wordt bepaald door het adres. Het adres bepaalt het woord en/of bit in een gebied waar de data geplaatst is. Het TC gebied bestaat uit TC nummers, waarvan elk wordt

gebruikt voor een specifieke timer of counter in het programma. Raadpleeg het hoofdstuk "TC (timer/counter) gebied" op pagina 69 voor meer details over TC nummers en hoofdstuk "Timer en counter instructies" op pagina 86 voor informatie over de werking van timers en counters. Alhoewel voor het timer/counter gebied de acroniem TC wordt gebruikt wordt tijdens het programmeren TIM voor timers en CNT voor counters gebruikt.

De overige datagebieden (d.w.z. de IR, SR, HR, DM, AR en LR gebieden) bestaan uit woorden, waarbij elk woord bestaat uit 16 bits, die genummerd zijn van 00 t/m 15 van rechts naar links. De IR woorden 000 en 001 zijn hieronder getoond met bitnummers. De inhoud van elk woord is getoond met alleen nullen. Bit 00 wordt het meest rechter bit en 15 het meest linker bit genoemd.

De term minst significant wordt ook vaak gebruikt voor het meest rechter bit en de term meest significant voor het meest linker bit. Deze termen worden in deze handleiding niet vaak gebruikt, omdat een enkel data woord vaak wordt gesplitst in twee of meer delen, waarbij elk deel wordt gebruikt voor verschillende parameters of operands. Hierbij kan bij gebruik van de termen “meest” en “minst” significant verwarring ontstaan.

| Bit nummer | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IR woord 000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IR woord 001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Het DM gebied is alleen op woordniveau te adresseren. Het is niet direct mogelijk om een individueel bit in een DM woord te adresseren. Data in de IR, SR, HR, AR en LR gebieden is bereikbaar op zowel woord- als bitniveau, afhankelijk van de instructie die voor het verwerken van de data gebruikt wordt.

Om een woord uit een van deze gebieden te benoemen, is alles wat nodig is het acroniem (indien nodig) en het adres van het woord. Om een bit uit een gebied te benoemen moet het acroniem ingevoerd worden gevolgd door het adres van het woord en het adres van het bit in dat woord. In SYSWIN worden woord- en bitadres van elkaar gescheiden door een punt (.). Wordt de punt in SYSWIN niet ingevoerd dan wordt aangenomen dat de laatst twee ingevoerde cijfers het bitadres in het woord zijn. Het nummer van een bit mag tussen de 00 en 15 liggen en wordt decimaal ingevoerd.

Hetzelfde TC nummer kan gebruikt worden om zowel de actuele waarde van de timer of counter te adresseren of het bit dat functioneert als de completionvlag voor de timer of counter. Dit wordt in meer detail beschreven in hoofdstuk "TC (timer/counter) gebied" op pagina 69.

| Gebied | Woord toewijzing | Bit toewijzing |
|---|---|---|
| IR | 000 | 000.15 (meest linker bit in woord 000) |
| SR | 252 | 252.00 (meest rechter bit in woord 252) |
| DM | DM0250 | Niet mogelijk |
| TC | TIM057 of CNT115 (voor de actuele waarde) | TIM057 of CNT115 (voor de completionvlag) |
| LR | LR12 | LR12.00 |

### *3.4.1 Data structuur*

Woorddata die ingevoerd wordt als decimale waarde wordt opgeslagen in BCD formaat (binary-coded decimal); woorddata die ingevoerd wordt als hexadecimale waarde wordt opgeslagen in binair formaat. Elke vier bits van een woord representeren één digit. Dit element van vier bits wordt ook wel een nibble genoemd. Zowel het hexadecimale als het decimale digit is numeriek equivalent aan de waarde van de vier bits waaruit het digit is opgebouwd. Eén woord data bevat dus vier digits, die van rechts naar links worden genummerd. De aan deze digits toegewezen nummers en de daarbij horende bitnummers voor een woord worden hieronder getoond.

| Woord | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte | 1 (msb) | | | | | | | | 0 (lsb) | | | | | | | |
| Digit | 3 | | | | 2 | | | | 1 | | | | 0 | | | |
| Bit | 15 | 15 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |

Wanneer gerefereerd wordt naar de digits in een woord, wordt het digit met het nummer 0 het meest rechter of meest significante digit (LSD) genoemd en het digit met het nummer 3 het meest linker of meest significante digit (MSD).

Wanneer data wordt geplaatst op datagebieden moet het in het juiste formaat worden ingevoerd. Dit is geen probleem wanneer individuele bits geadresseerd worden. Deze worden alleen aan (equivalent van de binaire waarde 1) of uit (een binaire waarde 0) gezet. Echter bij het invoeren van woorddata is het belangrijk of dit decimaal (BCD) of hexadecimaal ingevoerd moet worden, afhankelijk van wat de gebruikte instructie wenst. Het hoofdstuk "Instructieset" op pagina 71 specificeert het door instructies gebruikte data formaat.

### *3.4.2 Verschillende vormen data omzetten*

Binaire en hexadecimale data kan eenvoudig omgezet worden tussen deze twee talstelsels, aangezien elke vier bits van een binair nummer numeriek gelijk zijn aan één cijfer van een hexadecimaal nummer. Het binaire getal 0101111101011111 wordt omgezet naar een hexadecimaal getal door steeds een groepje van vier bits apart te nemen vanaf het meest rechter bit. Binair 1111 is gelijk aan hexadecimaal F; binair 0101 is gelijk aan hexadecimaal 5. Het hexadecimale equivalent is dus 5F5F of 24,415 decimaal (4095 x 5 + 265 x 15 + 16 x 5 + 15).

Decimaal en BCD kunnen eenvoudig omgezet worden. In dit geval is elk BCD cijfer (d.w.z. elke groep van vier bits) numeriek equivalent aan het corresponderende decimale cijfer. De BCD bits 0101011101010111 worden geconverteerd naar decimaal door elke groep van vier bits vanaf rechts apart te nemen. Binair 0101 is gelijk aan decimaal 5; binair 0111 is gelijk aan decimal 7. Het decimale equivalent is dus 5757. Houd in de gaten dat dit niet dezelfde waarde is als het hexadecimale equivalent van 0101011101010111 dat 5757 hexadecimaal is of 22359 decimaal (4095 x 5 + 256 x 7 + 16 x 5 + 7).

Omdat het numerieke equivalent van elke vier BCD binaire bits gelijk is aan een decimaal cijfer, kunnen bits combinaties die een numeriek groter resultaat opleveren dan 9 niet gebruikt worden. D.w.z., 1011 is niet toegestaan omdat dit het numerieke equivalent is van hexadecimaal B, wat niet uitgedrukt kan worden in één cijfer in decimale notatie. De binaire code 1011 is natuurlijk wel toegestaan in een hexadecimale code en is equivalent aan het hexadecimale cijfer B.

Er zijn instructies beschikbaar om data van BCD naar hexadecimaal of vice versa om te zetten. Raadpleeg het hoofdstuk "Dataconversie" op pagina 120 voor meer details. Tabellen met het binaire equivalent van hexadecimale en BCD cijfers zijn opgenomen in de appendix (“conversietabel hexadecimaal, bcd, binair” op pagina 159).

### *3.4.3 Decimale punt*

De decimale punt wordt alleen bij timers gebruikt. Hier representeert het minst significante cijfer tienden van een seconde. (De punt wordt echter bij de timer waarde niet ingevoerd). Alle andere instructies in de PLC werken in principe met integers.

## 3.5 IR (interne relais) gebied

Het IR gebied wordt gebruikt om I/O punten aan te sturen en om intern in de PLC data op te kunnen slaan en te manipuleren. Het IR gebied is op bit- en woordbasis benaderbaar.

Woorden in het IR gebied die worden gebruikt om I/O punten aan te sturen worden I/O woorden genoemd. Bits in I/O woorden worden I/O bits genoemd. Bits in het IR gebied die niet zijn toegewezen aan I/O kunnen worden gebruikt als werkbits. Werkbits uit het IR gebied worden gereset als de voedingsspanning wordt onderbroken of als de verwerking van het PLC programma wordt gestopt of gestart.

**I/O woorden**

Een signaal dat van buiten de PLC naar binnen gaat is een inputsignaal, het bit dat eraan is toegewezen is een inputbit. Een signaal dat vanuit de PLC naar buiten gaat is een outputsignaal, het erbij horende bit is een outputbit. Om een uitgang

aan te zetten moet het erbij horende outputbit aangezet worden. Wanneer een input aan gaat, gaat het erbij horende inputbit ook aan. Deze feiten kunnen gebruikt worden in het programma in inputstatussen te bereiken en outputstatussen aan te sturen via I/O bits.

**Inputbit gebruik**

Inputbits kunnen gebruikt worden om externe signalen op de PLC aan te sluiten en kunnen op elke mogelijke manier in het programma gebruikt worden. Elke inputbit kan zo vaak als gewenst in het programma gebruikt worden om een juiste aansturing te krijgen. Inputbits kunnen niet gebruikt worden in instructies die een bitstatus aansturen zoals de OUT, DIFU en KEEP instructies.

**Outputbit gebruik**

Outputbits worden gebruikt om resultaten uit het programma naar buiten te brengen en kunnen op elke plaats in het programma gebruikt worden. Omdat uitgangen maar één keer per scan worden aangestuurd (d.w.z. een keer per uitvoer van het programma), kan elk outputbit maar in één instructie die de status aanstuurt gebruikt worden. Dit zijn instructies als OUT, KEEP(11), DIFU(13), DIFD(14) en SFT(10). Als contact mag het outputbit zo vaak gebruikt worden als gewenst is. Als een outputbit meer dan één keer in het programma wordt aangestuurd, wordt de status van de uitgang bepaald door de laatste instructie die het bit aanstuurt.

**I/O adressen**

Inputbits beginnen op 000.00 en outputbits beginnen op 010.00. Bij de CPM1(A) kan alleen 000.00 t/m 009.15 gebruikt worden voor inputbits en alleen 010.00 t/m 019.15 kan gebruikt worden voor outputbits. Alhoewel het I/O gebied uit 20 woorden bestaat kan de CPM1(A) momenteel maximaal 50I/O aansturen.

**1, 2, 3...**

1) Bij het IR gebied wordt het acroniem IR niet voor het bit of woordnummer geplaatst. Bij de andere datagebieden gebeurd dit wel.
2) Inputbits kunnen niet worden gebruikt in output instructies. Gebruik hetzelfde outputbit niet in meer dan één OUT en/of OUT NOT instructie. In principe is het niet echt fout om het wel te doen, maar het kan vreemde effecten opleveren die voor veel gebruikers niet verklaarbaar zijn.

**Werkbits**

Bij de CPM1(A) kunnen de bits 200.00 t/m 231.15 vrijelijk in het programma gebruikt worden als werkbit. Ze kunnen echter alleen gebruikt worden in het programma en niet voor externe I/O. Werkbits worden allemaal uitgezet wanneer de CPM1(A) voedingsspanning uitgezet wordt of wanneer de PLC begint of stopt met het verwerken van het programma.

## 3.6 SR (speciale relais) gebied

Het SR gebied bevat vlaggen en controlebits die gebruikt kunnen worden voor het bekijken van de werking van de PLC, voor toegang tot de klokpulsen en instructie status en errorvlaggen. Het adresbereik van het SR gebied hangt af van het type CPU dat wordt geprogrammeerd.

De volgende tabellen geven een lijst van vlaggen en controlebits in het SR gebied en hun functie. De meeste van deze bits worden verderop in dit hoofdstuk meer in detail beschreven. In principe is de beschrijving op volgorde van het bitnummer. Alleen bits die bij dezelfde functie horen zijn gegroepeerd.

Tenzij anders vermeld wordt zijn vlaggen uit tot de gespecificeerde conditie optreedt, waarop ze aangezet worden. Restartbits zijn normaal uit maar als de gebruiker ze aan en vervolgens uit zet zal de gespecificeerde functie gereset worden. Andere controlebits zijn uit tot de gebruiker de status verandert.

### *3.6.1 SR gebied overzicht*

De toewijzing van vlaggen en bits in het Special Relais Gebied is afhankelijk van het geselecteerde CPU type. Niet alle CPU typen ondersteunen alle bits of woorden in het SR Gebied.

| Woord | Bit(s) | Functie |
|---|---|---|
| 232 t/m 235 | 00 t/m 15 | Macro functie input gebied (werkbits als de MCRO(99) instructie niet gebruikt wordt). |
| 236 t/m 239 | 00 t/m 15 | Macro functie output gebied (werkbits als de MCRO(99) instructie niet gebruikt wordt). |
| 240 | 00 t/m 15 | Input interrupt 0 counter mode ingestelde waarde (hex) |

| Woord | Bit(s) | Functie |
|---|---|---|
| 241 | 00 t/m 15 | Input interrupt 1 counter mode ingestelde waarde (hex) |
| 242 | 00 t/m 15 | Input interrupt 2 counter mode ingestelde waarde (hex) |
| 243 | 00 t/m 15 | Input interrupt 3 counter mode ingestelde waarde (hex) |
| 244 | 00 t/m 15 | Input interrupt 0 counter mode actuele waarde min één (hex) |
| 245 | 00 t/m 15 | Input interrupt 1 counter mode actuele waarde min één (hex) |
| 246 | 00 t/m 15 | Input interrupt 2 counter mode actuele waarde min één (hex) |
| 247 | 00 t/m 15 | Input interrupt 3 counter mode actuele waarde min één (hex) |
| 248 t/m 249 | 00 t/m 15 | Highspeed counter actuele waarde |
| 250 | 00 t/m 15 | Analoge potentiometer 0 ingestelde waarde (0 tot 200 BCD) |
| 251 | 00 t/m 15 | Analoge potentiometer 1 ingestelde waarde (0 tot 200 BCD) |
| 252 | 00 | Highspeed counter resetbit |
| | 01 t/m 07 | Niet gebruikt |
| | 08 | Periferie poort resetbit |
| | 09 | Niet gebruikt |
| | 10 | PC Setup resetbit |
| | 11 | Forced status hold bit<br>uit: De status van bits die geforceerd zijn ge(re)set wordt gewist wanneer de PLC van PROGRAM mode naar MONITOR mode wordt geschakeld.<br>aan: De status van bits die geforceerd zijn ge(re)set wordt vastgehouden wanneer de PLC van PROGRAM mode naar MONITOR mode wordt geschakeld. |
| | 12 | Data retention control bit<br>uit: De status van IR en LR bits wordt gereset wanneer de werking van de PLC wordt gestart of gestopt.<br>aan: De status van IR en LR bits wordt vastgehouden wanneer de werking van de PLC wordt gestart of gestopt. |
| | 13 | Niet gebruikt |
| | 14 | Errorlog resetbit |
| | 15 | Niet gebruikt |
| 253 | 00 t/m 07 | FAL error code |
| | 08 | Niet gebruikt |
| | 09 | Cyclustijd te groot errorvlag (Hoog als de cyclustijd boven de 100ms komt) |
| | 10 t/m 12 | Niet gebruikt |
| | 13 | Altijd aan vlag |
| | 14 | Altijd uit vlag |
| | 15 | Eerste scan vlag |
| 254 | 00 | 1 minuut klokpuls |
| | 01 | 0.02 seconde klokpuls |
| | 02 | Negatief (N) vlag |
| | 03 t/m 05 | Niet gebruikt |
| | 06 | Differential monitor complete vlag |
| | 07 | STEP(08) uitgevoerd vlag |
| | 18 t/m 15 | Niet gebruikt |
| 255 | 00 | 0.1-seconde klokpuls |
| | 01 | 0.2-seconde klokpuls |
| | 02 | 1.0-seconde klokpuls |
| | 03 | Instructie executie error (ER) vlag |
| | 04 | Carry (CY) vlag |
| | 05 | Groter dan (GR) vlag |
| | 06 | Gelijk aan (EQ) vlag |
| | 07 | Kleiner dan (LE) vlag |

**Opmerking** Schrijven is niet mogelijk op een aantal woorden uit het SR gebied

### *3.6.2 Forced status hold bit*

Het *Forced status hold bit* bepaalt of de status van bits die geforceerd geset of gereset zijn hetzelfde blijft als de PLC wordt geschakeld van PROGRAM naar MONITOR mode om de verwerking van het programma te stoppen of te starten. Als het *forced status hold bit* aan is, zullen de bit statussen vastgehouden worden. Als het bit uit is worden alle bits in hun standaard staat (uit) gezet wanneer de werking van het programma wordt gestart of gestopt. Het *forced status hold bit* is alleen effectief wanneer het in de PC Setup geactiveerd is.

De status van het *forced status hold bit* verandert niet door een spanningsinterruptie. Het *forced status hold bit* is niet werkzaam wanneer de PLC in de RUN mode wordt gezet. Het *forced status hold bit* kan alleen met periferie ingesteld worden. De PLC kan het niet vanuit het programma aansturen.

**Status handhaven tijdens opstarten**

De status van het *forced status hold bit* en dus de status van de geforceerd gesette en geresette bits kan worden vastgehouden wanneer de spanning uit- / aangezet wordt. Dit wordt gerealiseerd door het in de PC Setup in te stellen. Als deze optie wordt gebruikt zal de status van het *forced status hold bit* worden bewaard als de voedingsspanning uit/aan wordt gezet. Als dit wordt gedaan en het *forced status hold bit* is aan, dan zal de status van de geforceerd gesette en geresette bits worden bewaard zoals getoond in de volgende tabel. Het gebruik van het *Forced status hold* bit beïnvloedt de werking niet als de PLC in RUN mode staat. Bij het schakelen naar de RUN mode worden geforceerde bits altijd op hun standaard waarde (dat is uit) ingesteld.

| Status voor shutdown | | Status na het opstarten | |
|---|---|---|---|
| ***Forced status hold bit*** | **PC Setup** | ***Forced status hold bit*** | **geforceerde set / reset bits** |
| AAN | Uitgevoerd | AAN | Status bewaard |
| | Niet uitgevoerd | UIT | Standaard status |
| UIT | Uitgevoerd | UIT | Standaard status |
| | Niet uitgevoerd | UIT | Standaard status |

Raadpleeg het hoofdstuk "Instructieset" op pagina 71 voor details over de SYS(49) instructie.

### *3.6.3 I/O status hold bit*

Het *I/O status hold bit* bepaalt of de status van bits in het IR en LR gebied wordt bewaard wanneer de werking van de PLC wordt gestart en gestopt of van de PROGRAM naar MONITOR of RUN mode geschakeld. Als het *I/O status hold bit* aan is, worden de statussen bewaard; als het uit is, worden alle bits in het IR en LR gebied gereset. Het *I/O status hold bit* is alleen effectief wanneer het in de PC Setup geactiveerd is.

De status van het *I/O status hold bit* veranderd niet door een spanningsinterruptie. Het *I/O status hold bit* kan alleen met een periferie ingesteld worden. De PLC kan het niet vanuit het programma aansturen.

**Status handhaven tijdens opstarten**

De status van het *I/O status hold bit* en dus de status van de bits in het IR en LR gebied kan worden vastgehouden wanneer de spanning uit- en aangezet wordt. Dit wordt gerealiseerd door in de PC Setup een instelling te maken. Als deze optie wordt gebruikt zal de status van het *I/O status hold bit* worden bewaard als de voedingsspanning uit/aan wordt gezet. Als dit wordt gedaan en het *I/O status hold bit* is aan, dan zal de status van de bits in het IR en LR gebied worden bewaard zoals getoond in de volgende tabel.

| Status voor shutdown | | Status na het opstarten | |
|---|---|---|---|
| **I/O status hold bit** | **PC Setup** | **I/O status hold bit** | **IR en LR bits** |
| AAN | Uitgevoerd | AAN | Status bewaard |
| | Niet uitgevoerd | UIT | Gereset |
| UIT | Uitgevoerd | UIT | Gereset |
| | Niet uitgevoerd | UIT | Gereset |

Raadpleeg het hoofdstuk "Instructieset" op pagina 71 voor details over de SYS(49) instructie.

### *3.6.4 FAL (failure alarm) gebied*

Een 2 cijferige BCD FAL code wordt hier geplaatst wanneer een FAL of FALS instructie wordt uitgevoerd. Deze errorcodes kunnen door de gebruiker gedefinieerd worden om fouten die in een proces optreden middels errorcodes zichtbaar te maken voor errordiagnose. De PLC kan ook FAL codes op deze bits plaatsen, zoals bijvoorbeeld bij het te laag worden van de batterijspanning.

Dit gebied kan gereset worden door een FAL instructie uit te voeren met een operand 00, door een failure read uit te voeren met de programmingconsole of door een error clear te geven in het status venster van SYSWIN.

### *3.6.5 Cyclustijd te groot errorvlag*

De cyclustijd te groot errorvlag gaat aan als de cyclustijd boven de 100ms komt. De ALARM/ERROR indicator op het front van de CPU zal in dit geval gaan knipperen. Programma uitvoer zal niet stoppen tot de voor de watchdog timer maximale tijdlimiet is overschreden. Timing in het programma kan onnauwkeurig worden als de cyclustijd boven de 100ms komt.

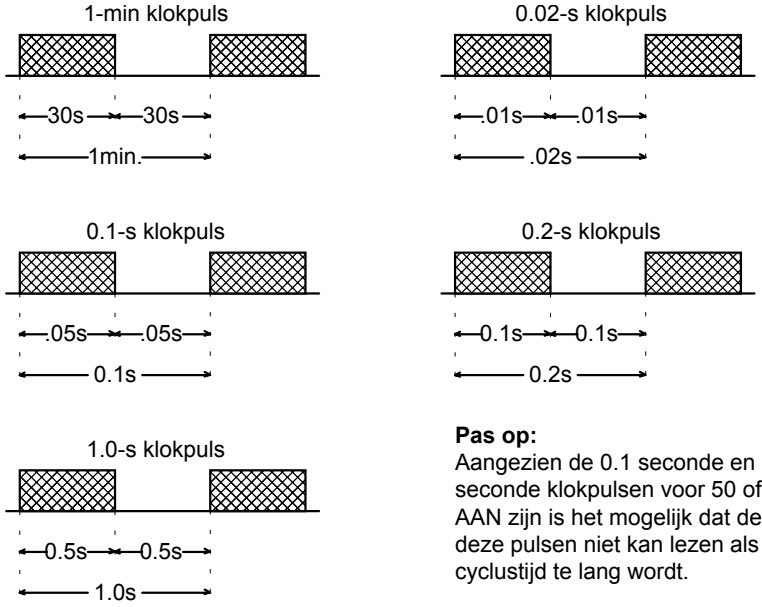
### *3.6.6 Eerste scan vlag*

De *eerste scan vlag* gaat aan als de PLC begint met de verwerking van het programma en gaat uit als het programma één keer is uitgevoerd. De *eerste scan vlag* kan gebruikt worden om counters of delen van het programma te initialiseren. Een voorbeeld van het initialiseren van een counter wordt gegeven in hoofdstuk "Timer en counter instructies" op pagina 86.

### *3.6.7 Klokpuls bits*

Vijf klokpulsen zijn beschikbaar voor timing in het programma. Elk klokpuls bit is aan voor de eerste helft van de tijd en uit voor de tweede helft. Met andere woorden, elke klokpuls heeft een duty factor van 50%.

Deze klokpuls bits worden vaak gebruikt om van counters timers te maken. Een voorbeeld hiervan wordt gegeven in het hoofdstuk "Timer en counter instructies" op pagina 86.




**Pas op:**
Aangezien de 0.1 seconde en 0.02 seconde klokpulsen voor 50 of 100 ms AAN zijn is het mogelijk dat de CPU deze pulsen niet kan lezen als de cyclustijd te lang wordt.

### *3.6.8 STEP(08) uitgevoerd vlag*

De *STEP(08) uitgevoerd vlag* gaat voor één programma cyclus aan wanneer een STEP wordt geactiveerd met de SNXT(09) instructie.

### *3.6.9 Instructie executie errorvlag, ER*

De *Instructie executie errorvlag* gaat aan als een poging is gedaan om een instructie uit te voeren met incorrecte operand data. De meest voorkomende oorzaak van een instructie error is gebruik van niet-BCD operand data wanneer BCD data gewenst is of een indirect geadresseerd DM woord dat niet bestaat wordt aangeroepen. Wanneer de ER vlag aan is zal de actuele instructie niet uitgevoerd worden. Aangezien elke functie in de PLC dit bit gebruikt is het alleen direct achter een instructie uit te lezen.

### *3.6.10 Rekenkundige vlaggen*

De volgende vlaggen worden gebruikt bij het schuiven van data, rekenkundige bewerkingen en vergelijkingen. Ze worden in deze handleiding meestal aangeduid meet tweeletterige afkortingen.

**Opmerking** Deze vlaggen worden allemaal gereset wanneer de END(01) instructie wordt uitgevoerd en kunnen daarom niet met een programmeerapparaat bekeken worden.

Raadpleeg "Schuiven van data" op pagina 101, "Datavergelijking" op pagina 115, "BCD calculaties" op pagina 127 en "Binaire berekeningen" op pagina 136 voor details over het gebruik van deze vlaggen.

**Carry vlag, CY** De *carry vlag* gaat aan wanneer er een overdracht plaatsvindt in het resultaat van een rekenkundige bewerking of wanneer er een “1” uit een shift of rotate instructie wordt geschoven. De inhoud van CY wordt gebruikt door sommige rekenkundige instructies, d.w.z. deze wordt bij het resultaat opgeteld of ervan afgetrokken. Deze vlag kan worden geset of gereset vanuit het programma met de *set carry* STC en *clear carry* CLC instructies.

**Groter dan vlag, GR** De *groter dan vlag* gaat aan als het resultaat van een vergelijking toont dat de eerste van twee operands groter is dan de tweede.

**Gelijk aan vlag, EQ** De *gelijk aan vlag* gaat aan als het resultaat van een vergelijking toont dat de eerste van twee operands gelijk is aan de tweede of wanneer het resultaat van een rekenkundige bewerking gelijk is aan 0.

**Kleiner dan vlag, LE** De *kleiner dan vlag* gaat aan als het resultaat van een vergelijking toont dat de eerste van twee operands kleiner is dan de tweede.

**Opmerking** De vier rekenkundige vlaggen worden uitgezet als de END(01) instructie wordt uitgevoerd.

## 3.7 AR (auxiliary relais) gebied

AR woordadressen lopen van AR00 t/m AR15; AR bitadressen lopen van AR00.00 t/m AR15.15. De meeste woorden en bits in het AR gebied zijn toegewezen aan speciale functies zoals transmissie counters, vlaggen en control bits.

Het AR gebied onthoudt statussen tijdens spanningsuitval en tijdens het schakelen van MONITOR of RUN mode naar PROGRAM mode. De toewijzing van de bits is getoond in de volgende sectie en wordt verklaard in de daaropvolgende tekst.

### *3.7.1 AR gebied overzicht*

De vlaggen en bits toewijzing in het auxiliary relais gebied is afhankelijk van de geselecteerde CPU.

| Woord(en) | Bit(s) | Functie |
|---|---|---|
| 00 t/m 01 | --- | Niet gebruikt |
| AR02 | 00 t/m 07 | Niet gebruikt |
| | 08 t/m 11 | Aantal op de CPU aangesloten I/O units |
| | 12 t/m 15 | Niet gebruikt |
| 03 t/m 07 | --- | Niet gebruikt |
| 08 | 00 t/m 07 | Niet gebruikt |
| | 08 t/m 11 | Periferie device error code (1 cijfer nummer) |
| | 12 | Periferie device errorvlag |
| | 13 t/m 15 | Niet gebruikt |
| 09 | 00 t/m 15 | Niet gebruikt |
| 10 | 00 t/m 15 | Spanning uit counter (4 cijfers BCD) |
| 11 | 00 t/m 07 | Highspeed counter 0 bereik vergelijking vlaggen |
| | 08 t/m 14 | Niet gebruikt |
| | 15 | Puls uitgang status |
| 12 | 00 t/m 15 | Niet gebruikt |
| 13 | 00 | Power-up PC Setup error vlag (DM6600 t/m 6614) |
| | 01 | Start-up PC Setup error vlag (DM6615 t/m 6644) |
| | 02 | RUN PC Setup error vlag (DM6645 t/m 6655) |
| | 03, 04 | Niet gebruikt |

| Woord(en) | Bit(s) | Functie |
|---|---|---|
| | 05 | Lange cyclustijd vlag (hoog als cyclustijd boven de op DM6619 ingestelde tijd komt) |
| | 06, 07 | Niet gebruikt |
| | 08 | Niet aanwezig adres gebruikt (Is hoog als een niet in de CPM1(A) aanwezig adres in het programma wordt gebruikt) |
| | 09 | Flash memory checksum error |
| | 10 | Read only DM checksum error |
| | 11 | PC Setup checksum error |
| | 12 | Programma checksum error of een niet toegestane instructie gebruikt |
| | 13 | Instructietabel checksum error |
| | 14, 15 | Niet gebruikt |
| 14 | 00 t/m 15 | Maximum cyclustijd (4 cijfers BCD) |
| 15 | 00 t/m 15 | Actuele cyclustijd (4 cijfers BCD) |

De onderstaande secties beschrijven de belangrijkste AR bits. Voor een verklaring van de overige bits wordt verwezen naar de Engelstalige manuals. De meeste niet verklaarde bits worden alleen voor bepaalde zeer specifieke functies gebruikt; in dat geval worden ze in deze handleiding behandeld bij deze functie.

### *3.7.2 Power-OFF counter*

AR10 bevat in 4-cijfers BCD het aantal maal dat de spanning van de PLC is uitgezet. Indien nodig kan deze waarde veranderd worden door er 0000 in te schrijven met SYSWIN. De power-OFF counter wordt elke keer gerefreshed als de spanning op CPU/Powersupply wordt gezet.

### *3.7.3 Lange cyclustijd vlag*

AR13.05 is hoog wanneer de cyclustijd die ingesteld is in de PC Setup korter is dan de actuele cyclustijd.

AR13.05 wordt elke scan gerefreshed als de PLC in RUN of MONITOR mode is.

### *3.7.4 Cyclustijd indicators*

AR14 bevat de grootste cyclustijd die is opgetreden sinds de uitvoer van het programma is begonnen. AR1527 bevat de cyclustijd van de vorige scan.

Beide tijden worden default opgegeven in tienden van milliseconden in een 4 cijferig BCD getal (000.0ms t/m 999.9ms) en worden elke cyclus gerefreshed Afhankelijk van de instelling op DM6618 kan deze tijd ook worden opgegeven in units van 10, 100 of 1000ms.

## 3.8 DM (data memory) gebied

Het DM gebied is een datageheugen waarin gelezen en geschreven kan worden. Bij sommige CPU's is een deel van het DM gebied alleen te lezen. Voor de een aantal CPU's wordt dit in de onderstaande tekst aangegeven. Raadpleeg voor specifieke CPU's het Operation Manual voor meer details.

Alhoewel het DM gebied is samengesteld uit 16 bit woorden zoals de andere data gebieden kan het DM geheugen niet op bitniveau geadresseerd worden. Het DM gebied behoudt statussen tijdens spanningsinterrupties.

**Error historie gebied**

Sommige CPU's gebruiken een deel van het Read/Write DM Gebied om records in op te slaan die de aard, tijd en datum van storingen bevatten die in de PLC zijn voorgekomen. De tijd en datum velden in deze records worden alleen ingevuld bij PLC's die zijn voorzien van een kalender/klok functie. Raadpleeg de handleiding van de gebruikte CPU voor details over de codering van errors en de structuur van de error records.

De inhoud van het fixed DM, de PC Setup, het PLC programma en de instructietabel worden in zijn geheel opgeslagen in een flash rom.

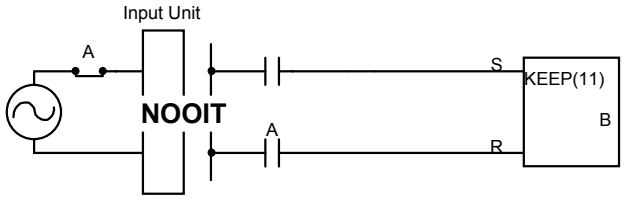
## 3.9 HR (holding relais) gebied

Het HR gebied wordt gebruikt om diverse soorten data in op te slaan of te manipuleren. Het gebied kan op woord- en bitniveau geadresseerd worden. HR bits kunnen in willekeurige volgorde gebruikt worden en kunnen zo vaak geprogrammeerd worden als gewenst is, met dien verstande dat het voor elk bit in de CPU aan te raden is om het maar één keer als output aan te sturen.

Het HR gebied onthoudt de status wanneer de operating mode van de PLC wordt veranderd en wanneer de spanning wordt uitgezet.

Bits en woorden uit het HR gebied kunnen gebruikt worden om data vast te houden als de PLC gestopt wordt. Daarnaast kunnen HR bits ook gebruikt worden voor diverse speciale toepassingen zoals het creëren van latchrelais met de KEEP instructie etc. Dit wordt besproken in het hoofdstuk "geheugengebieden" op pagina 58 en in "Programma uitvoer" op pagina 30.

| **Voorzichtig** | Gebruik nooit een input bit van een NC (normally closed) input op de reset conditie van een KEEP(11) instructie wanneer het input device een AC powersupply gebruikt. De vertraging in het uitgaan van de PLC's DC powersupply (relatief ten opzichte van de AC powersupply op het input device) kan ervoor zorgen dat het aan de KEEP(11) instructie toegewezen bit wordt gereset wanneer de besturing wordt uitgezet. Deze situatie wordt hieronder getoond. |
|---|---|




## 3.10 TC (timer/counter) gebied

Het TC gebied wordt gebruikt om timers en counters te creëren en te programmeren en bevat de completionvlaggen, ingestelde waarden en actuele waarden voor alle timers en counters. Tot al deze gebieden wordt toegang gekregen met het TC nummer. Het bereik van deze nummers voor de verschillende CPU typen kan worden gevonden in sectie "geheugengebieden" op pagina 58. Elk TC nummer kan als timer of counter gedefinieerd worden met één van de volgende instructies: TIM, TIMH, CNT of CNTR(12). Er hoeft geen prefix (TIM of CNT) ingevoerd te worden wanneer een TC nummer in een timer of counter instructie gebruikt wordt.

Zodra een TC nummer is toegewezen met een van de bovenstaande instructies is het niet mogelijk om deze een tweede keer toe te wijzen voor dezelfde of een andere instructie. Als hetzelfde TC nummer is gebruikt in meer dan één van deze instructies of tweemaal in dezelfde instructie dan wordt een error gegenereerd tijdens de program check. Er zijn geen beperkingen voor de volgorde waarin TC nummers gebruikt worden. Voor het dubbel gebruiken van dezelfde TC nummers bij meerdere instructies gelden dezelfde regels als voor het dubbel gebruiken van uitgangen. Alhoewel dit in sommige situaties nodig kan zijn, is het aan te raden om dit achterwege te laten.

Een TC nummer kan worden gebruikt als operand in één of meer instructies, anders dan de hierboven genoemde. Wanneer een timernummer gebruikt wordt als operand in een instructie moet het nummer vooraf worden gegaan door de prefix TIM. De TIM prefix wordt gebruikt onafhankelijk van de timer instructie die gebruikt is om de timer te definiëren. Wanneer een counternummer gebruikt wordt als operand in een instructie moet het nummer vooraf worden gegaan door de prefix CNT. De CNT prefix wordt ook onafhankelijk gebruikt van de counter instructie die gebruikt is om de counter te definiëren. De TIM en CNT prefixen worden gebruikt om een veld in het TC gebied aan te duiden. Het in mogelijk om met de TIM prefix een veld op te geven dat door een counter gebruikt wordt. In dit geval zal de counter waarde getoond worden. Alhoewel het verder geen consequenties heeft voor de uitvoer van het programma is het voor het overzicht in het programma niet aan te raden om TIM en CNT prefixen om te wisselen.

TC nummers kunnen worden gebruikt voor operands die bitdata willen hebben en voor operands die woorddata gebruiken. Wordt een TC nummer gebruikt in een instructie op bitniveau dan geeft het nummer toegang tot de completionvlag van de timer of counter. Wanneer een TC nummer wordt gebuikt in een instructie die woorddata verlangt dan geeft het TC nummer toegang tot een geheugenlocatie die de actuele waarde van de timer of counter bevat.

Het TC gebied bewaart de ingestelde waarden van timers en counters tijdens spanningsuitval. De actuele waarden van timers worden gereset wanneer PLC begint met de uitvoer van het programma. De actuele waarde van counters worden niet gereset bij spanningsuitval of het starten van het programma.

**Opmerking** TIM000 wordt gebruikt tijdens het programmeren om drie zaken aan te duiden. De timer met het TC nummer 000, de completionvlag van deze timer en de actuele waarde van deze timer. De samenhang zal duidelijk zijn. De eerste is altijd een instructie, de tweede is altijd een bit en de derde is altijd een woord. Dit geldt voor alle TC nummers, onafhankelijk of de prefix TIM of CNT is.

## 3.11 LR (link relais) gebied

Het LR gebied wordt gebruikt als gemeenschappelijk gebied om data tussen PLC's uit te wisselen. Deze data transfer is mogelijk bij een 1:1 link. Bepaalde woorden worden toegewezen als schrijf woorden aan elke PLC. Op deze woorden kan de PLC data plaatsen waarna het automatisch wordt verplaatst naar dezelfde LR woorden bij de andere PLC's in het netwerk. De schrijf woorden van elke PLC kunnen op deze manier door de andere PLC's in het netwerk gelezen worden. Een PLC kan alleen data op de schrijf woorden in het LR gebied plaatsen, alle andere woorden kunnen alleen gelezen worden. Wordt er data geplaatst op de lees woorden in het LR gebied dan zal dit worden overschreven door de data uit schrijf woorden uit andere PLC's.

Het LR gebied is op woord- of bitniveau toegankelijk. Het toegestane bereik aan LR woorden is gedefinieerd in "geheugengebieden" op pagina 58 voor de verschillende CPU typen. Elk deel van het LR gebied dat niet is gebruikt door de communicatie kan worden gebruikt als werkwoorden of werkbits.

Data in het LR gebied wordt niet onthouden bij spanningsuitval of wanneer de PLC in PROGRAM mode wordt gezet.

## 3.12 Programmageheugen

Programmageheugen is het geheugen waarin het door de gebruiker geschreven programma wordt opgeslagen. Het programmageheugen wordt meestal aangeduid met UM. De grootte van het programmageheugen is afhankelijk van het type van de CPU. Om instructies in het programmageheugen op te slaan kan een handprogrammeerapparaat of SYSWIN gebruikt worden.

## 3.13 TR (temporary relais) gebied

**TR gebied**

Wanneer een complex ladderdiagram niet kan worden geprogrammeerd in mnemonic code zoals het op het scherm staat, dan worden deze bits gebruikt om tijdelijk aan/uit executiecondities op programma aftakkingen op te slaan. Deze acht bits kunnen alleen in statementlist bij LD en OUT instructies gebruikt worden. TR adressen lopen van TR0 t/m TR7. Elk van deze bits kan zo vaak als gewenst gebruikt worden in elke gewenste volgorde. Wanneer in ladderdiagram geprogrammeerd wordt plaatst SYSWIN de benodigde TR bits automatisch. Het gebruik van TR bits wordt beschreven in de sectie "Vertakkende instructie regels" op pagina 19.

Hetzelfde TR bit kan in principe niet meer dan één keer gebruikt worden in hetzelfde instructie blok, maar kan opnieuw gebruikt worden in een ander instructie blok. De aan/uit status van TR bits kan niet worden bekeken met periferie zoals bijvoorbeeld SYSWIN.

# 4 Instructieset

De OMRON SYSMAC C- en CV-serie PLC's beschikken over een uitgebreide instructieset die het mogelijk maken dat gecompliceerde processen eenvoudige geprogrammeerd kunnen worden. De volgende secties beschrijven de instructies individueel en geven het ladderdiagramsymbool, de datagebieden die gebruikt kunnen worden en de vlaggen die door de instructie beïnvloed worden.

De instructies waarover OMRON PLC's beschikken worden in een aantal instructie groepen onderverdeeld. Deze groepen zijn:

- ⇨ Ladderdiagram instructies
- ⇨ Bit control instructies
- ⇨ Timer en counter instructies
- ⇨ Data schuif instructies
- ⇨ Data verplaatsing instructies
- ⇨ Data vergelijking instructies
- ⇨ Data conversie instructies
- ⇨ BCD calculatie instructies
- ⇨ Binaire calculatie instructies
- ⇨ Logische instructies
- ⇨ Subroutines
- ⇨ Speciale instructies
- ⇨ Block programma instructies
- ⇨ Netwerk instructies.

Sommige instructies, bijvoorbeeld timer en counter instructies, worden gebruikt om andere instructies aan te sturen. D.w.z., een timer completionvlag kan gebruikt worden om een bit aan te zetten wanneer de voor de timer ingestelde tijd is afgelopen. Alhoewel deze instructies vaak gebruikt worden om outputbits met de OUT instructie aan te sturen, kunnen ze ook gebruikt worden om weer andere instructies aan te sturen. De output instructies die in voorbeelden gebruikt worden kunnen over het algemeen vervangen worden door andere instructies om het programma te modificeren voor specifieke applicaties anders dan het direct aansturen van uitgangen.

## 4.1 Notatie

In het vervolg van deze handleiding worden alle instructies aangeduid met hun mnemonics. Bijvoorbeeld, de output instructie wordt OUT genoemd, de en instructie AND en de load instructie LD. Als u er niet zeker van bent welke mnemonic een instructie gebruikt, raadpleeg dan de sectie "Alfabetische instructielijst op mnemonic" op pagina 74.

Als een instructie een functiecode heeft wordt deze tussen haken achter de mnemonic geplaatst. Deze functiecodes bestaan bij de C-serie uit een tweecijferig decimaal nummer en worden gebruikt tijdens het programmeren van de functies. In SYSWIN is het mogelijk om naast het functienummer ook de naam in te voeren. De functie worden hieronder kort beschreven.

Een @ voor een mnemonic geeft aan dat de gedifferentieerde uitvoering van deze instructie is gebruikt. Gedifferentieerde instructies worden uitgelegd in "Gedifferentieerde instructies" op pagina 73.

## 4.2 Instructie formaat

De meeste instructies hebben minimaal één operand. Operands verwijzen naar de data waarop of waarmee de instructie zijn taak kan uitvoeren. Deze worden soms ingevoerd als numerieke waarden (constanten), maar zijn meestal de adressen van woorden in een datagebied die de data bevatten waarop de bewerking moet worden uitgevoerd. Een bit waarvan het adres wordt gebruikt als operand wordt een operandbit en een woord waarvan het adres wordt gebruikt als een operand wordt een operandwoord genoemd. Bij sommige instructies duidt het woordadres

in een instructie op het eerste woord van een reeks woorden die de gewenste data bevatten.

Elke instructie gebruikt één of meer woorden in het programma geheugen. Het eerste woord is het instructie woord. Dit woord specificeert de instructie en bevat de definers (hieronder beschreven) of operandbits die door de instructie gebruikt worden. De overige operands, die de instructie gebruikt, worden in de volgende woorden opgeslagen, één operand per woord. Sommige instructies gebruiken tot aan vier woorden uit het programmageheugen.

Een definer is een operand die nauw verbonden is met de instructie. Deze operands definiëren de instructie meer dan dat ze aangeven welke data gebruikt dient te worden. Voorbeelden van definers zijn TC nummers die gebruikt worden in timer en counter instructies en sprongnummers die bepalen welke spronginstructie is gekoppeld met welke sprong end instructie. Bit operands worden ook in hetzelfde woord als de instructie opgeslagen, alhoewel deze niet worden beschouwd als definers. Bij SYSWIN is het verschil tussen definers en operands niet duidelijk te zien.

## 4.3 Datagebieden, definer waarden en vlaggen

Deze sectie bevat, naast de beschrijving van elke instructie, het ladderdiagram symbool, de datagebieden die als operand en de waarden die als definer gebruikt kunnen worden. Daarnaast wordt ook het type van de data die de instructie gebruikt (HEX of BCD) en het type van de operand (bit, woord of meerdere woorden) aangegeven.

Niet alle adressen in de gespecificeerde datagebieden zijn toegestaan voor een operand. Bijvoorbeeld, als een operand bestaat uit twee woorden, dan kan het laatste woord uit een datagebied niet worden toegewezen als het eerste woord van de operand. Immers, alle woorden van een enkele operand moeten in hetzelfde datagebied liggen. Andere specifieke beperkingen worden per instructie gegeven in een subsectie. Raadpleeg het hoofdstuk “geheugengebieden" op pagina 58 voor conventies in het gebruik van adressen en de adressen van vlaggen en control bits.

**Opmerking** Het IR en SR gebied moeten beschouwd worden als aparte datagebieden. Als een operand toegang heeft tot het ene gebied hoeft dezelfde operand niet noodzakelijkerwijs toegang te hebben tot het andere gebied. Een enkele operand kan echter de grenzen tussen het IR en SR gebied overschrijden. D.w.z. dat het laatste woord in het IR gebied mag worden gespecificeerd voor een operand die meer dan één woord lang is, zolang het SR gebied is toegestaan voor deze operand.

Wanneer het IR gebied als input operand gebruikt kan worden is het ook toegestaan om het SR gebied te gebruiken. Ook als het IR gebied als output operand gebruikt kan worden is het toegestaan om het SR gebied te gebruiken, in dit geval mogen echter alleen die bits/woorden uit het SR gebied gebruikt worden waarop data mag worden geschreven. Voor een instructie die met woorden werkt, geldt dat als het resultaat op een woord in het SR gebied wordt geplaatst dit hele woord beschreven moet kunnen worden. Wordt het resultaat op meer dan één woord in het SR gebied geplaatst, dan moeten al deze woorden beschreven kunnen worden.

Bij de instructies die in dit hoofdstuk behandeld worden wordt het SR gebied niet als apart geheugengebied vermeld. De regels uit de bovenstaande alinea gelden in dat geval voor het gebruik van het SR gebied.

De vlaggen subsectie geeft een lijst van vlaggen die door de uitvoer van de instructie beïnvloed worden. De meest gebruikte vlaggen zijn hieronder weergegeven.

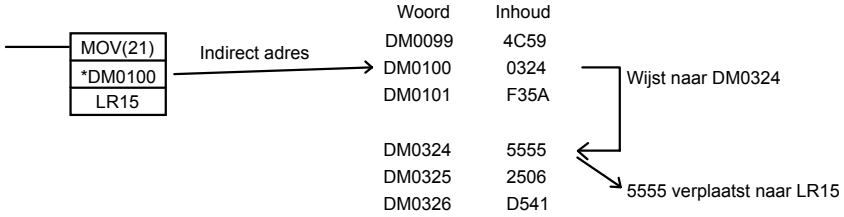
| Afkorting | Naam |
|---|---|
| ER | Instructie executie errorvlag |
| CY | Carry vlag |
| GR | Groter dan vlag |
| EQ | Gelijk aan vlag |
| LE | Kleiner dan vlag |

ER is de vlag die het meest gebruikt wordt om te controleren of een instructie correct uitgevoerd wordt. Wanneer ER aan gaat, geeft dit aan dat een error is voorgekomen tijdens het uitvoeren van deze instructie. Het is zaak om het ER bit direct achter de instructie uit te lezen en de status op te slaan, aangezien elke instructie (op de logische na) het ER bit aanstuurt. De vlaggen subsectie van elke instructie geeft een lijst van mogelijkheden waarom het ER bit aan zou kunnen zijn. ER gaat aan als operands niet correct zijn ingevoerd. Instructies zijn niet uitgevoerd wanneer ze het ER bit aanzetten. Een tabel met instructies en de vlaggen die erdoor beïnvloed worden is opgenomen in de appendix.

### *4.3.1 Indirect adresseren*

Normaal, wanneer een woord uit een datagebied is gespecificeerd bij een instructie wordt de bewerking van de instructie rechtstreeks op de inhoud van dat woord uitgevoerd. Bijvoorbeeld, stel een MOV(21) (uitleg op pagina 108) instructie wordt uitgevoerd met als eerste operand DM0100 en LR15 als de tweede. Dan wordt de inhoud van woord DM0100 door deze instructie verplaatst naar woord LR15.

Het is echter mogelijk om indirecte DM adressen te gebruiken als operand voor de meeste instructies. Om een indirect DM adres aan te geven wordt een asterix (*) voor het DM woord geplaatst, zoals bijvoorbeeld in *DM0100. Bij het gebruiken van een indirect adres bevat het operands adres niet de data die gebruikt gaan worden maar het adres van een ander DM woord. De inhoud van dat indirect aangewezen DM woord zal door de instructie gebruikt worden. Als *DM0100 wordt gebruikt in het voorbeeld hierboven en de inhoud van DM0100 is 0324, dan zal *DM0100 betekenen dat de inhoud van DM0324 als operand wordt gebruikt door de instructie en dat de inhoud van DM0324 wordt verplaatst naar LR15.




Wanneer indirect adresseren gebruikt wordt, moet het adres van het gewenste woord in BCD worden opgegeven en het moet een woord specificeren dat binnen de grenzen van het DM gebied ligt. In het bovenstaande voorbeeld zou de inhoud van *DM0000 voor een CPM1(A) in BCD tussen de 0000 en 999 moeten liggen (1000 en hoger is read only of bestaat niet). Wordt hier niet aan voldaan dan zal de instructie het ER bit hoog zetten en niet uitgevoerd worden.

### *4.3.2 Constanten benoemen*

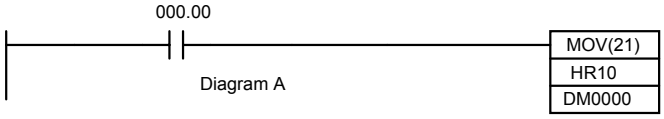
Alhoewel als operand meestal datagebied adressen worden ingevoerd, worden vaak ook constanten ingevoerd als operand. Definers kunnen zelfs alleen maar als constante worden ingevoerd. Het te gebruiken bereik voor een bepaalde definer of constante hangt af van de gebruikte instructie. Constanten moeten worden ingevoerd in het formaat dat de instructie wenst, d.w.z. in BCD of in hexadecimaal. Kan als operand van een instructie naast een constante ook een adres uit een datagebied worden opgegeven dan moet de constante worden voorafgegaan door #. Kan als operand alleen een constante worden ingevoerd dan mag de # meestal niet worden ingevoerd. Een voorbeeld zijn definers, die altijd een constante zijn. Bij definers wordt nooit een # ingevoerd voor het nummer. Denk hierbij bijvoorbeeld aan de TC nummers bij timers en counters.

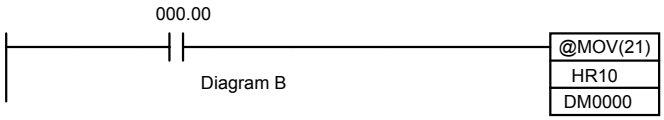
## 4.4 Gedifferentieerde instructies

De meeste instructies zijn beschikbaar in een gedifferentieerde en een niet-gedifferentieerde uitvoering. Gedifferentieerde instructies zijn te herkennen aan een @ voor de instructie mnemonic.

Een niet-gedifferentieerde instructie wordt elke keer als deze gescand wordt uitgevoerd, zolang de executieconditie aan is. Een gedifferentieerde instructie

wordt slechts één keer uitgevoerd als de executieconditie van uit naar aan gaat. Als de executieconditie niet verandert of veranderd is van aan naar uit sinds de vorige keer dat de instructie gescand is, dan zal deze niet uitgevoerd worden. De volgende twee voorbeelden tonen hoe dit werkt met MOV(21) en @MOV(21), die gebruikt worden om de data op het adres dat aangegeven wordt, door de eerste operand te verplaatsen naar het adres dat wordt aangegeven door de tweede operand.




| **Adres** | **Instructie** | **Operands** | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | MOV(21) | HR10 | DM0000 |

| **Adres** | **Instructie** | **Operands** | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | @MOV(21) | HR10 | DM0000 |

In diagram A zal de niet-gedifferentieerde MOV(21) instructie de inhoud van HR10 naar DM0000 kopiëren zolang deze gescand wordt en 000.00 aan is. Als de cyclustijd 80ms is en 000.00 is aan voor 2,0 seconden, dan wordt deze kopieer bewerking 25 maal uitgevoerd en alleen de laatste waarde die naar DM0000 is gekopieerd zal bewaard worden.

In diagram B zal de gedifferentieerde @MOV(21) de inhoud van HR10 maar één keer naar DM0000 kopiëren, op het moment dat 000.00 aan gaat. Zelfs in het geval dat 000.00 2,0 seconden aan blijft met dezelfde 80ms cyclustijd, zal de kopieer bewerking slechts één maal worden uitgevoerd, op het moment dat de status van 000.00 van uit in aan veranderd. Aangezien de inhoud van HR10 kan veranderen tijdens de 2 seconden dat 000.00 aan is, kan de uiteindelijke inhoud van DM0000 na 2 seconden verschillen, afhankelijk of MOV(21) of @MOV(21) is gebruikt.

Alle operands, ladderdiagram symbolen en andere specificaties voor instructies zijn hetzelfde, onafhankelijk of de gedifferentieerde of niet-gedifferentieerde uitvoering van de instructie is gebruikt. Tijdens invoer worden dezelfde functiecodes of mnemonics gebruikt, er moet echter een @ ingevoerd worden voor het functienummer of mnemonic om de gedifferentieerde uitvoering van de instructie te krijgen. In SYSWIN is het mogelijk om met de cursor op een functie die al is ingevoerd te gaan staan en de slash (/) op het toetsenbord te bedienen om deze alsnog te differentiëren. De meeste, maar niet alle, instructies hebben een gedifferentieerde uitvoering.

De SYSMAC C-serie PLC's beschikken ook over differentiatie instructies: DIFU(13) en DIFD(14). DIFU(13) werkt identiek aan de gedifferentieerde instructie, maar wordt gebruikt om een bit voor één scan aan te zetten. DIFD(14) zet ook een bit voor één scan aan, maar doet dit als de executieconditie verandert van aan naar uit. Raadpleeg "differentiate up en differentiate down" op pagina 25 voor details.

## 4.5 Alfabetische instructielijst op mnemonic

De volgende tabel geeft de instructies weer die bij de CPM1(A) PLC's beschikbaar zijn. Deze tabel kan gebruikt worden om de functiecode van een mnemonic op te zoeken. Het @ symbool geeft aan dat de instructies over een gedifferentieerde variant beschikt. Bedenk dat niet elke instructie toepasbaar is bij elke PLC/CPU. Raadpleeg de sectie over de specifieke instructies om te controleren bij welke CPU's een instructie toegepast kan worden (**"Toepasbaar bij"**). Instructies waarbij in de tabel een — is geplaatst zijn uitbreidingsinstructies die niet standaard in de instructietabel zitten.

| MNEMONIC | Code | Naam |
|---|---|---|
| ADB (@) | 50 | BINARY ADD |
| ADD (@) | 30 | BCD ADD |
| AND | Geen | AND |
| AND LD | Geen | AND LOAD |
| AND NOT | Geen | AND NOT |
| ANDW (@) | 34 | LOGICAL AND |
| ASC (@) | 86 | ASCII CONVERT |
| ASFT (@) | — | ASYNCHRONOUS SHIFT REGISTER |
| ASL (@) | 25 | ARITHMETIC SHIFT LEFT |
| ASR (@) | 26 | ARITHMETIC SHIFT RIGHT |
| BCD (@) | 24 | BINARY TO BCD |
| BCMP (@) | 68 | BLOCK COMPARE |
| BCNT (@) | 67 | BIT COUNTER |
| BIN (@) | 23 | BCD-TO-BINARY |
| BSET (@) | 71 | BLOCK SET |
| CLC (@) | 41 | CLEAR CARRY |
| CMP (@) | 20 | COMPARE |
| CMPL | 60 | DOUBLE COMPARE |
| CNT | Geen | COUNTER |
| CNTR | 12 | REVERSIBLE COUNTER |
| COLL (@) | 81 | DATA COLLECT |
| COM (@) | 29 | COMPLEMENT |
| CTBL (@) | — | COMPARISON TABLE LOAD |
| DEC (@) | 39 | DECREMENT BCD |
| DIFD | 14 | DIFFERENTIATE DOWN |
| DIFU | 13 | DIFFERENTIATE UP |
| DIST (@) | 80 | SINGLE WOORD DISTRIBUTE |
| DIV (@) | 33 | BCD DIVIDE |
| DIVL (@) | 57 | DOUBLE BCD DIVIDE |
| DMPX (@) | 77 | 16-to-4 ENCODER |
| DVB (@) | 53 | BINARY DIVIDE |
| END | 01 | END |
| FAL (@) | 06 | FAILURE ALARM |
| FALS | 07 | SEVERE FAILURE ALARM |
| IL | 02 | INTERLOCK |
| ILC | 03 | INTERLOCK CLEAR |
| INC (@) | 38 | INCREMENT |
| INI (@) | — | MODE CONTROL |
| INT (@) | 89 | INTERRUPT CONTROL |
| IORF (@) | 97 | I/O REFRESH |
| JME | 05 | SPRONG END |
| JMP | 04 | SPRONG |
| KEEP | 11 | KEEP |
| LD | Geen | LOAD |
| LD NOT | Geen | LOAD NOT |
| MCRO (@) | 99 | MACRO |
| MLB (@) | 52 | BINARY MULTIPLY |
| MLPX (@) | 76 | 4-TO-16 DECODER |
| MOV (@) | 21 | MOVE |
| MOVB (@) | 82 | MOVE BIT |
| MOVD (@) | 83 | MOVE DIGIT |
| MSG (@) | 46 | MESSAGE |
| MUL (@) | 32 | BCD MULTIPLY |
| MULL (@) | 56 | DOUBLE BCD MULTIPLY |
| MVN (@) | 22 | MOVE NOT |
| NOP | 00 | NO OPERATION |
| OR | Geen | OR |
| OR LOAD | Geen | OR LOAD |
| OR NOT | Geen | OR NOT |
| ORW (@) | 35 | LOGICAL OR |
| OUT | Geen | OUTPUT |
| OUT NOT | Geen | OUTPUT NOT |
| PRV (@) | — | HIGHSPEED COUNTER actuele waarde lezen |
| RET | 93 | SUBROUTINE RETURN |
| ROL (@) | 27 | ROTATE LEFT |

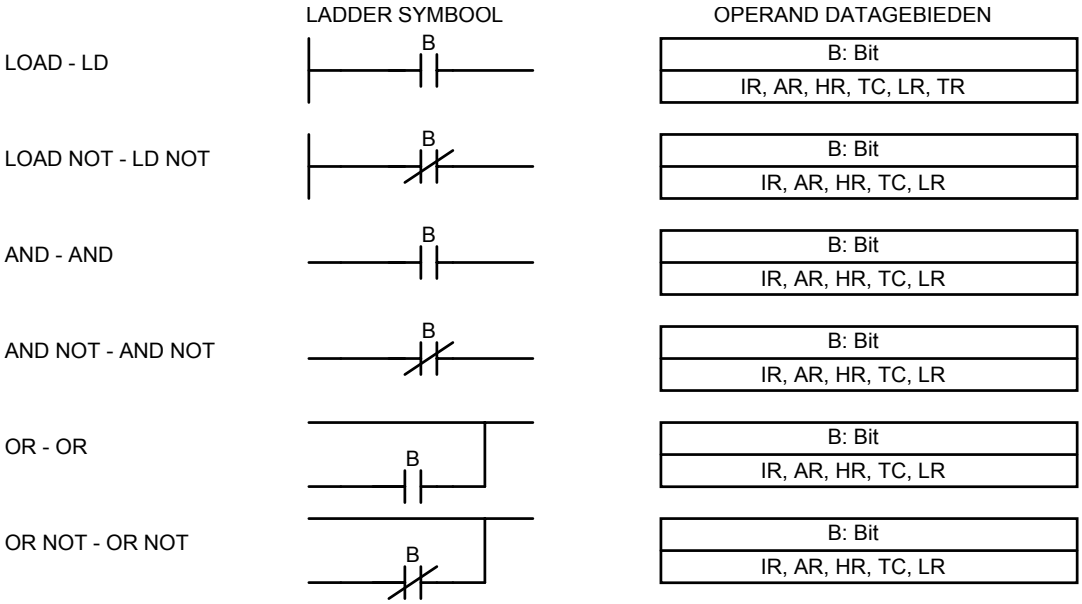
| MNEMONIC | Code | Naam |
|---|---|---|
| ROR (@) | 28 | ROTATE RIGHT |
| RSET | geen | RESET |
| SBB (@) | 51 | BINARY SUBTRACT |
| SBN | 92 | SUBROUTINE DEFINE |
| SBS (@) | 91 | SUBROUTINE ENTRY |
| SDEC (@) | 78 | 7-SEGMENT DECODER |
| SET | geen | SET |
| SFT | 10 | SHIFT REGISTER |
| SFTR (@) | 84 | REVERSIBLE SHIFT REGISTER |
| SLD (@) | 74 | ONE DIGIT SHIFT LEFT |
| SNXT | 09 | STEP START |
| SRD (@) | 75 | ONE DIGIT SHIFT RIGHT |
| STC (@) | 40 | SET CARRY |
| STEP | 08 | STEP DEFINE |
| STIM (@) | — | INTERVAL TIMER |
| SUB (@) | 31 | BCD SUBTRACT |
| SUBL (@) | 55 | DOUBLE BCD SUBTRACT |
| TCMP (@) | 85 | TABLE COMPARE |
| TIM | Geen | TIMER |
| TIMH | 15 | HIGHSPEED TIMER |
| WSFT (@) | 16 | WOORD SHIFT |
| XCHG (@) | 73 | DATA EXCHANGE |
| XFER (@) | 70 | BLOCK TRANSFER |
| XNRW (@) | 37 | EXCLUSIVE NOR |
| XORW (@) | 36 | EXCLUSIVE OR |

**Opmerking** Wanneer het programma ingevoerd is moet dit met een END(01) instructie op het laatste adres worden afgesloten.

## 4.6 Ladderdiagram instructies

Ladderdiagram instructies omvatten de logische instructies en logische blok instructies en worden gebruikt in de executiecondities in het ladderdiagram. Logische blok instructies worden gebruikt om complexere netwerken te kunnen programmeren.

### *4.6.1 LOAD, LOAD NOT, AND, AND NOT, OR en OR NOT*




**Beperkingen** Er is geen limiet aan het aantal maal dat deze instructies gebruikt kunnen worden noch aan de volgorde waarin ze gebruikt kunnen worden zolang als de geheugen capaciteit van de PLC niet wordt overschreden.

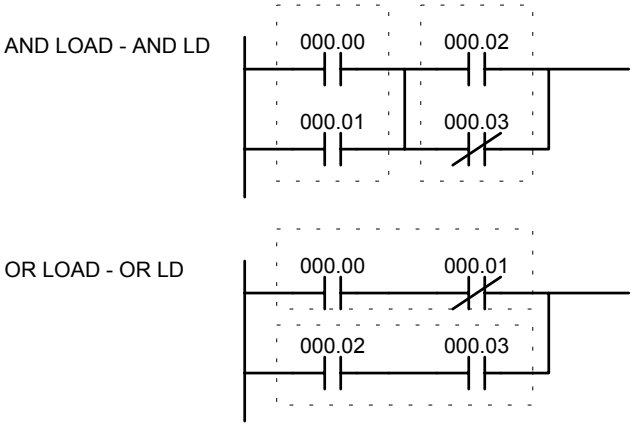
**Omschrijving** Deze zes basisinstructies vind u terug in de condities van het ladderdiagram. De status van de bits die toegewezen zijn aan deze instructies bepalen de executieconditie voor alle andere instructies in het ladderdiagram. Dit is uitgebreid

beschreven in "Programma uitvoer" op pagina 30. Elk van deze instructies en elk bitadres kan zo vaak gebruikt worden als nodig is in elk van de bovenstaande instructies.

De status van het bit operand (B) dat toegewezen is aan een LD of LD NOT bepaalde de eerste executieconditie. AND berekent de logische EN tussen de executieconditie en de status van het gebruikte operandbit; AND NOT berekent de logische EN tussen de executieconditie en de inverse status van de gebruikte operandbit. OR berekent de logische OF tussen de executieconditie en de status van het gebruikte operandbit; OR NOT berekent de logische OF tussen de executieconditie en de inverse status van de gebruikte operandbit. Gebruikte TR bits worden in een ladderdiagram niet weergegeven.

**Vlaggen** Er worden geen vlaggen beïnvloed door deze instructie.

### *4.6.2 AND LOAD en OR LOAD*




**Omschrijving** Wanneer instructies worden samengevoegd in blokken die niet logisch gecombineerd kunnen worden met alleen OR en AND bewerkingen moet AND LD of OR LD worden gebruikt. AND en OR instructies voeren een logische bewerking uit tussen de status van het gebruikte operandbit en de executieconditie. AND LD en OR LD voeren een logische bewerking uit tussen twee executiecondities, de actuele en de laatste ongebruikte.

Tijdens het invoeren van ladderdiagrammen is het niet noodzakelijk om AND LD en OR LD instructies te gebruiken. Deze instructies zijn echter wel noodzakelijk om het ladderdiagram programma om te zetten naar de mnemonic uitvoering (statement list) die uiteindelijk door de PLC verwerkt wordt.

Om het aantal gebruikte instructies in een programma te reduceren is een zeker begrip van de werking van logische blok instructies aan te raden. De procedures voor het invoeren, beperkingen en voorbeelden worden uitgelegd in hoofdstuk "Logische blok instructies" op pagina 12.

**Vlaggen** Er worden geen vlaggen beïnvloed door deze instructie.

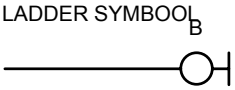
## 4.7 Bitcontrol instructies

Er zijn vijf instructies die gebruikt worden om de statussen van bits te manipuleren. deze instructies zijn OUT, OUT NOT, DIFU(13), DIFD(14) en KEEP(11). Deze instructies worden gebruikt om bits op verschillende manieren aan en uit te zetten. Voor de nieuwere PLC’s (CPM1(A), CQM1 en C200HS) zijn er twee instructies toegevoegd, SET en RSET.

### *4.7.1 Uitgangen en hulprelais aansturen - OUT en OUT NOT*

OUTPUT - OUT

LADDER SYMBOOL

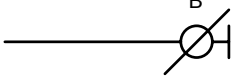



OPERAND DATAGEBIEDEN

| B; Bit |
|---|
| IR, AR, HR, TC, LR, TR |

OUTPUT NOT - OUT NOT

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| B; Bit |
|---|
| IR, AR, HR, TC, LR |

**Beperkingen** Het is aan te raden om elk outputbit slechts in één bitcontrol instructie te gebruiken. Raadpleeg de sectie "IR (interne relais) gebied" op pagina 62 voor details over dubbel gebruik van bits bij bitcontrol instructies.

**Omschrijving** OUT en OUT NOT worden gebruikt om de status van het aan de instructie toegewezen bit aan of uit te zetten, afhankelijk van status van de executieconditie.

OUT zet het toegewezen bit aan als de status van de executieconditie aan is en zet het toegewezen bit uit als de executieconditie uit is. Bij gebruik van TR bits wordt de OUT instructie op het aftak punt geplaatst in plaats van aan het einde van de instructieregel. Raadpleeg "Vertakkende instructie regels" op pagina 19 voor details.

OUT NOT zet het toegewezen bit uit als de status van de executieconditie aan is en aan als de executieconditie uit is.

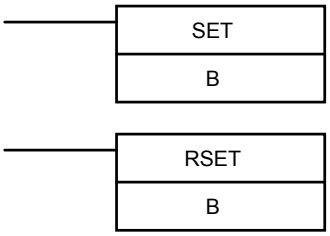
OUT en OUT NOT kunnen gebruikt worden om de uitvoer te beïnvloeden door de bits die toegewezen zijn aan condities in het ladderdiagram aan en uit te zetten, waardoor condities voor andere instructies veranderd kunnen worden. Dit maakt het mogelijk om een complex set van condities te gebruiken om de status van een enkel werkbit bepalen waarna dat werkbit kan worden gebruikt om andere instructies aan te sturen.

De tijd dat een bit aan of uit is kan worden bepaald door de OUT en OUT NOT instructies te combineren met TIM instructies. Voorbeelden hiervoor zijn opgenomen in "Timer - TIM" op pagina 87.

**Vlaggen** Er worden geen vlaggen beïnvloed door deze instructie.

### *4.7.2 Setten en resetten - SET en RSET*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| B: Bit |
|---|
| IR, AR, HR, LR |

| B: Bit |
|---|
| IR, AR, HR, LR |

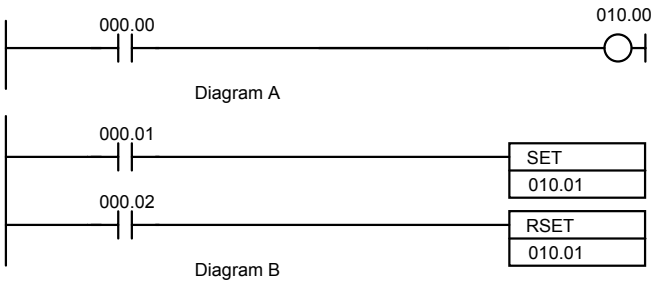
**Omschrijving** SET zet het operandbit aan wanneer de executieconditie aan is en beïnvloedt de status van het operandbit niet wanneer executieconditie uit is. RSET zet het operandbit uit wanneer de executieconditie aan is en beïnvloedt de status van het operandbit niet wanneer de executieconditie uit is.

De werking van SET verschilt van die van OUT omdat de OUT instructie het operandbit uit zet zolang de executieconditie uit is. Zo verschilt RSET ook van OUT NOT omdat OUT NOT het operandbit aan zet zolang de executieconditie uit is.

**Voorzorgen** De status van operandbits voor SET en RSET instructies geprogrammeerd tussen IL(02) en ILC(03) of JMP(04) en JME(05) verandert niet als de interlock- of sprong-conditie laag is.

**Vlaggen** Er worden geen vlaggen beïnvloed door deze instructie.

**Voorbeeld** Het volgende voorbeeld toont het verschil tussen OUT en SET/RSET. In het eerste netwerk (diagram A), wordt de status (aan of uit) van 010.00 direct bepaald door 000.00. In het tweede netwerk (diagram B), gaat 010.01 aan als 000.01 aan gaat en blijft aan (zelfs als 000.01 uit gaat) tot 000.02 aan gaat.

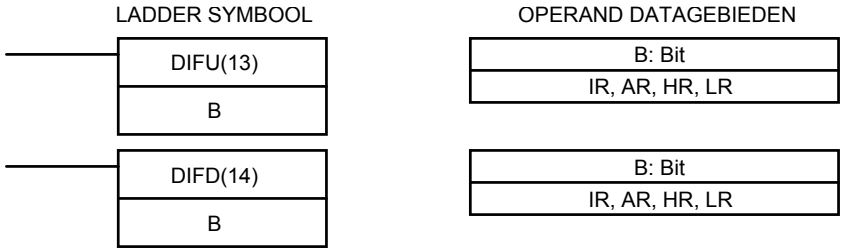



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | OUT | 010.00 |

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.01 |
| 00001 | SET | 010.01 |
| 00002 | LD | 000.02 |
| 00003 | RSET | 010.01 |

**Opmerking** Inplaats van SET en RSET kan ook de instructie KEEP(11) gebruikt worden (zie pagina 80). Bij gebruik van de instructie KEEP(11) moet binnen één netwerk zowel de SET als de RESET conditie bepaald worden. Dit bevordert de leesbaarheid van het programma.

### *4.7.3 Op- en neergaande flanken - DIFU(13) en DIFD(14)*




**Beperkingen** Het is aan te raden om elk outputbit slechts in één bitcontrol instructie te gebruiken. Raadpleeg de sectie "IR (interne relais) gebied" op pagina 62 voor details over dubbel gebruik van bits bij bitcontrol instructies.

**Omschrijving** DIFU(13) en DIFD(14) worden gebruikt om het toegewezen bit voor één scan aan te zetten.

Wanneer uitgevoerd, vergelijkt DIFU(13) de actuele executieconditie met de vorige executieconditie. Als de vorige executieconditie uit was en de actuele aan is, dan zal DIFU(13) het toegewezen bit aan zetten. Als de vorige executieconditie aan was en de actuele executieconditie is aan of uit, dan zal DIFU(13) het toegewezen bit uit zetten of het uit laten als het bit al uit was. Het toegewezen bit zal dus nooit langer dan één scan aan zijn (ervan uitgaande dat de instructie elke scan wordt uitgevoerd).

Wanneer uitgevoerd, vergelijkt DIFD(14) de actuele executieconditie met de vorige executieconditie. Als de vorige executieconditie aan was en de actuele uit is, dan zal DIFD(14) het toegewezen bit aan zetten. Als de vorige executieconditie uit was en de actuele executieconditie is aan of uit, dan zal DIFD(14) het toegewezen bit uit zetten of het uit laten als het bit al uit was. Het toegewezen bit zal dus nooit langer dan één scan aan zijn (ervan uitgaande dat de instructie elke scan wordt uitgevoerd).

Deze instructies worden gebruikt wanneer de gedifferentieerde variant van een bepaalde instructie (d.w.z. die met een @ als prefix) niet beschikbaar is en toch een eenmalige (één scan) uitvoering van de instructie gewenst is. Ze kunnen ook gebruikt worden bij de niet-gedifferentieerde uitvoering van instructies die wel een gedifferentieerde uitvoering hebben om het programma te vereenvoudigen. Voorbeelden hiervan worden hieronder getoond.

**Vlaggen** Er worden geen vlaggen beïnvloed door deze instructie.

**Voorzorgen** De werking van DIFU(13) en DIFD(14) kan onzeker zijn als de instructies tussen IL en ILC, tussen JMP en JME of in subroutines geprogrammeerd zijn. Raadpleeg pagina 82 en 84 hierover. Aangezien de flankdetectie (DIFU(13) en DIFD(14))

instructies de actuele executieconditie vergelijken met de vorige moet bekend zijn wat de vorige executieconditie was. Tijdens de eerste scan is niet bekend wat de vorige executieconditie was, dus worden eventuele flanken niet gedetecteerd. Bij gebruikt van interlocks, sprongen en subroutines wordt voor de flank detectie gekeken naar de vorige keer dat het programma tussen IL en ILC, tussen JMP en JME of in de subroutine werd uitgevoerd

**Voorbeeld 1: Wanneer geen gedifferentieerde instructie gebruikt word**

In diagram A hieronder zal de CMP (20) instructie, zodra deze wordt uitgevoerd met een aan executieconditie, de inhoud van de twee operand woorden (HR10 en DM0000) vergelijken en afhankelijk hiervan de rekenkundige vlaggen (GR, EQ en LE) aan of uit zetten. Als de executieconditie aan blijft, kunnen de statussen van deze vlaggen elke scan veranderen als de inhoud van één of beide operands verandert. Diagram B echter, geeft een voorbeeld van hoe DIFU(13) kan worden gebruikt om er zeker van te zijn dat CMP(20) alleen wordt uitgevoerd als de executieconditie aan gaat (op de opgaande flank).

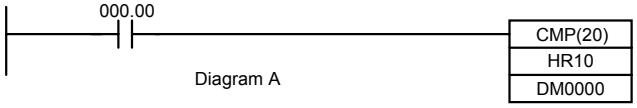



Diagram A

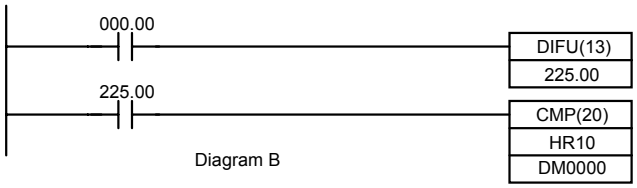


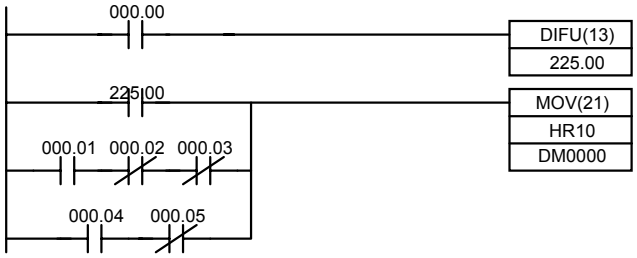

Diagram B

| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | CMP(20) | HR10 | DM0000 |

| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | DIFU(13) | 225.00 | |
| 00002 | LD | 225.00 | |
| 00003 | CMP(20) | HR10 | DM0000 |

**Voorbeeld 2: Vereenvoudigen van een programma**

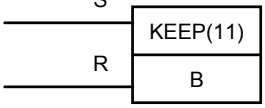
Alhoewel een gedifferentieerde uitvoering van MOV(21) beschikbaar is, is het niet mogelijk het onderstaande diagram te tekenen met deze uitvoering omdat maar één van de condities in de executieconditie voor de MOV(21) gedifferentieerd uitgevoerd is.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | DIFU(13) | 225.00 | |
| 00002 | LD | 225.00 | |
| 00003 | LD | 000.01 | |
| 00004 | AND NOT | 000.02 | |
| 00005 | AND NOT | 000.03 | |
| 00006 | OR LD | | |
| 00007 | LD | 000.04 | |
| 00008 | AND NOT | 000.05 | |
| 00009 | OR LD | | |
| 00010 | MOV(21) | HR10 | DM0000 |

### *4.7.4 Status vasthouden - KEEP(11)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| B: Bit |
|---|
| IR, AR, HR, LR |

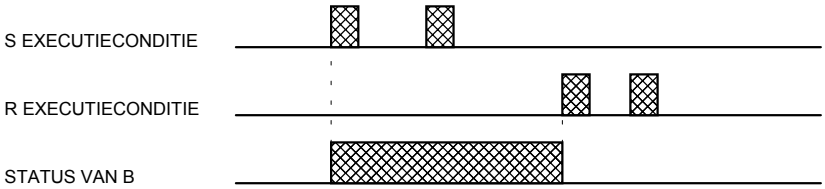
**Beperkingen**

Het is aan te raden om elk outputbit slechts in één bitcontrol instructie te gebruiken. Raadpleeg de sectie "IR (interne relais) gebied" op pagina 62 voor details over dubbel gebruik van bits bij bitcontrol instructies.

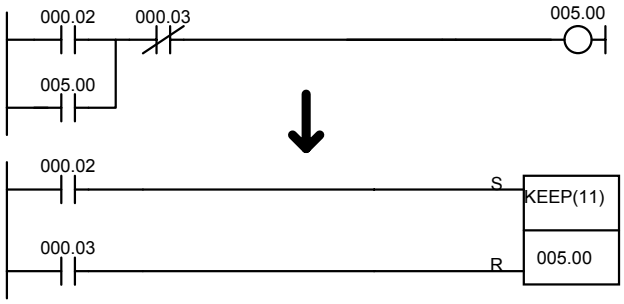
**Omschrijving**

KEEP(11) wordt gebruikt om de status van het toegewezen bit vast te houden afhankelijk van twee executiecondities. Deze executiecondities hebben in de bovenstaande figuur de labels S en R. S is de set input; R de reset input. KEEP(11) werkt als een latchrelais dat wordt geset door S en gereset door R.

Wanneer S aangaat zal het toegewezen bit aangaan en aanblijven tot het gereset wordt, onafhankelijk of S aanblijft of uitgaat. Wanneer R aangaat zal het toegewezen bit uitgaan en uitblijven, onafhankelijk of R aanblijft of uitgaat. De relatie tussen de executiecondities en de KEEP(11) bit status wordt hieronder getoond.




KEEP(11) werkt zoals de zelfhandhavende bits zoals beschreven in “zelfhandhavende bits” op pagina 26. De volgende twee netwerken hebben dezelfde functie. Eén van de twee netwerken gebruikt de KEEP(11) instructie en gebruikt één instructie minder dan het andere netwerk. Bij gebruik van een KEEP instructie houden bits hun status vast, zelfs als het netwerk geprogrammeerd wordt tussen en IL en ILC instructie waarvan de executieconditie laag is.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.02 |
| 00001 | OR | 005.00 |
| 00002 | AND NOT | 000.03 |
| 00003 | OUT | 005.00 |

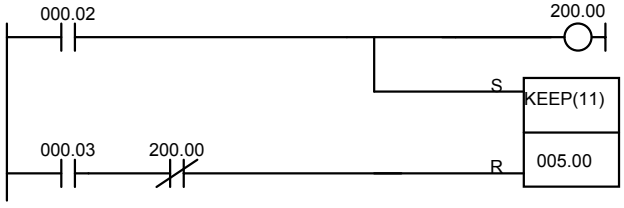
| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.02 |
| 00001 | LD | 000.03 |
| 00002 | KEEP(11) | 005.00 |

**Opmerking**

Het voordeel van het gebruik van KEEP(11) in plaats van logica om een “houd circuit” te creëren is dat bij het gebruik van een KEEP(11) functie in één oogopslag duidelijk is dat het netwerk een houd functie bevat. Bij gebruik van een overname contact (zoals hierboven met logica gerealiseerd is) moet het netwerk meer geïnterpreteerd worden.

**Prioriteit**

Bij de KEEP(11) instructie heeft de reset prioriteit. D.w.z. dat als de reset ingang aan is, het bij KEEP(11) gebruikte operandbit altijd uit is. Een eenvoudige manier om de set prioriteit te geven bij een KEEP(11) instructie is hieronder getoond. Door het extra bit dat met een OUT instructie aan de SET conditie hangt in de reset op te nemen wordt deze geblokkeerd op het moment dat de SET aan is.

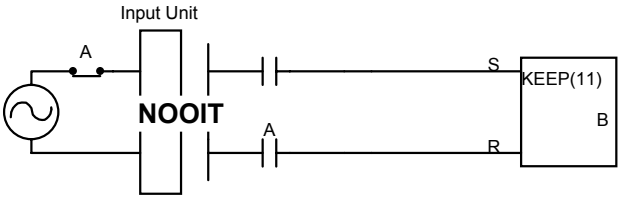



**Vlaggen**

Er worden geen vlaggen beïnvloed door deze instructie.

**Voorzorgen**

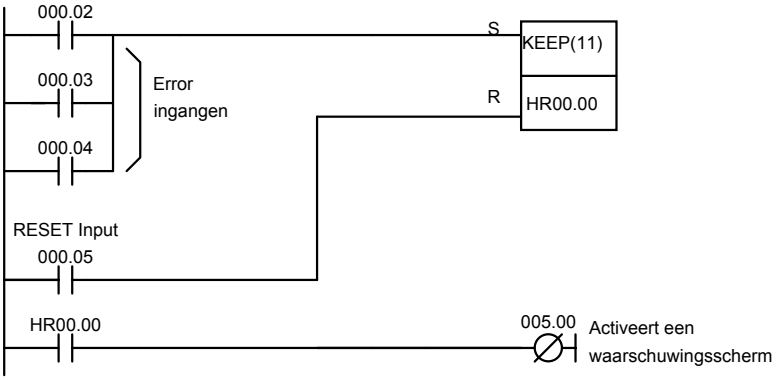
Gebruik nooit een input bit van een NC (normally closed) input op de reset conditie van een KEEP(11) instructie wanneer het input device een AC powersupply gebruikt. De vertraging in het uitgaan van de PLC's DC powersupply (relatief ten opzichte van de AC powersupply op het input device) kan ervoor zorgen dat het aan de KEEP(11) instructie toegewezen bit wordt gereset wanneer de besturing wordt uitgezet. Deze situatie wordt hieronder getoond.




Bits die gebruikt zijn als operandbit voor een KEEP(11) instructie worden niet gereset door interlocks. Raadpleeg pagina 82 voor details.

**Voorbeeld**

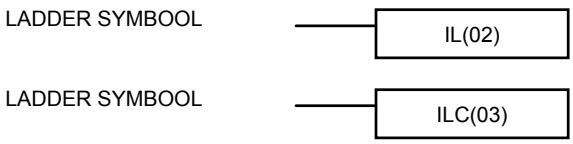
Als een AR of HR bit wordt gebruikt, wordt de status van dit bit vastgehouden, zelfs als de spanning van de CPU wegvalt. KEEP(11) kan dus gebruikt worden om bits te programmeren die de status onthouden die ze hadden toen de PLC werd uitgezet. Een voorbeeld hiervan kan een waarschuwingsscherm zijn dat op een terminal getoond moet worden als een installatie opgestart wordt nadat het, vanwege een noodsituatie, op een abnormale manier is afgesloten. De bits 000.02, 000.03 en 000.04 geven aan dat een bepaald type error is voorgekomen. Bit 000.05 wordt aangezet om het waarschuwingsscherm te resetten. HR00.00, die aangaat als één van de error ingangen aangaat, geeft aan dat een error situatie is voorgekomen en wordt gebruikt om het waarschuwingsscherm via 005.00 op te roepen.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.02 |
| 00001 | OR | 000.03 |
| 00002 | OR | 000.04 |
| 00003 | LD | 000.05 |
| 00004 | KEEP(11) | HR00.00 |
| 00005 | LD | HR00.00 |
| 00006 | OUT | 005.00 |

KEEP(11) kan ook gebruikt worden in combinatie met TIM om bits te produceren die bepaalde vertragingen hebben in het aan en uitgaan. Raadpleeg hiervoor "Timer - TIM" op pagina 87.

## 4.8 Interlocks - IL(02) en ILC(03)




**Omschrijving**

IL(02) wordt altijd gebruikt in combinatie met ILC(03) om interlocks in een programma te creëren. Interlocks worden gebruikt om, zoals bij gebruik van TR relais, aftakkingen te creëren. Echter, wanneer de executieconditie voor IL(02) uit is, verschilt de werking van instructies tussen IL(02) en ILC(03) van vertakkingen die met TR bits zijn geprogrammeerd. Als de executieconditie van IL(02) aan is, wordt het programma uitgevoerd zoals het is geschreven, met een *aan*

executieconditie gebruikt om elke instructie regel te starten van het punt waar IL(02) is geplaatst tot de volgende ILC(03). Raadpleeg "Vertakkende instructie regels" op pagina 19 voor basis beschrijvingen van beide methoden.

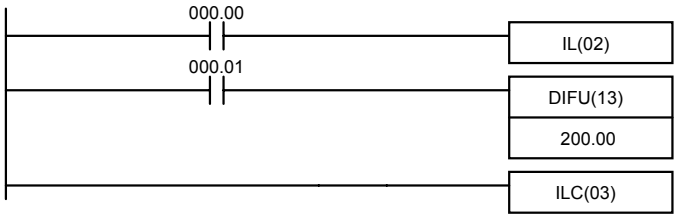
Als de executieconditie voor IL(02) uit is, zullen de instructies in de sectie tussen IL(02) en ILC(03) worden behandeld zoals in de volgende tabel is getoond:

| Instructie | Behandeling |
|---|---|
| OUT en OUT NOT | Toegewezen bit gaat uit. |
| TIM en TIMH(15) | Wordt gereset. |
| CNT en CNTR(12) | Actuele waarde wordt gehandhaafd. |
| KEEP(11) | Bit status wordt gehandhaafd. |
| DIFU(13) en DIFD(14) | Niet uitgevoerd (zie hieronder). |
| Alle andere | Niet uitgevoerd. |

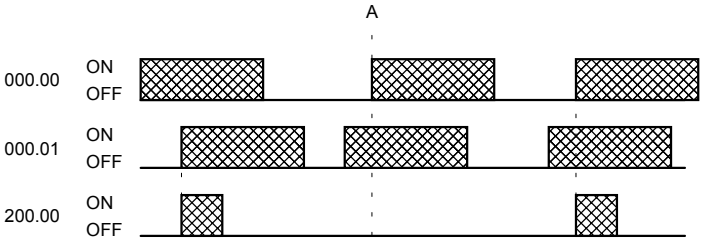
IL(02) en ILC(03) hoeven niet noodzakelijkerwijs in paren gebruikt te worden. IL(02) kan diverse keren achter elkaar worden gebruikt, waarbij elke IL(02) een interlock sectie creëert tot de volgende ILC(03). ILC(03) kan niet worden gebruikt tot er minimaal één IL(02) is ingevoerd tussen deze ILC(03) en een voorgaande ILC(03) of het begin van het programma.

**DIFU(13) en DIFD(14) in interlocks**

Veranderingen in de executieconditie voor een DIFU(13) of DIFD(14) instructie worden niet opgeslagen als de DIFU(13) of DIFD(14) is geprogrammeerd tussen een IL(02) en ILC(03) en de executieconditie voor de IL(02) is uit. Wanneer DIFU(13) of DIFD(14) wordt uitgevoerd in een interlock sectie direct nadat de executieconditie voor de IL(02) is aangegaan, dan zal de executieconditie voor de DIFU(13) of DIFD(14) worden vergeleken met de executieconditie die bestond voordat de interlock actief werd (d.w.z. voordat de interlock conditie voor IL(02) uitging). Het ladderdiagram en de bitstatus veranderingen hiervoor zijn hieronder getoond. De interlock is in werking als 000.00 uit is. Merk op dat 200.00 niet aangaat op punt A ook al is 000.01 uit en vervolgens weer aangegaan.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | IL(02) | |
| 00002 | LD | 000.01 |
| 00003 | DIFU(13) | 200.00 |
| 00004 | ILC(03) | |




**Voorzorgen**

Er moet minimaal één ILC (03) geprogrammeerd worden na één of meerdere IL(02) instructies. Alhoewel zoveel IL(02) instructies als nodig kunnen worden gebruikt met één ILC(03), kunnen ILC(03) instructies niet worden gebruikt zonder minimaal één IL(02) ervoor, d.w.z. dat nesten niet mogelijk is. Zodra een ILC(03) wordt uitgevoerd worden alle interlocks tussen de actieve ILC(03) en de voorgaande ILC(03) gewist.

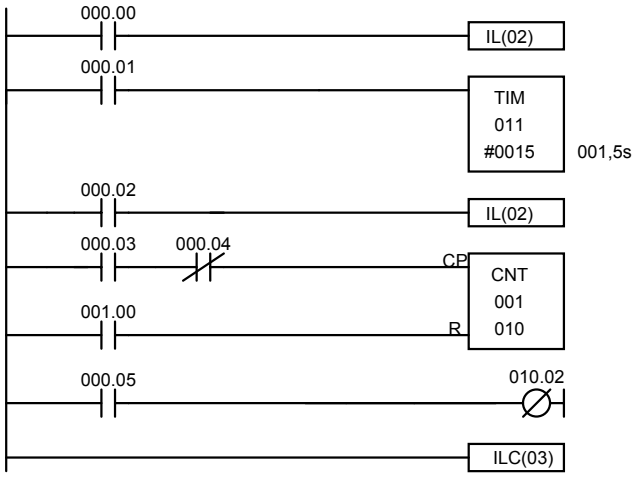
Wanneer meer dat één IL(02) wordt gebruikt met een enkele ILC(03), verschijnt een error boodschap als de program check wordt uitgevoerd. Het programma wordt echter normaal uitgevoerd.

**Vlaggen**

Er worden geen vlaggen beïnvloed door deze instructie.

**Voorbeeld**

Het volgende voorbeeld toont IL(02) die tweemaal gebruikt wordt voor een ILC(03).

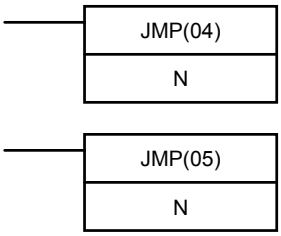



| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | IL(02) | | |
| 00002 | LD | 000.01 | |
| 00003 | TIM | 011 | #0015 |
| 00004 | LD | 000.02 | |
| 00005 | IL(02) | | |
| 00006 | LD | 000.03 | |
| 00007 | AND NOT | 000.04 | |
| 00008 | LD 001.00 | | |
| 00009 | CNT | 001 | 010 |
| 00010 | LD | 000.05 | |
| 00011 | OUT | 010.02 | |
| 00012 | ILC(03) | | |

Wanneer de executieconditie voor de eerste IL(02) uit is, zal TIM011 worden gereset naar 1.5 seconde, CNT001 zal niet veranderen en 010.02 wordt uitgezet. Wanneer de executieconditie voor de eerste IL(02) aan is en de executieconditie voor de tweede IL(02) uit is, zal TIM011 worden uitgevoerd afhankelijk van de status van 000.01, CNT001 verandert niet en 010.02 wordt uitgezet. Wanneer de executieconditie voor beide IL(02)'s aan zijn wordt het programma normaal uitgevoerd.

## 4.9 Springen - JMP(04) en JME(05)

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| N: Sprongnummer |
|---|
| # (00 to 49) |

| N: Sprongnummer |
|---|
| # (00 to 49) |

**Beperkingen**

Sprongnummers 01 t/m 49 mogen maar één keer gebruikt worden in een JMP(04) en één keer in een JME(05) instructie. D.w.z. dat elk nummer gebruikt kan worden om één sprong te definiëren. Sprong nummer 00 kan zo vaak als gewenst gebruikt worden.

**Omschrijving**

JMP(04) wordt altijd in combinatie met JME(05) gebruikt om sprongen te definiëren, d.w.z. om direct van het ene punt naar het andere punt in een ladderdiagram te gaan. JMP(04) definieert het punt waar de sprong gemaakt wordt, JME(05) definieert de bestemming van de sprong. Wanneer de executieconditie voor JMP(04) aan is, wordt geen sprong gemaakt en wordt het programma vervolgd bij de instructie achter de JMP(04) instructie. Wanneer de executieconditie voor JMP(04) uit is, wordt een sprong gemaakt naar de JME(05) instructie met hetzelfde nummer en wordt het programma vervolgd met de instructie die volgt op de JME(05) instructie.

Als het sprongnummer voor JMP(04) tussen 01 en 49 ligt worden sprongen, wanneer gemaakt, direct uitgevoerd naar de JME(05) met hetzelfde sprongnummer zonder dat enige instructie tussen de JMP(04) en JME(05) wordt

uitgevoerd. De status van timers, counters, bits gebruikt met OUT, bits gebruikt met OUT NOT en alle andere statusbits aangestuurd door de instructies tussen de JMP(04) en JMP(05) zullen niet veranderen. Elk van de sprongnummers tussen 01 en 99 kan worden gebruikt om één sprong te definiëren. Omdat alle instructies tussen de JMP(04) en JME(05) worden overgeslagen kunnen de sprongnummers 01 t/m 99 worden gebruikt om de cyclustijd te reduceren.

Als het sprongnummer voor JMP(04) 00 is, zal de CPU zoeken naar de volgende JME(05) met een sprongnummer 00. Om dit te doen moet deze door het programma zoeken. Dit veroorzaakt een langere cyclustijd (wanneer de executieconditie uit is) dan bij de andere sprongen. De status van timers, counters, bits gebruikt in OUT, bits gebruikt in OUT NOT en alle andere statussen aangestuurd door instructies tussen de JMP(04) 00 en JMP(05) 00 worden niet veranderd. Sprongnummer 00 kan zo vaak als gewenst gebruikt worden. Een sprong van JMP(04) 00 gaat altijd naar de volgende JME(05) 00 in het programma. Het is dus mogelijk om een aantal maal achtereenvolgend JMP(04) 00 te gebruiken om naar dezelfde JME(05) 00 te springen. Het heeft echter geen zin om een reeks JME(05) 00 instructies te programmeren, omdat alle sprongen naar JME(05) 00 zullen eindigen bij de eerste die gevonden wordt. Aangezien bij JMP(04) 00 wordt gesprongen naar de volgende JME(05) 00 in het programma is het niet mogelijk om met JMP(04) 00 terug te springen in het programma.

**DIFU(13) en DIFD(14) in sprongen**

Alhoewel DIFU(13) en DIFD(14) zijn gemaakt om het toegewezen bit voor één scan aan te zetten, hoeven zij dit niet noodzakelijkerwijs te doen wanneer ze geprogrammeerd zijn tussen een JMP(04) en JME(05). Zodra DIFU(13) of DIFD(14) een bit heeft aangezet, zal het bit aanblijven tot de volgende keer dat DIFU(13) of DIFD(14) wordt uitgevoerd. In het normale programma houdt dit de volgende scan in. Bij een sprong betekent dit de volgende keer dat de sprong van JMP(04) naar JME(05) niet is gemaakt. D.w.z. dat wanneer een bit wordt aangezet door DIFU(13) of DIFD(14) en vervolgens wordt, in de volgende scan, een sprong gemaakt zodat DIFU(13) of DIFD(14) wordt overgeslagen, dan zal het toegewezen bit aan blijven tot de executieconditie voor de JMP(04) die de sprong beheerst aan gaat.

**Voorzorgen**

Wanneer na de JMP(04) geen JME(05) met hetzelfde nummer gebruikt is zal een error melding getoond worden wanneer de program check wordt uitgevoerd. Ook wanneer de JME(05) instructie voor de JMP(04) instructie wordt geplaatst wordt deze melding gegenereerd, het programma wordt in dit geval normaal uitgevoerd. Deze error boodschap verschijnt ook als er voor een JME(05) 00 meermaals JMP(04) 00 is geplaatst. Ook in dit geval zal het programma normaal worden uitgevoerd.

**Vlaggen**

Er worden geen vlaggen beïnvloed door deze instructie.

**Voorbeelden**

Voorbeelden van programma's met sprongen worden getoond in "Springen" op pagina 23.

## 4.10 Programma einde - END(01)

LADDER SYMBOOL


**Omschrijving**

END(01) is nodig als de laatste instructie in een programma. Wanneer een programma is voorzien van subroutines, dan wordt de END(01) geplaatst achter de laatste subroutine. Geen enkele instructie achter de END(01) zal worden uitgevoerd. END(01) kan overal in het programma geplaatst worden om alle instructies tot aan dat punt uit te voeren. Dit wordt soms gedaan om fouten te kunnen vinden in een programma. Deze extra END(01) instructie moet echter weer verwijderd worden om de rest van het programma uit te voeren.

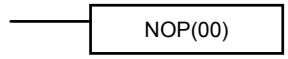
Als er geen END(01) in het programma is gezet, wordt geen instructie uitgevoerd en verschijnt de errormelding "no END instruction". SYSWIN voegt automatisch een END(01) instructie toe tijdens het downloaden als u deze bent vergeten. In dit geval betekent deze melding vaak dat u een te groot programma in het geheugen van de PLC heeft geladen.

**Vlaggen**

END(01) zet de ER, CY, GR, EQ en LE vlaggen uit.

## 4.11 No operation - NOP(00)

LADDER SYMBOOL




**Omschrijving** De instructie NOP(00) heeft geen functie. Wanneer NOP(00) wordt gevonden in een programma wordt niets gedaan en gaat de CPU verder met de volgende instructie in het programma. Wanneer het geheugen wordt gewist voor het downloaden wordt een NOP(00) geplaatst op alle adressen.

**Vlaggen** Er worden geen vlaggen beïnvloed door deze instructie.

## 4.12 Timer en counter instructies

TIM en TIMH zijn aftellende opkomtijd vertragende timer instructies die een TC nummer en een ingestelde waarde nodig hebben.

CNT is een aftellende counter instructie en CNTR is een omkeerbare counter instructie. Beide gebruiken een TC nummer en een ingestelde waarde. Beide gebruiken eveneens meerdere executiecondities voor tel en reset signalen.

Elk TC nummer kan maar één keer gebruikt worden, d.w.z. zodra het is gebruikt als definer in één van de timer of counter instructies, kan het niet opnieuw gebruikt worden. Zodra gedefinieerd, kunnen TC nummers zo vaak als nodig gebruikt worden als operands in instructies anders dan timer en counter instructies.

TC nummers lopen van 000 t/m 127. Bij het gebruik van een TC nummer als definer in een timer of counter instructie moet de prefix niet ingevoerd worden. Zodra gedefinieerd als een timer kan een TC nummer voorafgegaan door een prefix TIM gebruikt worden als operand in diverse instructies. De TIM prefix wordt gebruikt, onafhankelijk van de timerinstructie die gebruikt is om de timer te definiëren. Zodra gedefinieerd als een counter kan het TC nummer voorafgegaan door de prefix CNT gebruikt worden als operand in bepaalde instructies. De prefix CNT wordt ook gebruikt, onafhankelijk van de counter instructie die was gebruikt om de counter te definiëren.

De TIM en CNT prefixen worden gebruikt om een veld in het TC gebied aan te duiden. Het is mogelijk om met de TIM prefix een veld op te geven dat door een counter gebruikt wordt. In dit geval zal de counter waarde getoond worden. Alhoewel het verder geen consequenties heeft voor de uitvoer van het programma is het voor het overzicht in het programma niet aan te raden om TIM en CNT prefixen om te wisselen.

TC nummers kunnen worden gebruikt voor operands die bitdata willen hebben en voor operands die woorddata gebruiken. Wordt een TC nummer gebruikt in een instructie op bitniveau dan geeft het nummer toegang tot de completionvlag van de timer of counter. Wanneer een TC nummer wordt gebruikt in een instructie die woorddata verlangt dan geeft het TC nummer toegang tot een geheugenlocatie die de actuele waarde van de timer of counter bevat. De actuele waarde van een timer of counter kan dus gebruikt worden als operand in een CMP(20) of een andere instructie die gebruik van het TC gebied toestaat.

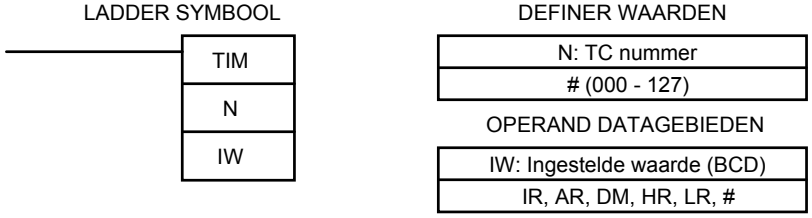
Het TC gebied bewaart de ingestelde waarden van timers en counters tijdens spanningsuitval. De actuele waarden van timers worden gereset wanneer de PLC begint met de uitvoer van het programma. Raadpleeg de sectie over interlocks voor details over de werking van timers en counters in interlock circuits. De actuele waarde van counters worden niet gereset bij spanningsuitval of het starten van het programma.

**Opmerking** TIM000 wordt gebruikt tijdens het programmeren om drie zaken aan te duiden. De timer met het TC nummer 000, de completionvlag van deze timer en de actuele waarde van deze timer. De samenhang zal duidelijk zijn. De eerste is altijd een instructie, de tweede is altijd een bit gebruikt in een instructie en de derde is altijd een woord gebruikt in een functie. Dit geldt voor alle TC nummers, onafhankelijk of de prefix TIM of CNT is.

Een ingestelde waarde kan ingevoerd worden als een constante of als een woord-adres in een datagebied. Als een woord uit het IR gebied, dat toegewezen is aan een input unit, wordt toegewezen als de ingestelde waarde kan de input unit dusdanig aangesloten worden dat de ingestelde waarde kan worden ingevoerd

met bijvoorbeeld duimwielschakelaars. Timers en counters die op deze manier zijn aangesloten kunnen alleen ingesteld worden als de PLC in de RUN of MONITOR mode staat. Alle ingestelde waarden (inclusief extern ingestelde waarden) moeten worden opgegeven in BCD.

### *4.12.1 Timer - TIM*




**Beperkingen**

De ingestelde waarde ligt tussen 000,0 en 999,9 seconde. De decimale punt wordt niet ingevoerd. Bij het invoeren van een ingestelde waarde van 0001 ligt de tijd tussen de 0 en 0,1 seconde.

Elk TC nummer kan gebruikt worden als definer in één timer of counter instructie.

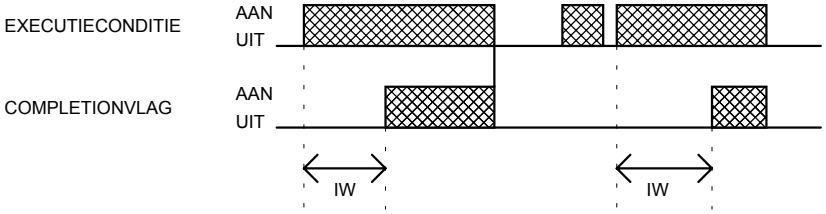
TC000 t/m TC003 kunnen beter niet gebruikt worden voor TIM, aangezien deze nummers bij gebruik van TIMH(15) instructies een betere nauwkeurigheid geven. Worden ze niet gebruikt voor TIMH(15) instructies dan kunnen ze echter voor gewone TIM instructies gebruikt worden.

**Omschrijving**

Een timer wordt geactiveerd wanneer de executieconditie aan gaat en wordt gereset (naar de ingestelde waarde) als de executieconditie uit gaat. De TIM instructie meet de tijd in units van 0,1 seconde. Een ingestelde waarde van #10 betekent dus een tijd van 1,0 seconden. De actuele waarde van een timer telt af van de ingestelde waarde (in units van 0,1s) naar 0.

Als de executieconditie lang genoeg aan blijft voor de TIM instructie om terug te tellen naar nul, dan zal de completionvlag van het gebruikte TC nummer aan gaan en aan blijven tot de timer wordt gereset (d.w.z. tot de executieconditie uitgaat).

Het volgende figuur toont de relatie tussen de executieconditie voor TIM en de eraan toegewezen completionvlag.




**Voorzorgen**

Timers in een interlock worden gereset als de executieconditie voor IL (02) uit is. Door het uitzetten van de PLC worden timers eveneens gereset. Als een timer gewenst is die onder deze condities niet gereset wordt, kunnen klokpulsen uit het SR gebied gebruikt worden om timers te maken met CNT instructies. Raadpleeg "counter - cnt" op pagina 93 voor details.

Programma uitvoer wordt voortgezet, zelfs als niet BCD ingestelde waarden worden gebruikt. De tijden worden hierdoor echter onbetrouwbaar.

**Vlaggen**

**ER**: IW (ingestelde waarde) is niet in BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

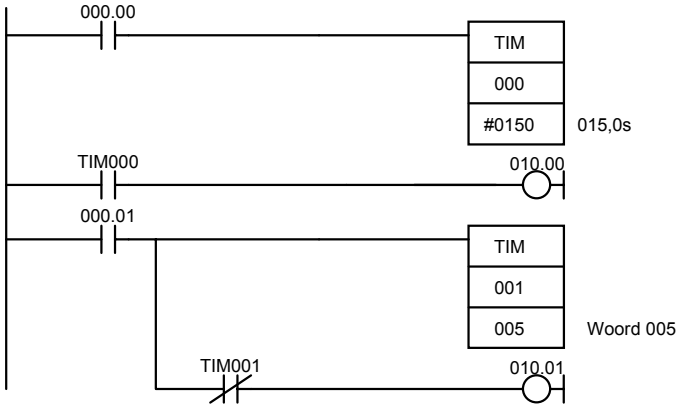
**Voorbeelden**

Alle hier volgende voorbeelden gebruiken OUT instructies met operandbits uit het IR gebied om het resultaat van timers te verwerken. In plaats van deze OUT instructies kunnen natuurlijk ook andere instructies toegepast worden.

**Voorbeeld1: Basis toepassing**

Het volgende voorbeeld toont twee timers, de ingestelde waarde van de ene wordt opgegeven met een constante, de andere via inputwoord 005. De eerste timer zal gaan lopen als 000.00 aangaat. Na 15 seconden loopt de timer af en zal 010.00 aan gaan. Wanneer 000.00 uitgaat zal de timer gereset worden en wordt 010.00 uitgezet. Wordt de timer gereset voordat deze is afgelopen dan zal 010.00 niet aangaan. Wanneer 000.01 aangaat wordt TIM001 gestart. Deze begint te tellen met de waarde die op het IR woord 005 staat. Bit 010.01 wordt aangezet als

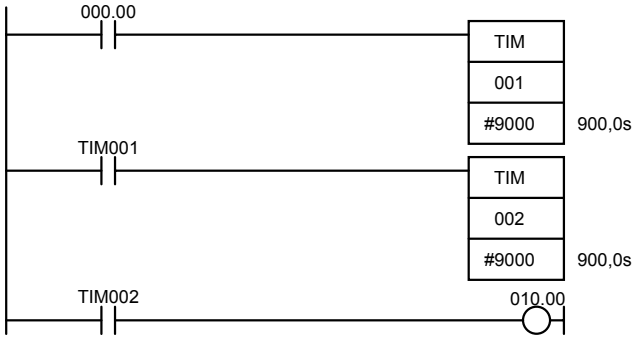
000.01 aan gaat. Wanneer de ingestelde waarde uit woord 005 is afgelopen dan wordt 010.01 uitgezet. Dit bit wordt ook uitgezet als TIM001 wordt gereset, onafhankelijk of deze al is afgelopen of niet.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | TIM | 000 | #0150 |
| 00002 | LD | TIM000 | |
| 00003 | OUT | 010.00 | |
| 00004 | LD | 000.01 | |
| 00005 | TIM | 001 | 005 |
| 00006 | AND NOT | TIM001 | |
| 00007 | OUT | 010.01 | |

**Voorbeeld 2: Verlengde timers**

Er zijn twee manieren om timers te creëren met een tijd van meer dan 999,9 seconden. Eén manier is om meer dan één timer te gebruiken, de completionvlag van elke timer wordt gebruikt om de volgende te activeren. Een eenvoudig voorbeeld staat hieronder, hier worden twee 900.0 seconde (15 minuten) timers gecombineerd om functioneel een 30 minuten timer te creëren.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | TIM | 001 | #9000 |
| 00002 | LD | TIM001 | |
| 00003 | TIM | 002 | #9000 |
| 00004 | LD | TIM002 | |
| 00005 | OUT | 010.00 | |

In dit voorbeeld wordt 010.00 30 minuten nadat 000.00 is aangegaan aangezet.

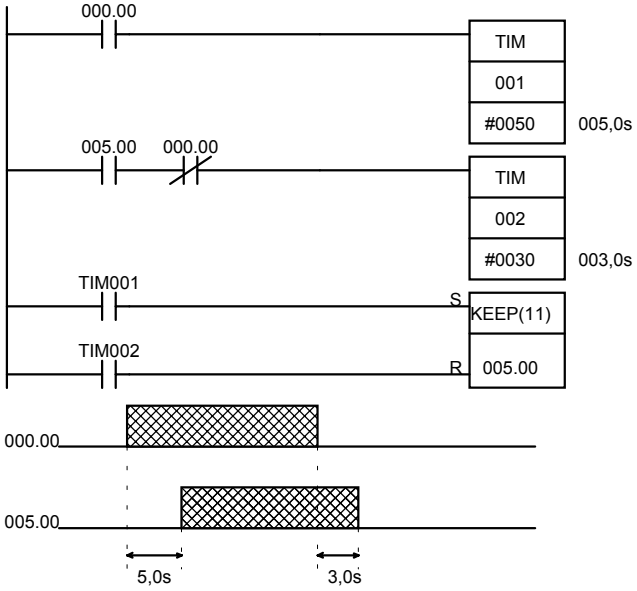
TIM kan ook gecombineerd met CNT gebruikt worden of CNT kan worden gebruikt in combinatie met klokpuls bits uit het SR gebied om langere timers te creëren. Een voorbeeld hiervoor wordt gegeven in "counter - cnt" op pagina 93.

**Voorbeeld 3: Aan / uit vertragingen**

TIM kan in combinatie met KEEP(11) worden gebruikt om een vertraging in het aan- en uitzetten van een bit te creëren afhankelijk van een gewenste executieconditie.

De completionvlaggen van twee timers worden gebruikt in de executiecondities voor het setten en het resetten van de KEEP(11) instructie. Op deze manier is een aan- en een uitvertraging te creëren. Het bit wiens werking vertraagd dient te worden, wordt als operand in de KEEP(11) gebruikt. Het aan- en uitgaan van het toegewezen bit wordt dus vertraagd door twee ingestelde waarden van twee timers. De twee ingestelde waarden kunnen natuurlijk, als dat gewenst is, hetzelfde zijn.

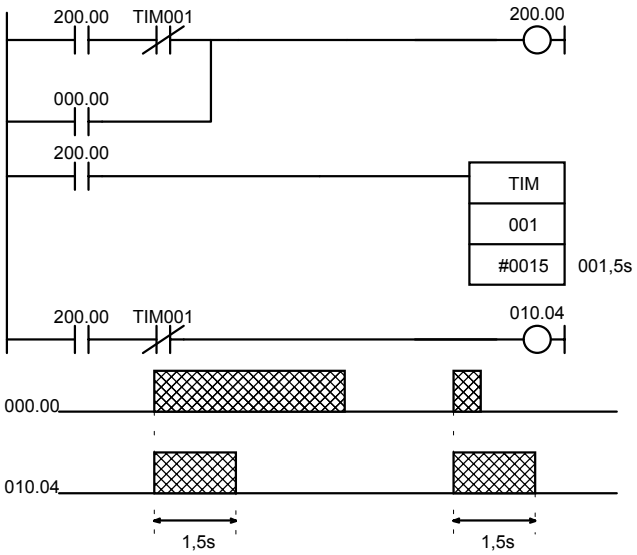
In het volgende voorbeeld wordt 005.00 5,0 seconden nadat 000.00 aangaat geset en 3,0 seconden nadat 000.00 uitgaat gereset. Het is noodzakelijk om zowel 005.00 als 000.00 te gebruiken in de executieconditie voor TIM002; 000.00 in een geïnverteerde conditie is nodig om TIM002 te resetten als 000.00 aangaat en 005.00 is nodig om TIM002 te activeren (als 000.00 uit is).




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | TIM | 001 | #0050 |
| 00002 | LD | 005.00 | |
| 00003 | AND NOT | 000.00 | |
| 00004 | TIM | 002 | #0030 |
| 00005 | LD | TIM001 | |
| 00006 | LD | TIM002 | |
| 00007 | KEEP(11) | 005.00 | |

**Voorbeeld 4: One-shot bits**

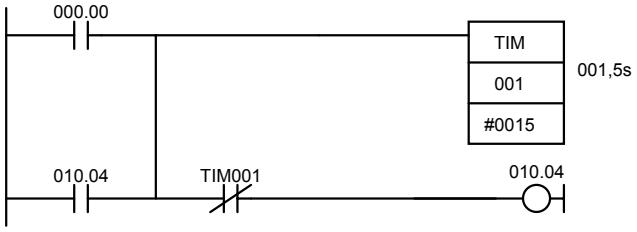
De tijd dat een bit aan of uit is kan worden ingesteld door TIM te gebruiken met OUT of OUT NOT. Het volgende diagram toont hoe dit mogelijk is. In dit voorbeeld blijft 010.04 aan voor 1,5 seconde nadat 000.00 aan is gegaan, onafhankelijk van de tijd dat 000.00 aanblijft. Dit wordt bereikt door 200.00 als een zelfhandhavend bit te gebruiken, geactiveerd door 000.00, dat 010.04 aan zet. Wanneer TIM001 afloopt, dus als de ingestelde waarde van TIM001 afgelopen is, wordt 010.04 uitgezet door de completionvlag van TIM001. D.w.z. de geïnverteerde completionvlag die gebruikt is in de instructieregel, creëert een uit executieconditie voor de instructie OUT 010.04.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 200.00 |
| 00001 | AND NOT | TIM001 |
| 00002 | OR | 000.00 |
| 00003 | OUT | 200.00 |

| | | | |
|---|---|---|---|
| 00004 | LD | 200.00 | |
| 00005 | TIM | 001 | #0015 |
| 00006 | LD | 200.00 | |
| 00007 | AND NOT | TIM001 | |
| 00008 | OUT | 010.04 | |

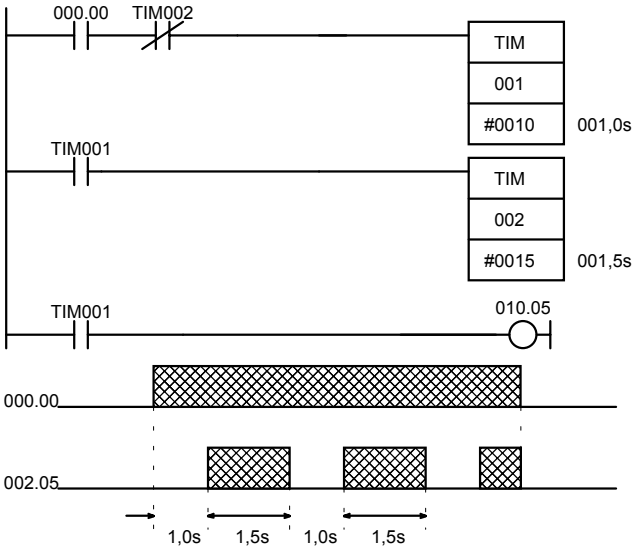
Het volgende netwerk heeft exact dezelfde functie en kan gebruikt worden om geheugen te besparen.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | OR | 010.04 | |
| 00002 | TIM | 001 | #0015 |
| 00003 | AND NOT | TIM001 | |
| 00004 | OUT | 010.04 | |

**Voorbeeld 5: Knipperbits**

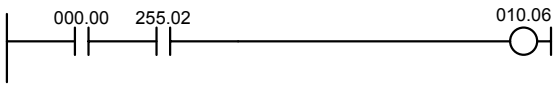
Met twee timers kunnen knipperbits gecreëerd worden die werken op een vast interval als een bepaalde executieconditie aan is. Eén TIM dient om het gespecificeerde bit aan en uit te zetten als de completionvlag van deze timer aan- of uitgaat. De andere TIM dient om de werking van de eerste te beheersen. Wanneer de completionvlag van de eerste timer aangaat wordt de tweede gestart. Wanneer de completionvlag van de tweede timer afloopt worden beide timers gereset en wordt TIM001 opnieuw gestart.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | AND NOT | TIM002 | |
| 00002 | TIM | 001 | #0010 |
| 00003 | LD | TIM001 | |
| 00004 | TIM | 002 | #0015 |
| 00005 | LD | TIM001 | |
| 00006 | OUT | 010.05 | |

Een eenvoudiger maar minder flexibele methode om een knipperbit te creëren is een AND tussen een klokpuls uit het SR gebied en de executieconditie die aan is als het knipperbit moet werken. Alhoewel deze methode geen TIM gebruikt, wordt het hier getoond zodat het vergeleken kan worden. Beperkingen van deze methode zijn dat de aan en uit tijd gelijk zijn en afhankelijk van de klokpulsen die in de CPU beschikbaar zijn. De met systeem klokpulsen gecreëerde knipperbits vergen natuurlijk wel minder programma.

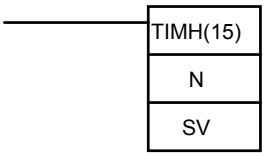
In het volgende voorbeeld is de seconde klokpuls (255.02) gebruikt om 010.06 elke seconde aan / uit te zetten. D.w.z. het bit zal 0,5 seconde aan en 0,5 seconde uit zijn. De precieze timing en de initiële status van 010.06 zijn afhankelijk van de status van de klok puls wanneer 000.00 aangaat.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND | 255.02 |
| 00002 | OUT | 010.06 |

### *4.12.2 Highspeed timer - TIMH(15)*

LADDER SYMBOOL




DEFINER WAARDEN

| N: TC nummer |
|---|
| # (000 - 127 (voorkeur 000 - 003)) |

OPERAND DATAGEBIEDEN

| IW: Ingestelde Waarde (BCD) |
|---|
| IR, AR, DM, HR, LR, # |

**Beperkingen**

De ingestelde waarde ligt tussen 00,00 en 99,99 seconden. De komma wordt niet ingevoerd. Bij het invoeren van een ingestelde waarde van 0001 ligt de tijd tussen de 0 en 0,01 seconde. Een ingestelde waarde van 0000 deactiveert de timer en maakt de completionvlag direct hoog.

Elk TC nummer kan gebruikt worden als definer in één timer of counter instructie.

Het is aan te raden om TC000 t/m TC003 te gebruiken als definer. Het gebruiken van andere TC nummers leidt tot onnauwkeurigheid als de cyclustijd groter is dan 10ms.

**Omschrijving**

TIMH(15) werkt op dezelfde manier om als TIM behalve dat TIMH de tijd meet in units van 0,01 seconde.

De cyclustijd beïnvloedt de nauwkeurigheid van TIMH(15) als TC004 t/m TC127 worden gebruikt. Gebruik TC000 t/m TC003 als de cyclustijd groter is dan 10ms. Bij gebruik van de TC nummers 000 t/m 003 wordt de TIMH(15) instructie op interrupt basis aangestuurd. De completionvlag van een normale timer instructie wordt door de instructie zelf aangezet. De onnauwkeurigheid van een normale timer is dus afhankelijk van de cyclustijd. Bij highspeed timers die op interrupt basis worden aangestuurd komt de completionvlag op als de tijd is afgelopen. Dus niet op het moment dat de instructie weer gescand wordt.

Raadpleeg de sectie "Timer - TIM" op pagina 87 voor details over de werking en voorbeelden. Op de hierboven aangegeven verschillen na zijn de werking van TIM en TIMH(15) identiek.

**Voorzorgen**

Timers in een interlock worden gereset als de executieconditie voor IL (02) uit is. Door het uitzetten van de PLC worden timers eveneens gereset. Als een timer gewenst is die onder deze condities niet gereset wordt, kunnen klokpulsen uit het SR gebied gebruikt worden om timers te maken met CNT instructies. Raadpleeg "counter - cnt" op pagina 93 voor details.

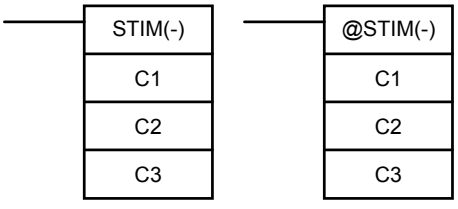
Programma uitvoer wordt voortgezet, zelfs als niet BCD ingestelde waarden worden gebruikt. De tijden worden hierdoor echter onbetrouwbaar.

**Vlaggen**

**ER**: IW (ingestelde waarde) is niet in BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.12.3 Interval timer - STIM(—)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| C1: Control data #1 |
|---|
| 000, 003, 006, 010 |

| C2: Control data #2 |
|---|
| IR, AR, DM, HR, TC, LR, # |

| C3: Control data #3 |
|---|
| IR, AR, DM, HR, TC, LR, # |

**Beperkingen**

C1 moet 000, 003, 006 of 010 zijn. Wanneer C1 000 of 003 is dan kan een constante groter dan 255 niet gebruikt worden op C3. Als C1 006 is dan kunnen DM6144 t/m DM6655 niet gebruikt worden voor C2 of C3. Wanneer C1 010 is moeten C2 en C3 000 zijn.

**Omschrijving**

STIM(—) wordt gebruikt om de interval timer te beheersen door te voorzien in vier basisfuncties: starten van de timer voor een one-shot interrupt, starten van de timer voor scheduled interrupts (interrupts op een vaste tijdbasis), stoppen van de timer en het uitlezen van de actuele waarde van de timer. Afhankelijk van de instelling op C1 wordt één van deze functies uitgevoerd. Mogelijke waarden van C1 zijn in de volgende tabel opgenomen. In het hoofdstuk “Interval timer interrupts” op pagina 46 staat meer informatie over de interval timer.

| Functie | C1 Waarde |
|---|---|
| Start timer | 000 |
| Start scheduled interrupt | 003 |
| Lees de actuele waarde | 006 |
| Stop de timer | 010 |

**Start interrupts**

Zet C1 op 000 om de interval timer te activeren in de one-shot mode. Wanneer C1 wordt ingesteld op 003 dan wordt de interval timer geactiveerd in de scheduled interrupt mode.

C2 specificeert de ingestelde waarde (IW) van de timer en kan een constante zijn of het eerste woord van twee opeenvolgende woorden die deze instelling bevatten. De instellingen kunnen een minimaal verschil vertonen afhankelijk van de gebruikte mode.

Als C2 een constante is (#) specificeert het een waarde van 0001 t/m 9999 in BCD. Het interval dat gebruikt wordt is in dit geval 1ms. De tijd kan in dit geval dus worden ingesteld tussen de 1 en 9999ms.

Als C2 een woord adres is, specificeert C2 de waarde (BCD, 0001 t/m 9999) en C2+1 het interval dat gebruikt wordt (BCD, 0005 t/m 0320) in units van 0,1ms. Het interval dat gebruikt kan worden ligt dus tussen de 0,5 en 32ms.

C3 specificeert het subroutine nummer 000 t/m 049.

**Opmerking** De uiteindelijke ingestelde tijd van de interval timer is:

(de inhoud van C2) x (de inhoud van C2+1) x 0,1ms

**Actuele waarde uitlezen**

Zet C1 op 006 om de actuele waarde van de timer uit te lezen.

C2 is de eerste van twee woorden die gebruikt worden om de actuele waarde van de timer in op te slaan. Op C2 wordt een aantal opgeslagen (BCD, 0000 t/m 9999), op C2+1 wordt het gebruikt interval opgeslagen (BCD in units van 0,1ms). Elke keer dat het interval dat op C2+1 is opgeslagen afloopt wordt de inhoud van C2 met één verlaagd (mits de STIM instructie elke keer wordt uitgevoerd).

C3 specificeert het woord dat gebruikt wordt om de tijd in op te slaan die verstreken is sinds de waarde op C2 met één verlaagd is (BCD in units van 0,1ms).

**Opmerking** De tijd die verstreken is sinds de timer is gestart kan als volgt berekend worden:

[(de inhoud van C2) x (de inhoud van C2+1) + (de inhoud van C3)] x 0,1ms.

**Timers stoppen**

Stel C1 in op 010 om de timer te stoppen.

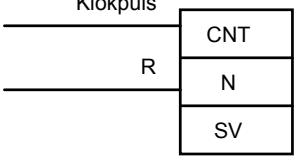
C2 en C3 hebben in dit geval geen functie en moeten worden ingesteld op 000.

**Vlaggen**

**ER**: Een waarde is niet in het juiste formaat opgegeven.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
De grootte van een geheugengebied is overschreden.

### *4.12.4 Counter - CNT*

LADDER SYMBOOL




DEFINER WAARDEN

| N: TC nummer |
|---|
| # (000 - 127) |

OPERAND DATAGEBIEDEN

| IW: Ingestelde Waarde (BCD) |
|---|
| IR, AR, DM, HR, LR, # |

**Beperkingen**

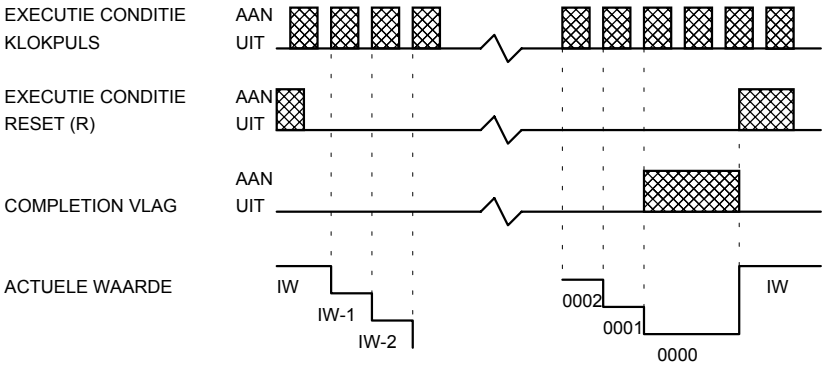
Elk TC nummer kan gebruikt worden als definer in één timer of counter instructie.

**Omschrijving**

CNT wordt gebruikt om vanaf de ingestelde waarde (IW) af te tellen naar nul. Elke keer als de executieconditie op de klokpuls ingang van uit naar aan gaat wordt de actuele waarde (IW) met één verlaagd. Als de executieconditie, sinds de vorige keer dat de instructie gescand werd, niet verandert of veranderd is van aan naar uit, wordt de actuele waarde van CNT niet verlaagd. De completionvlag van een counter gaat aan wanneer de actuele waarde nul wordt en blijft aan tot de counter gereset wordt.

CNT wordt gereset met de reset ingang R. Wanneer R aan is wordt de actuele waarde ingesteld op de ingestelde waarde. De counter zal klokpulsen niet tellen zolang de reset ingang (R) aan is. Het terugtellen van de ingestelde waarde naar nul begint vervolgens weer overnieuw als R uitgaat. De actuele waarde van CNT wordt niet gereset door spanningsuitval, het in program mode zetten van de PLC of tijdens actieve interlocks.

Veranderingen in de executiecondities, de completionvlag en de actuele waarde zijn in de volgende figuur getoond. De verschillen in hoogte van de lijn die de actuele waarde weergeeft geven alleen een indicatie van de veranderingen van de actuele waarde.




**Voorzorgen**

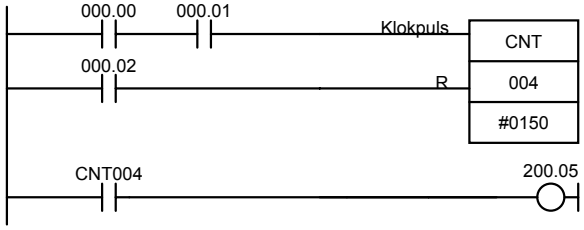
Programma-uitvoer zal normaal doorgaan als een niet-BCD getal als ingestelde waarde wordt gebruikt. De werking van de counter wordt er echter onbetrouwbaar door.

**Vlaggen**

**ER**: IW (ingestelde waarde) is niet in BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**Voorbeeld 1: Basis toepassing**

In het volgende voorbeeld wordt de actuele waarde verlaagd zodra zowel 000.00 als 000.01 aan zijn en 000.02 uit is en 000.00 of 000.01 de vorige keer dat CNT004 werd uitgevoerd laag waren. Zodra 150 pulsen zijn geteld (de actuele waarde bereikt nul) wordt 200.05 aangezet.

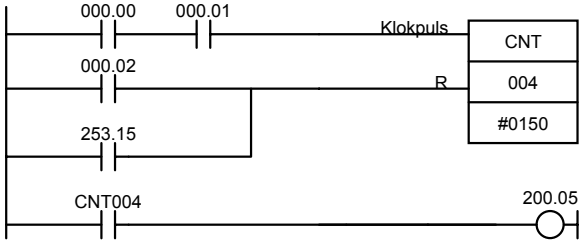



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND | 000.01 |
| 00002 | LD | 000.02 |

| | | | |
|---|---|---|---|
| 00003 | CNT | 004 | #0150 |
| 00004 | LD | CNT004 | |
| 00005 | OUT | 200.05 | |

In het bovenstaande programma zou 000.00 kunnen worden gebruikt om te bepalen wanneer CNT in werking is en 000.01 kan worden gebruikt als het bit wiens veranderingen worden geteld.

Het bovenstaande voorbeeld kan aangepast worden zodat de counter elke keer als de spanning aan wordt gezet gereset wordt. Dit wordt gerealiseerd door de *eerste scan vlag* uit het SR gebied (253.15) in de reset conditie van CNT op te nemen. Dit wordt hieronder getoond.



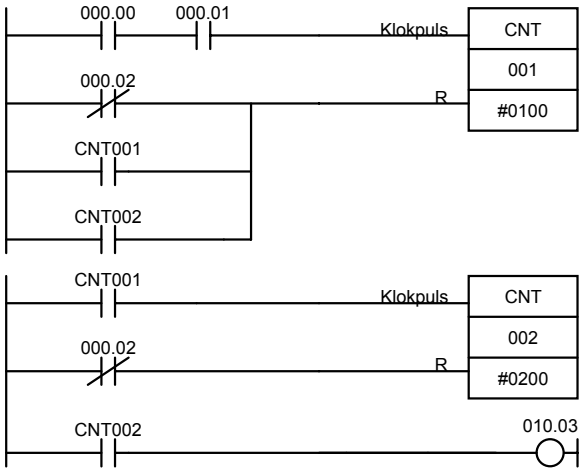

| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | AND | 000.01 | |
| 00002 | LD | 000.02 | |
| 00003 | OR | 253.15 | |
| 00004 | CNT | 004 | #0150 |
| 00005 | LD | CNT004 | |
| 00006 | OUT | 200.05 | |

**Voorbeeld 2: Verlengde counters**

Counters die verder moeten tellen dan 9.999 kunnen worden geprogrammeerd door een counter te gebruiken die het aantal maal telt dat een andere counter van de ingestelde waarde heeft teruggeteld naar nul.

In het volgende voorbeeld wordt 000.00 gebruikt om te bepalen wanneer CNT001 actief is. CNT001 telt, wanneer 000.00 aan is, het aantal uit/aan veranderingen van 000.01. CNT001 wordt gereset door zijn completionvlag. Hierdoor begint deze counter direct nadat de actuele waarde nul bereikt direct overnieuw met tellen. CNT002 telt het aantal maal dat de completionvlag van CNT001 aangaat. Bit 000.02 dient als een reset voor de gehele verlengde counter en reset zowel CNT001 als CNT002 wanneer het uit is. De completionvlag van CNT002 wordt gebruikt om CNT001 te resetten om te voorkomen dat CNT001 door kan gaan met tellen als de ingestelde waarde van CNT002 bereikt is.

Omdat in dit voorbeeld de ingestelde waarde voor CNT001 #100 is en die van CNT002 #200 komt de completionvlag van CNT002 op wanneer 100 x 200 of 20,000 uit / aan veranderingen zijn gemaakt door 000.01. Als resultaat zal 010.03 aangaan.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND | 000.01 |
| 00002 | LD NOT | 000.02 |
| 00003 | OR | CNT001 |
| 00004 | OR | CNT002 |

| | | | |
|---|---|---|---|
| 00005 | CNT | 001 | #0100 |
| 00006 | LD | CNT001 | |
| 00007 | LD NOT | 000.02 | |
| 00008 | CNT | 002 | #0200 |
| 00009 | LD | CNT002 | |
| 00010 | OUT | 010.03 | |

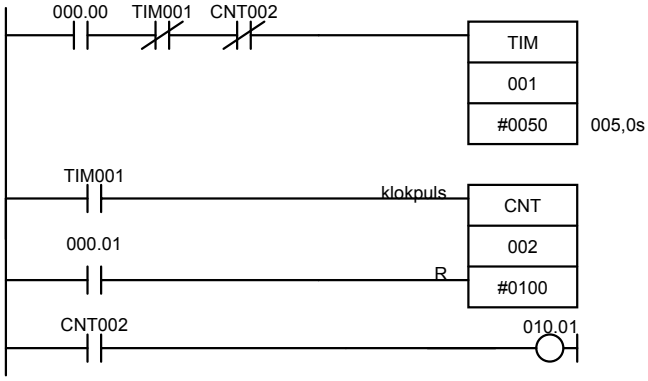
CNT kan op deze manier zo vaak als gewenst gebruikt worden om counters te maken die elke gewenste waarde kunnen tellen.

**Voorbeeld 3: Verlengde timers**

CNT kan op twee manieren gebruikt worden om verlengde timers te creëren, door TIM en CNT te combineren of door klokpulsen uit het SR gebied te gebruiken.

In het volgende voorbeeld telt CNT002 het aantal maal dat TIM001 zijn ingestelde waarde afgeteld heeft. De completionvlag van TIM001 wordt gebruikt om TIM001 te resetten zodat deze continu loopt en CNT002 het aantal maal kan tellen dat de completionvlag van TIM001 aangaat (CNT002 wordt met één verlaagd op het moment dat de completionvlag van TIM001 aangaat en TIM001 wordt door dezelfde completionvlag gereset). TIM001 wordt ook gereset door de completionvlag van CNT002 zodat de verlengde timer niet kan starten als deze is afgelopen. Om de timer na het aflopen weer te kunnen laten tellen moet eerst CNT002 gereset worden met 000.01, dat fungeert als de reset voor de gehele verlengde timer.

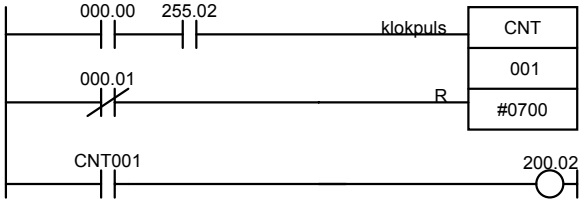
Omdat in dit voorbeeld de ingestelde waarde voor TIM001 5,0 seconden is en die van CNT002 #100, zal de completionvlag van CNT002 aan gaan als 5 seconden x 100, dat is 500 seconden (of 8 minuten en 20 seconden) zijn verlopend. Als resultaat zal 010.01 aangezet worden.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | AND NOT | TIM001 | |
| 00002 | AND NOT | CNT002 | |
| 00003 | TIM | 001 | #0050 |
| 00004 | LD | TIM001 | |
| 00005 | LD | 000.01 | |
| 00006 | CNT | 002 | #0100 |
| 00007 | LD | CNT002 | |
| 00008 | OUT | 010.01 | |

In het volgende voorbeeld telt CNT001 het aantal maal dat de 1 seconde klokpuls (bit 255.02) van uit naar aan verandert. Ook hier wordt 000.00 gebruikt om te bepalen of de counter wel of niet moet tellen.

Omdat in dit voorbeeld de ingestelde waarde voor CNT001 #700 is zal de completionvlag van CNT001 aan gaan als 1 seconde x 700, dat is 700 seconden (of 11 minuten en 40 seconden) verlopen zijn. Als resultaat zal 200.02 aangezet worden. Door het kiezen van bijvoorbeeld de 1 minuut klokpuls in plaats van de 1 seconde klokpuls kunnen makkelijk nog langere tijden gecreëerd worden.



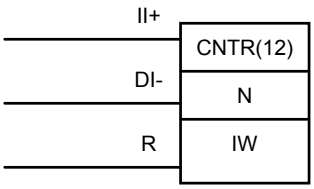

| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |

```
00001    AND       255.02
00002    LD NOT    000.01
00003    CNT       001        #0700
00004    LD        CNT001
00005    OUT       200.02
```

**Opmerking** De kortere klokpulsen hoeven niet noodzakelijkerwijs nauwkeuriger timers op te leveren omdat hun korte aan/uit tijden misschien niet goed gedetecteerd kunnen worden door de CPU, speciaal tijdens langere cyclustijden. In het bijzonder de 0.02 seconde en 0.1 seconde klokpulsen kunnen beter niet gebruikt worden om timers te creëren met counter instructies. Alleen als u er zeker van bent dat de cyclustijd van de PLC altijd kort genoeg is (een vuistregel hiervoor is minder dan de helft van de looptijd van de klokpuls) kunt u deze klokpulsen gebruiken.

### *4.12.5 Omkeerbare counter - CNTR(12)*

LADDER SYMBOOL




DEFINER WAARDEN

| N: TC nummer |
|---|
| # (000 - 127) |

OPERAND DATAGEBIEDEN

| IW: Ingestelde Waarde (BCD) |
|---|
| IR, AR, DM, HR, LR, # |

**Beperkingen** Elk TC nummer kan gebruikt worden als definer in één timer of counter instructie.

**Omschrijving** CNTR(12) is een omkeerbare circulaire counter die wordt gebruikt om te tellen tussen nul en de ingestelde waarde afhankelijk van veranderingen in twee executiecondities, die in de increment input (II+) en die in de decrement input (DI-).

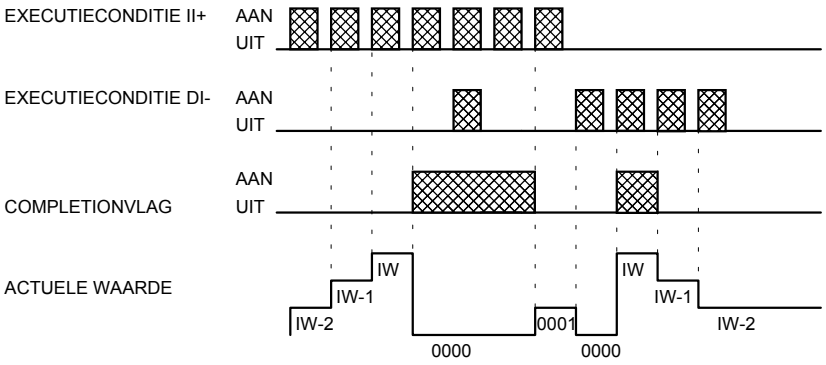
De actuele waarde wordt verhoogd met één zodra CNTR(12) wordt uitgevoerd met een *aan* executieconditie voor II+ en de vorige keer dat CNTR(12) werd uitgevoerd de executieconditie voor II+ *uit* was. D.w.z. voor de instructie op de opgaande flank op de II+ executieconditie. De actuele waarde wordt met één verlaagd wanneer de CNTR(12) instructie wordt uitgevoerd met een opgaande flank op de DI- executieconditie. Als zowel op de II+ als DI- voorwaarde een opgaande flank wordt gedetecteerd zal de actuele waarde van CNTR(12) niet veranderen.

Als de executiecondities van II+ en DI- niet zijn veranderd of zijn veranderd van aan naar uit dan zal de actuele waarde niet veranderen.

Wanneer de actuele waarde 0000 is en verlaagd wordt dan wordt deze op de ingestelde waarde (IW) gezet en wordt de completionvlag aangezet tot de actuele waarde weer wordt verlaagd. Wanneer de actuele waarde van de counter gelijk is aan de ingestelde waarde en verhoogd wordt, dan wordt de actuele waarde ingesteld op 0000 en wordt de completionvlag aangezet tot de actuele waarde opnieuw verhoogd wordt.

CNTR(12) wordt gereset met de reset ingang R. Wanneer R aan is, wordt de actuele waarde ingesteld op nul. Zolang als de reset ingang aan is, kan de actuele waarde niet verhoogd of verlaagd worden. Tellen is weer mogelijk als R uit is. De actuele waarde van CNTR(12) wordt niet gereset tijdens actieve interlocks, wanneer de spanning van de CPU uitgezet wordt en wanneer de CPU in de program mode gezet wordt.

Veranderingen in de II+ en DI- executiecondities, de completionvlag en de actuele waarde worden hieronder geïllustreerd vanaf een willekeurig moment in de tijd. De hoogte van de lijn die de actuele waarde weergeeft is alleen bedoeld als een indicatie van de verandering in de actuele waarde.




**Voorzorgen**

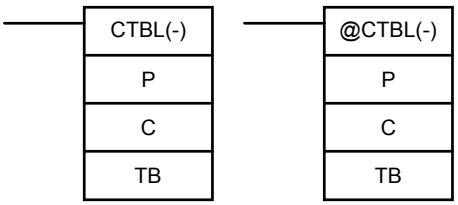
Programma-uitvoer zal normaal doorgaan als een niet-BCD getal als ingestelde waarde wordt gebruikt. De werking van de counter wordt er echter onbetrouwbaar door.

**Vlaggen**

**ER**: IW (ingestelde waarde) is niet in BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.12.6 Registreer vergelijkingstabel - CTBL(—)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| P: Poort |
|---|
| 000 |

| C: Control data |
|---|
| 000 t/m 003 |

| TB: Eerste woord vergelijkingstabel |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

Het eerste en het laatste woord van de vergelijkingstabel moeten in hetzelfde datagebied liggen. De lengte van de vergelijkingstabel kan afhankelijk van instellingen variëren.

**Omschrijving**

Wanneer de executieconditie uit is wordt CTBL(—) niet uitgevoerd. Wanneer de executieconditie aan is, slaat CTBL(—) een vergelijkingstabel op die de geregistreerde waarden vergelijkt met de actuele waarde van de highspeed counter. Afhankelijk van de waarde van C zal de vergelijking met de actuele waarde van de highspeed counter direct beginnen of apart gestart moeten worden met INI(—).

De poort operand (P) specificeert de highspeed counter die in de vergelijking gebruikt wordt. Bij de CPM1(A) is deze altijd 000.

De functie van CTBL(—) wordt bepaald door control data C, zoals in de volgende tabel getoond is. De functies worden na de tabel verklaard.

| C | CTBL(—) functie |
|---|---|
| 000 | Registreert een doelwaarde vergelijking en start het vergelijken |
| 001 | Registreert een bereik vergelijkingstabel en start het vergelijken |
| 002 | Registreert een doelwaarde vergelijking. Het vergelijken moet gestart worden met INI(—) |
| 003 | Registreert een bereik vergelijkingstabel. Het vergelijken moet gestart worden met INI(—) |

Wanneer de actuele waarde overeenkomt met de doelwaarde of binnen het gespecificeerde bereik valt wordt de betreffende subroutine aangeroepen en uitgevoerd. Zie hiervoor ook "Highspeed counter interrupts" op pagina 47.

Als de highspeed counter is geactiveerd in de PC Setup (DM6642) zal deze beginnen te tellen vanaf 0 als de CPM1 begint te werken. De actuele waarde zal niet worden vergeleken met de vergelijkingstabel tot deze is geregistreerd en de vergelijking is geactiveerd met INI(—) of CTBL(—). Het vergelijken kan worden gestart en gestopt of de actuele waarde kan worden gereset met INI(—).

Zodra een vergelijkingstabel is geregistreerd is deze geldig tot de CPM1 in program mode wordt gezet, uit wordt gezet of wanneer een fout optreedt tijdens

het registreren van een nieuwe tabel. Het is aan te raden om, indien mogelijk de gedifferentieerde uitvoering van CTBL(—) te gebruiken om de cyclustijd te reduceren.

**Doel waarde vergelijking**

Een doelwaarde vergelijkingstabel bevat tot aan zestien doelwaarden en een subroutinenummer voor elke doelwaarde. De corresponderende subroutine wordt aangeroepen en uitgevoerd als de actuele waarde overeenkomt met de doelwaarde. Wanneer een interruptroutine niet noodzakelijk is kan een niet gedefinieerde subroutine in de tabel geplaatst worden.

Het volgende diagram toont de opbouw van een doelwaarde vergelijkingstabel voor gebruik met highspeed counter 0 of highspeed counter 1 of 2 in lineaire mode.

| | | |
|---|---|---|
| TB | Aantal doelwaarden (BCD) | 0001 t/m 0016 |
| TB+1 | Doelwaarde #1, lage 4 cijfers (BCD) | Eén instelling |
| TB+2 | Doelwaarde #1, hoge 4 cijfers (BCD) | |
| TB+3 | Subroutinenummer voor doelwaarde 1 (zie noot) | |
| --- | --- | Overige instellingen |

**1, 2, 3...**

1. Het subroutinenummer kan 0000 t/m 0049 zijn en de subroutine wordt uitgevoerd zolang de actuele waarde van de counter binnen het gespecificeerde bereik ligt. Een waarde FFFF geeft aan dat geen subroutine uitgevoerd moet worden.

**Bereik vergelijking**

Een bereik vergelijkingstabel bevat acht bereiken die worden gedefinieerd door een achtcijferige lage limiet en een achtcijferige hoge limiet en het corresponderende subroutinenummer. De corresponderende subroutine wordt aangeroepen en uitgevoerd wanneer de actuele waarde binnen het opgegeven bereik valt. Wanneer interrupt uitvoer niet noodzakelijk is kan een niet gedefinieerd subroutinenummer ingevoerd worden.

Stel altijd acht bereiken in. Als minder dan acht bereiken nodig zijn stel dan de overige subroutine nummers in op FFFF. Als meer dan acht bereiken nodig zijn kan een andere vergelijkingsinstructie zoals BCMP(—) gebruikt worden om de overige bereiken te vergelijken met de actuele waarde van de highspeed counter. Denk eraan dat deze woorden één keer per cyclus worden gerefreshed.

Er zijn vlaggen in het AR gebied die aangeven dat de actuele waarde van een highspeed counter binnen een of meer van de acht bereiken valt. Deze vlaggen gaan aan wanneer de actuele waarde binnen het bijbehorende bereik valt.

| Counter | Vlaggen in het AR gebied |
|---|---|
| Highspeed counter 0 | AR11.00 t/m AR11.07 corresponderen met bereik 1 t/m 8 |

Het volgende diagram toont de opbouw van een bereik vergelijkingstabel voor gebruik met de highspeed counter

| | | |
|---|---|---|
| TB | Lage limiet #1, lage 4 cijfers (BCD) | Eerste bereik instelling |
| TB+1 | Lage limiet #1, hoge 4 cijfers (BCD) | |
| TB+2 | Hoge limiet #1, lage 4 cijfers (BCD) | |
| TB+3 | Hoge limiet #1, hoge 4 cijfers (BCD) | |
| TB+4 | Subroutine nummer (Zie noot 1) | |
| | | Overige instellingen |
| TB+35 | Lage limiet #8, lage 4 cijfers (BCD) | Achtste bereik instelling |
| TB+36 | Lage limiet #8, hoge 4 cijfers (BCD) | |
| TB+37 | Hoge limiet #8, lage 4 cijfers (BCD) | |
| TB+38 | Hoge limiet #8, hoge 4 cijfers (BCD) | |
| TB+39 | Subroutine nummer (Zie noot 1) | |

**1, 2, 3...**

1. Het subroutinenummer kan 0000 t/m 0049 zijn en de subroutine wordt uitgevoerd zolang de actuele waarde van de counter binnen het gespecificeerde bereik ligt. Een waarde FFFF geeft aan dat geen subroutine uitgevoerd moet worden.

De volgende tabel toont de mogelijke waarden voor doelwaarden, lage limiet waarden en hoge limiet waarden. Het hexadecimale cijfer F op het meest significante cijfer geeft aan dat de waarde negatief is.

| Counter | Mogelijke waarden |
|---|---|
| Highspeed counter | Up/Down mode: F003 2767 t/m 0003 2767 |

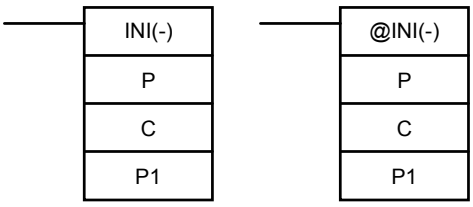
**Vlaggen**

**ER**: Er is een fout in de instellingen van de highspeed counter
De gespecificeerde poort en functie zijn niet compatibel.
Er staat een CTBL(—) instructie in de subroutine die aangeroepen wordt door een andere CTBL(—) instructie.
Een CTBL(—) instructie met een vergelijkingstabel is in een ander formaat uitgevoerd tijdens het vergelijken.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
De vergelijkingstabel overschrijdt de grootte van het datagebied of er is een fout gemaakt in de instellingen in de vergelijkingstabel.
CTBL(—) is uitgevoerd in een interrupt subroutine terwijl een puls I/O of highspeed counter instructie wordt uitgevoerd in het hoofdprogramma.

**AR05**: De vlaggen AR05.00 t/m AR05.07 worden aangezet om aan te geven dat de actuele waarde van de highspeed counter binnen het bereik 1 t/m 8 ligt.

### *4.12.7 Mode control - INI(—)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| P: Poort |
|---|
| 000 |

| C: Control data |
|---|
| 000 to 003 |

| P1: Eerste datawoord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

P moet 000 en C moet 000 t/m 003 zijn. P1 moet 000 zijn tenzij C 002 is. P1 en P1+1 moeten in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor P1.

**Omschrijving**

Wanneer de executieconditie uit is wordt INI(—) niet uitgevoerd. Wanneer de executieconditie aan is wordt INI(—) gebruikt om de werking van de highspeed counter operation in te stellen en om de pulsuitgang te stoppen.

De poort specifier (P) specificeert de highspeed counter of pulsuitgang die door de instructie wordt aangestuurd. P is bij de CPM1(A) altijd 000.

De functie van INI(—) wordt bepaald door de control data C. P1 en P1+1 bevatten de nieuwe actuele waarde van de highspeed counter als deze veranderd moet worden.

| C | P1 | INI(—) functie |
|---|---|---|
| 000 | 000 | Start CTBL(—) tabel vergelijking |
| 001 | 000 | Stopt CTBL(—) tabel vergelijking |
| 002 | Nieuwe actuele highspeed counter waarde | Verandert highspeed counter actuele waarde |
| 003 | 000 | Stopt pulsuitgang |

**CTBL(—) tabel vergelijking**

Als C 000 of 001 is, start of stopt INI(—) het vergelijken van de actuele waarde van de highspeed counter met de vergelijkingstabel die met CTBL(—) geregistreerd is. Dit onderwerp wordt gedetailleerd uitgelegd in "Highspeed counter interrupts" op pagina 47.

**Actuele waarde veranderen**

Als C 002 is, veranderd INI(—) de actuele waarde van de highspeed counter naar het achtcijferige getal op P1 en P1+1.

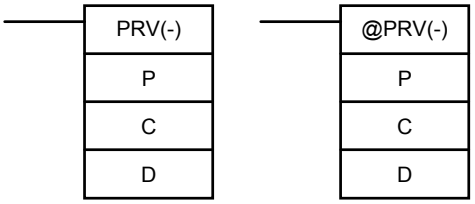
Bij de highspeed counter kan de actuele waarde F003 2767 t/m 0003 2767 in up/down mode of 0000 0000 t/m 0006 5535 in incrementele mode zijn. De hexadecimale waarde F op het meest significante cijfer van de actuele waarde geeft aan dat de actuele waarde negatief is.

| Linker 4 cijfers | Rechter 4 cijfers | Up/down mode | Incrementele mode |
|---|---|---|---|
| P1+1 | P1 | F003 2767 t/m 0003 2767 | 0000 0000 t/m 0006 5535 |

**Vlaggen** **ER**: De gespecificeerde poort en functie zijn niet compatibel.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
P1+1 overschrijdt de grootte van het datagebied (C=002).
Er is een fout gemaakt in de operand instellingen.
INI(—) is uitgevoerd in een interrupt subroutine terwijl een puls I/O of highspeed counter instructie wordt uitgevoerd in het hoofdprogramma.

### *4.12.8 Actuele waarde highspeed counter lezen - PRV(—)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| P: Poort |
|---|
| 000 |

| C: Control data |
|---|
| 000 to 003 |

| D: Eerste bestemmingswoord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen** P mag 000 en C 000 t/m 003 zijn. D en D+1 moeten in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor D.

**Omschrijving** Wanneer de executieconditie uit is, wordt PRV(—) niet uitgevoerd. Wanneer de executieconditie aan is, leest PRV(—) de door P en C gespecificeerde data en schrijft deze naar D of D en D+1.

De poort specifier (P) specificeert de highspeed counter of pulsuitgang die door de instructie wordt aangestuurd. Bij de CPM1(A) is P altijd 000.

De control data C bepaalt tot welk type data toegang wordt verkregen.

| C | Data | Bestemmingswoord(en) |
|---|---|---|
| 000 | Highspeed counter actuele waarde | D en D+1 |
| 001 | Status van highspeed counter of pulsuitgang | D |
| 002 | Bereik vergelijking resultaat | D |

**Highspeed counter actuele waarde (C=000)** Als C 000 is leest PRV(—) de actuele waarde van de gespecificeerde highspeed counter en schrijft deze achtcijferige waarde naar D en D+1.

Bij de highspeed counter kan de actuele waarde F003 2767 t/m 0003 2767 in up/down mode of 0000 0000 t/m 0006 5535 in incrementele mode zijn. De hexadecimale waarde F op het meest significante cijfer van de actuele waarde geeft aan dat de actuele waarde negatief is.

| Linker 4 cijfers | Rechter 4 cijfers | Up/down mode | Incrementele mode |
|---|---|---|---|
| P1+1 | P1 | F003 2767 t/m 0003 2767 | 0000 0000 t/m 0006 5535 |

**Highspeed counter of puls output status (C=001)** Als C 001 is, leest PRV(—) de status van de gespecificeerde highspeed counter of pulsuitgang en schrijft deze data naar D.

**Bereik vergelijking resultaten (C=002** Als C 002 is, leest PRV(—) het resultaat van de vergelijking van de actuele waarde met de 8 bereiken die met CTBL(—) gedefinieerd zijn en schrijft deze data naar D. Bits 00 t/m 07 van D bevatten de vergelijkingsresultaat vlaggen voor de bereiken 1 t/m 8 (0: Niet in het bereik; 1: In het bereik).

**Opmerking** Deze vlaggen staan ook in AR05 en AR06, maar die woorden worden elke cyclus maar één keer gerefreshed. De data die met PRV(—) wordt verkregen is dus van een recenter tijdstip.

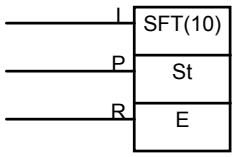
**Vlaggen** **ER**: De gespecificeerde poort en functie zijn niet compatibel.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
D+1 overschrijdt de grenzen van het datagebied (C=000).
Er is een fout gemaakt in de operand instellingen.
PRV(—) is uitgevoerd in een interruptsubroutine terwijl een puls I/O of highspeed counter instructie wordt uitgevoerd in het hoofdprogramma.

# 4.13 Schuiven van data

Alle hier beschreven instructies worden gebruikt om met data te schuiven in verschillende richtingen en met verschillende hoeveelheden. De eerste schuif-instructie, SFT (10), schuift een executieconditie in een schuifregister. De overige instructies verschuiven hoofdzakelijk data die al in het geheugen staat.

## *4.13.1 Schuifregister - SFT(10)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| St: Start woord |
|---|
| IR, AR, HR, LR |

| E: Eind woord |
|---|
| IR, AR, HR, LR |

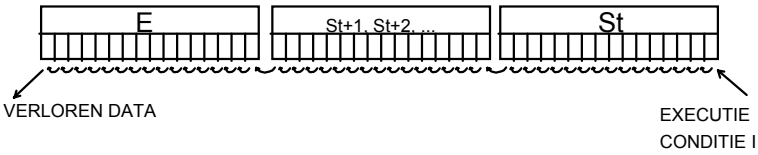
**Beperkingen**

E moet kleiner of gelijk zijn aan St en St en E moet in hetzelfde datagebied liggen.

Als een bit adres in één van de woorden die door het schuifregister wordt gebruikt, ook wordt gebruikt door een instructie die de individuele bitstatus (d.w.z., OUT, KEEP(11)) kan aansturen, wordt een foutmelding gegenereerd als de programma syntax wordt gecontroleerd. Het programma zal echter worden uitgevoerd zoals het is geschreven. Een voorbeeld hiervoor is het tweede voorbeeld voor het gebruik van schuifregisters dat hieronder behandeld wordt.

**Omschrijving**

SFT(10) wordt aangestuurd door drie executiecondities, I, P en R. Als SFT(10) wordt uitgevoerd en de executieconditie P aan is en de vorige keer dat de instructie werd uitgevoerd uit was en R is uit, dan wordt de executieconditie I op het meest rechtse bit van het schuifregister dat is gedefinieerd van St t/m E geschoven. D.w.z. wanneer I aan is, wordt een 1 in het register geschoven en wanneer I uit is wordt een 0 in het register geschoven. Wanneer I in het register wordt geschoven worden alle overige bits in het register één positie naar links geschoven. Het meest linker bit van het register gaat hierbij verloren.




De executieconditie op P functioneert als een gedifferentieerde instructie. D.w.z. dat I alleen in het register wordt geschoven op de opgaande flank van P. Een opgaande flank van executieconditie P ontstaat alleen wanneer P nu aan is en de vorige keer dat de instructie uitgevoerd werd uit was. Als de executieconditie P niet is veranderd of is veranderd van aan naar uit, zal het schuifregister niet veranderd worden.

St definieert het eerste woord van het schuifregister; E definieert het laatste woord. Het schuifregister omvat het deze beide woorden en alle woorden die er tussen liggen. Wanneer St en E hetzelfde woord toegewezen krijgen wordt een 16 bit (= 1 woord) schuifregister gecreëerd. St moet altijd een kleinere waarde hebben dan E.

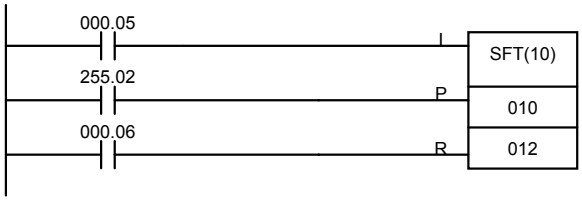
Wanneer de executieconditie R aangaat, worden alle bits in het schuifregister uitgezet (ingesteld op 0) en zal het schuifregister niet werken tot R weer uitgaat.

**Vlaggen**

Er worden geen vlaggen beïnvloed door SFT(10).

**Voorbeeld 1: Basis toepassing**

Het volgende voorbeeld gebruikt het 1 seconde klokpuls bit (255.02) zodat de executieconditie die geproduceerd wordt door input 000.05 elke seconde in een schuifregister van drie woorden (IR gebied 010 t/m 012) wordt geschoven.

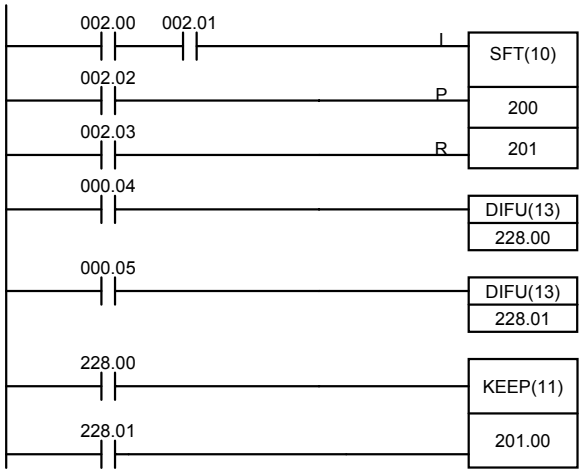



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.05 |
| 00001 | LD | 255.02 |

| | | | |
|---|---|---|---|
| 00002 | LD | 000.06 | |
| 00003 | SFT(10) | 010 | 012 |

**Voorbeeld 2: Aansturen van bits in schuifregisters**

Het volgende programma wordt gebruikt om de status van het 17e bit uit een schuifregister dat loopt van woord 200 t/m 201, aan te sturen. Het 17e bit kan aangezet worden met 000.04 en uitgezet worden met 000.05. De keep instructie die gebruikt wordt om dit bit te manipuleren wordt niet direct door deze contacten aangestuurd. Door de opgaande flanken van de contacten te nemen wordt het 17e geset of gereset als de ingang hiervoor aan gaat, schuift het register daarna een positie door dan wordt het nieuwe 17e bit niet aangestuurd. Worden ingangen 000.04 en 000.05 direct gebruikt om de KEEP(11) instructie aan te sturen dan zal het 17e bit van status veranderen zolang deze ingangen aan zijn, dus ook als het register data verschuift.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 002.00 | |
| 00001 | AND | 002.01 | |
| 00002 | LD | 002.02 | |
| 00003 | LD | 002.03 | |
| 00004 | SFT(10) | 200 | 201 |
| 00005 | LD | 000.04 | |
| 00006 | DIFU(13) | 228.00 | |
| 00007 | LD | 000.05 | |
| 00008 | DIFU(13) | 228.01 | |
| 00009 | LD | 228.00 | |
| 00010 | LD | 228.01 | |
| 00011 | KEEP(11) | 201.00 | |

Hoeft het bit alleen geset of gereset te worden dan kunnen de SET en RSET instructie gebruikt worden om een bit uit het schuifregister aan te sturen. Wanneer het bit met een OUT instructie aangestuurd wordt moet over dit programmadeel gesprongen worden als het bit niet aangestuurd moet worden. Een OUT instructie wordt immers altijd uitgevoerd. Is de executieconditie voor de OUT instructie hoog, dan wordt het operand bit altijd aangezet en is de executieconditie laag dan wordt het operandbit altijd uitgezet, ook als data door wordt geschoven in het schuifregister.

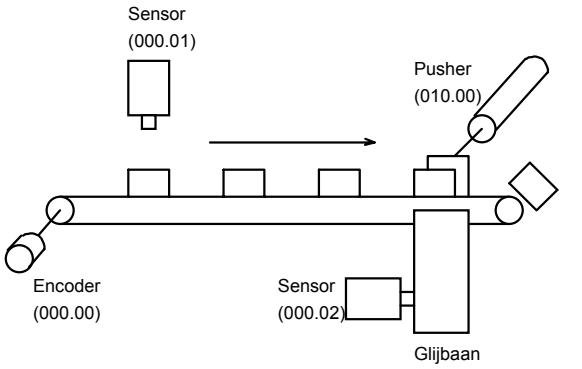
Als een bit adres in één van de woorden die door het schuifregister wordt gebruikt ook wordt gebruikt door een OUT instructie (of een andere die de bitstatus kan aansturen) wordt een foutmelding gegenereerd als de programma syntax wordt gecontroleerd. Het programma zal echter worden uitgevoerd zoals het is geschreven.

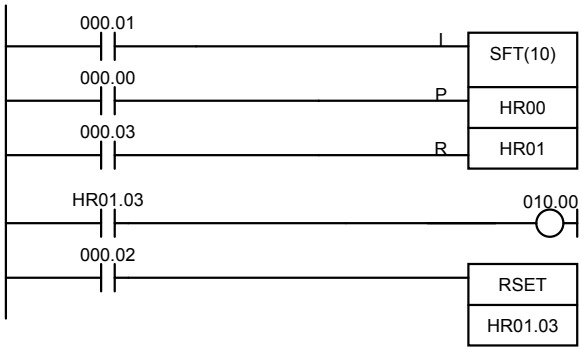
**Voorbeeld 3: Toepassing**

Het volgende programma stuurt de getekende installatie zo aan dat verkeerde producten (door een sensor gedetecteerd) van de band worden verwijderd. Om dit te realiseren wordt de executieconditie die bepaald wordt door de status van de eerste sensor (000.01), opgeslagen in een schuifregister. De bits in het schuifregister staan aan voor goede producten en uit voor foute. De band is zo ingesteld dat bit HR01.03 uit het schuifregister gebruikt kan worden om een pusher te activeren (010.00) wanneer het bit aangeeft dat het product fout is. Als HR01.03 aangaat zal dus 010.00 aangezet worden om de pusher uit te sturen.

De encoder (000.00) fungeert als de klokpuls voor de SFT(10) instructie. Het systeem is zo geconfigureerd dat het encoder signaal van laag naar hoog verandert als een te controleren product voor de sensor staat. Op de opgaande

flank van input 000.00 wordt de status van de sensor (goed of fout) opgeslagen in het schuifregister. Een andere sensor (000.02) wordt gebruikt om de verkeerde producten te detecteren nadat deze van de band verwijderd zijn. Met dit contact wordt de pusher weer ingestuurd en bit HR01.03 uit het schuifregister gereset.

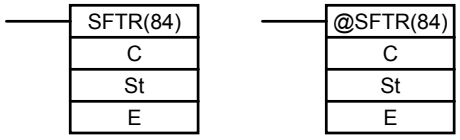



| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.01 | |
| 00001 | LD | 000.00 | |
| 00002 | LD | 000.03 | |
| 00003 | SFT(10) | HR00 | HR01 |
| 00004 | LD | HR01.03 | |
| 00005 | OUT | 010.00 | |
| 00005 | LD | 000.02 | |
| 00008 | RSET | HR01.03 | |

### *4.13.2 Omkeerbaar schuifregister - SFTR(84)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

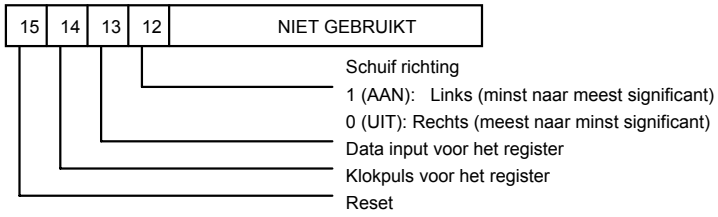
| |
|---|
| C: Control woord |
| IR, AR, DM, HR, LR |
| St: Start woord |
| IR, AR, DM, HR, LR |
| E: Eind woord |
| IR, AR, DM, HR, LR |

**Beperkingen**

E moet kleiner of gelijk zijn aan St en St en E moet in hetzelfde datagebied liggen.

**Omschrijving**

SFTR(84) wordt gebruikt om een schuifregister te creëren dat data naar links of naar rechts kan schuiven en dat uit één of meer woorden bestaat. Om een schuifregister te definiëren dat uit één woord bestaat moet voor St en E hetzelfde adres als operand ingevuld worden. Het control woord zorgt voor de schuifrichting, de status (data) die in het register moet worden geschoven, de puls waarop wordt geschoven en de reset ingang. De toewijzing van de bits in het control woord is als volgt:



SFTR(84) schuift data wanneer de instructie wordt uitgevoerd met een aan executieconditie, het resetbit uit is en zolang bit 14 uit het control woord aan is. In tegenstelling tot de SFT(10) instructie schuift SFTR(84) dus niet op de opgaande flank van bit 14, maar elke keer als de executieconditie voor de instructie aan is en bit 14 uit het controlewoord aan is. De data in het schuifregister wordt één bit verschoven in de richting die wordt aangegeven door bit 12. De status van bit 13 uit het control woord wordt in het register geschoven, de status van het bit dat aan de andere kant uit het register wordt geschoven wordt op de Carry (CY) geplaatst. Wanneer SFTR(84) wordt uitgevoerd met een uit executieconditie of wanneer SFTR(84) wordt uitgevoerd met bit 14 uit zal het schuifregister onveranderd blijven. Wanneer SFTR(84) wordt uitgevoerd met een aan executieconditie en het resetbit (bit 15) is uit, dan wordt het gehele schuifregister en de Carry ingesteld op nul.

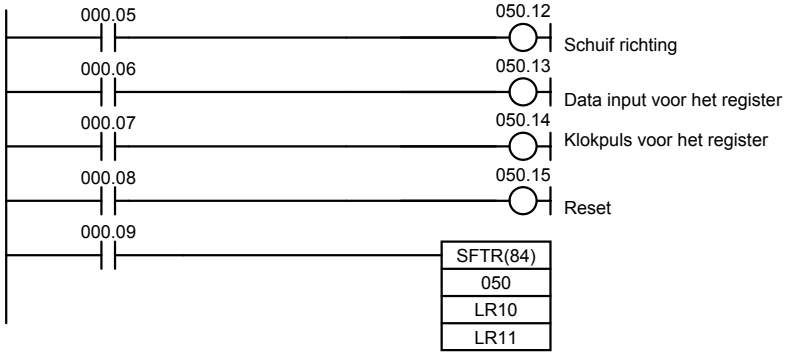
**Vlaggen**

**ER**: St en E liggen niet in hetzelfde datagebied of ST is groter dan E.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY**: Ontvangt na het schuiven de status van bit 00 uit St of bit 15 uit E afhankelijk van de schuifrichting.

**Voorbeeld:**

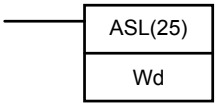
In het volgende voorbeeld worden 000.05, 000.06, 000.07 en 000.08 gebruikt om de bits uit het control woord van een SHIFT(84) instructie aan te sturen. Het schuifregister schuift over twee woorden (LR10 en LR11) en wordt geactiveerd door 000.09.

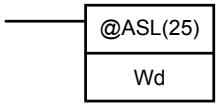



| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.05 | | |
| 00001 | OUT | 050.12 | | |
| 00002 | LD | 000.06 | | |
| 00003 | OUT | 050.13 | | |
| 00004 | LD | 000.07 | | |
| 00005 | OUT | 050.14 | | |
| 00006 | LD | 000.08 | | |
| 00007 | OUT | 050.15 | | |
| 00008 | LD | 000.09 | | |
| 00009 | SFT(10) | 050 | LR10 | LR11 |

### *4.13.3 Arithmetic shift left - ASL(25)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

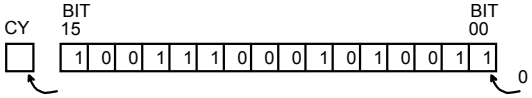
| Wd: Schuif woord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

DM6144 t/m DM6655 kan niet gebruikt worden voor Wd.

**Omschrijving**

Wanneer de executieconditie uit is, wordt ASL(25) niet uitgevoerd. Wanneer de executieconditie aan is, schuift ASL(25) een 0 in bit 00 van Wd, schuift de bits in Wd één bi naar links en schuift de status van bit 15 in CY.




**Vlaggen**

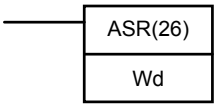
**ER**: Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

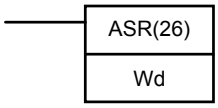
**CY**: Ontvangt de status van bit 15.

**EQ**: Aan wanneer de inhoud van Wd nul is, anders uit.

### *4.13.4 Arithmetic shift right - ASR(26)*

LADDER SYMBOOL



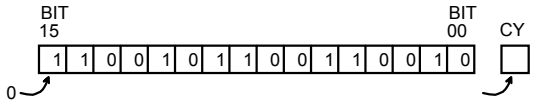




OPERAND DATAGEBIEDEN

| Wd: Schuif woord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor Wd.

**Omschrijving** Wanneer de executieconditie uit is, wordt ASR(26) niet uitgevoerd. Wanneer de executieconditie aan is, schuift ASR(26) een 0 in bit 15 van Wd, schuift de bits in Wd één bit naar rechts en schuift de status van bit 00 in CY.




**Vlaggen**

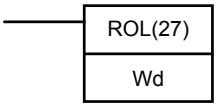
**ER**: Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

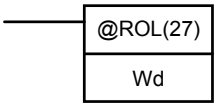
**CY**: Ontvangt de status van bit 00.

**EQ**: Aan wanneer de inhoud van Wd nul is, anders uit.

### *4.13.5 Roteer links - ROL(27)*

LADDER SYMBOOL



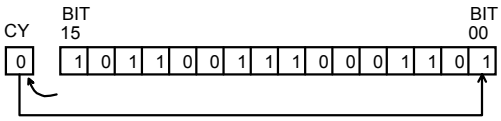




OPERAND DATAGEBIEDEN

| Wd: Roteer woord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor Wd.

**Omschrijving** Wanneer de executieconditie uit is, wordt ROL(27) niet uitgevoerd. Wanneer de executieconditie aan is, schuift ROL(27) alle bits in Wd één bit naar links, de status van CY wordt in bit 00 van Wd geplaatst en de status van bit 15 uit Wd wordt in CY gezet. ROL(27) roteert dus over 17 bits, de 16 uit woord Wd én CY.




**Voorzorgen** Gebruik STC(41) om CY te setten of CLC(41) om CY te resetten voor een roteer bewerking wordt uitgevoerd om zeker te zijn van een juiste status van CY voor de uitvoer van ROL(27). Tenzij natuurlijk de CY status van een voorgaande instructie met ROL(27) verwerkt moet worden.

**Vlaggen**

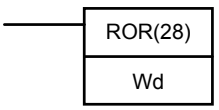
**ER**: Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

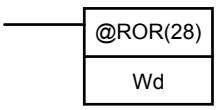
**CY**: Ontvangt de status van bit 15.

**EQ**: Aan wanneer de inhoud van Wd nul is, anders uit.

### *4.13.6 Roteer rechts - ROR(28)*

LADDER SYMBOOL



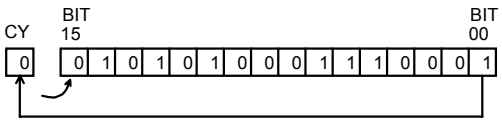




OPERAND DATAGEBIEDEN

| Wd: Roteer woord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor Wd.

**Omschrijving** Wanneer de executieconditie uit is, wordt ROR(28) niet uitgevoerd. Wanneer de executieconditie aan is, schuift ROR(28) alle bits in Wd één bit naar rechts, de status van CY wordt in bit 15 van Wd geplaatst en de status van bit 00 uit Wd wordt in CY gezet. ROR(28) roteert dus over 17 bits, de 16 uit woord Wd én CY.




**Voorzorgen** Gebruik STC(41) om CY te setten of CLC(41) om CY te resetten voor een roteer bewerking wordt uitgevoerd om zeker te zijn van een juiste status van CY voor de uitvoer van ROR(28). Tenzij natuurlijk de CY status van een voorgaande instructie met ROR(28) verwerkt moet worden.

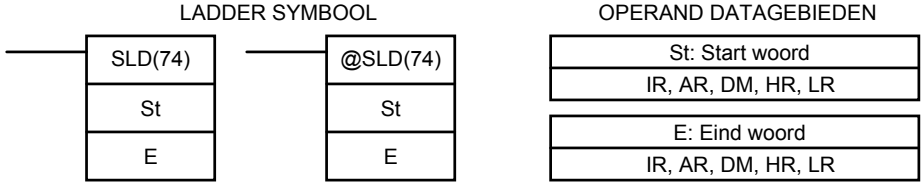
**Vlaggen**

**ER**: Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY**: Ontvangt de status van bit 00.

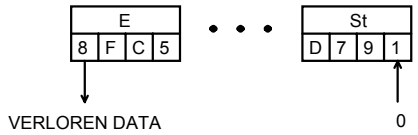
**EQ**: Aan wanneer de inhoud van Wd nul is, anders uit.

### *4.13.7 Schuif één digit naar links - SLD(74)*




**Beperkingen** St en E moeten in hetzelfde datagebied liggen en E moet groter of gelijk zijn aan St. DM6144 t/m DM6655 kunnen niet gebruikt worden voor St en E.

**Omschrijving** Wanneer de executieconditie uit is wordt SLD(74) niet uitgevoerd. Wanneer de executieconditie aan is, schuift SLD(74) de data tussen St en E (inclusief) één digit (vier bits) naar links. Een 0 wordt op het meest rechter digit van St geplaatst en de inhoud van het meest linker digit van E is verloren.



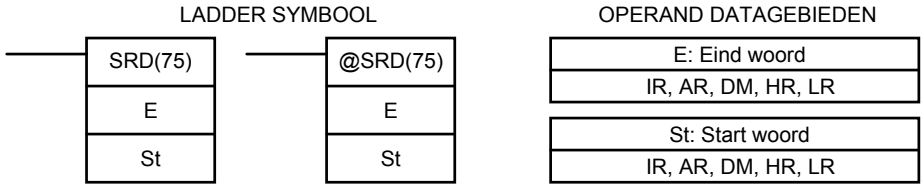

**Voorzorgen** Als de spanning van de CPU wegvalt tijdens een schuifbewerking over meer dan 50 woorden kan de schuifbewerking gedeeltelijk uitgevoerd zijn.

**Vlaggen**

**ER**: De St en E woorden liggen in verschillende gebieden of St is groter dan E.

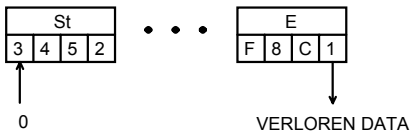
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.13.8 Schuif één digit naar rechts - SRD(75)*




**Beperkingen** St en E moeten in hetzelfde datagebied liggen en E moet kleiner of gelijk zijn aan St. DM6144 t/m DM6655 kunnen niet gebruikt worden voor E en St.

**Omschrijving** Wanneer de executieconditie uit is wordt SRD(75) niet uitgevoerd. Wanneer de executieconditie aan is, schuift SRD(75) de data tussen St en E (inclusief) één digit (vier bits) naar rechts. Een 0 wordt op het meest linker digit van St geplaatst en de inhoud van het meest rechter digit van E is verloren.

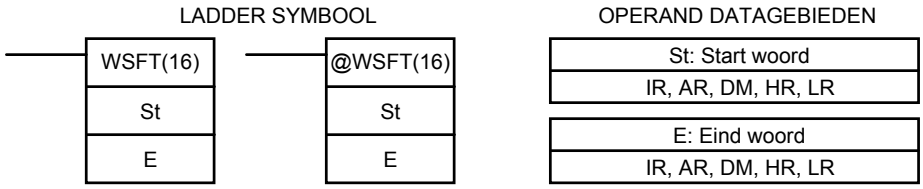



**Voorzorgen** Als de spanning van de CPU wegvalt tijdens een schuifbewerking over meer dan 50 woorden kan de schuifbewerking gedeeltelijk uitgevoerd zijn.

**Vlaggen** **ER**: De St en E woorden liggen in verschillende gebieden of St is kleiner dan E.

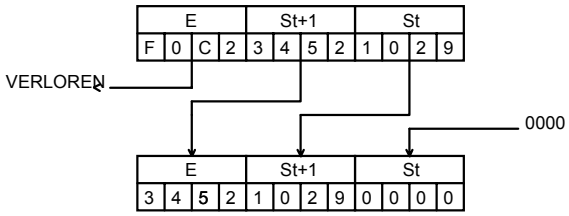
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.13.9 Schuif woord - WSFT(16)*




**Beperkingen** St en E moeten in hetzelfde datagebied liggen en E moet groter of gelijk zijn aan St. DM6144 t/m DM6655 kunnen niet gebruikt worden voor St en E.

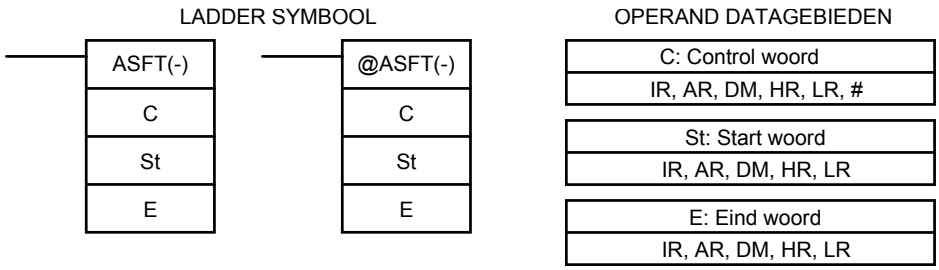
**Omschrijving** Wanneer de executieconditie uit is wordt WSFT(16) niet uitgevoerd. Wanneer de executieconditie aan is, verschuift WSFT(16) de data tussen St en E in units van één woord. Woord St wordt gevuld met nullen en de inhoud van woord E is verloren.




**Vlaggen** **ER**: De St en E woorden liggen in verschillende gebieden of St is groter dan E.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.13.10 Asynchroon schuifregister - ASFT(—)*




**Beperkingen** St en E moeten in hetzelfde datagebied liggen en E moet groter of gelijk zijn aan St. DM6144 t/m DM6655 kunnen niet gebruikt worden voor St en E.

**Omschrijving** Wanneer de executieconditie uit is, doet ASFT(—) niets en gaat het programma door met de volgende instructie. Wanneer de executieconditie aan is, wordt ASFT(—) gebruikt om een omkeerbaar, asynchroon op woorden gebaseerd schuifregister tussen St en E te creëren en aan te sturen. Dit register schuift alleen woorden wanneer op het volgende woord nul staat. Wanneer geen enkel woord in het register nul is, wordt er dus ook niet geschoven met data door deze instructie. Er wordt maar één woord verschoven voor elk woord in het register dat nul bevat. Wanneer de inhoud van een woord wordt verschoven naar het volgende woord, wordt de inhoud van het originele woord ingesteld op nul. In essentie komt het erop neer dat elk woord in het register dat nul bevat van plaats verwisselt met het volgende woord (Zie het voorbeeld verderop).

De schuifrichting (d.w.z. de keuze of het “volgende woord" het woord één positie hoger in het register of één positie lager in het register is) wordt toegewezen in C. C wordt ook gebruikt om het register te resetten. Het hele register een deel ervan kan worden gereset door het gewenste deel met St en E in te stellen.

**Control woord** Bits 00 t/m 12 van C worden niet gebruikt. Bit 13 is de schuifrichting. Zet bit 13 aan om omlaag te schuiven (richting woorden met lagere adressen) en uit om omhoog

te schuiven (richting woorden met hogere adressen). Bit 14 is het *shift enable bit.* Zet bit 14 aan om de werking van het schuifregister te activeren en zet bit 13 uit om het register te deactiveren. Bit 15 is het resetbit. Het register wordt gereset (ingesteld op nul) tussen St en E als ASFT(-) wordt uitgevoerd met bit 15 aan. Zet bit 15 uit voor normale werking van de instructie.

**Vlaggen**

**ER**: De St en E woorden liggen in verschillende gebieden of St is groter dan E.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**Opmerking** Als de niet gedifferentieerde uitvoering van ASFT(—) wordt gebruikt, wordt data elke cyclus geschoven als de executieconditie aan is. Gebruik de gedifferentieerde uitvoering om dit te voorkomen.

**Voorbeeld** Het volgende voorbeeld toont de instructie ASFT(—) in gebruik om woorden uit een schuifregister dat uit 11 woorden bestaat (DM0100 t/m DM0110) aan te sturen. Met de constante waarde #6000 die op C is geplaatst, worden woorden die geen nul bevatten geschoven richting St (DM0110).


| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 00000 | | |
| 00001 | ASFT(—) | #6000 | DM0100 | DM0110 |

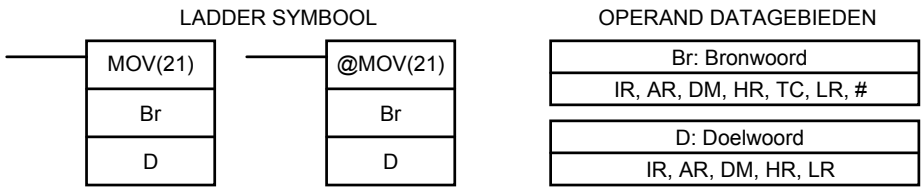
| | Voor uitvoer | Na 1 maal uitvoer | Na 7 maal uitvoer |
|---|---|---|---|
| DM0100 | 1234 | 1234 | 1234 |
| DM0101 | 0000 | 0000 | 2345 |
| DM0102 | 0000 | 2345 | 3456 |
| DM0103 | 2345 | 0000 | 4567 |
| DM0104 | 3456 | 3456 | 5678 |
| DM0105 | 0000 | 4567 | 6789 |
| DM0106 | 4567 | 0000 | 789A |
| DM0107 | 5678 | 5678 | 0000 |
| DM0108 | 6789 | 6789 | 0000 |
| DM0109 | 0000 | 789A | 0000 |
| DM0110 | 789A | 0000 | 0000 |

**Opmerking** De nullen worden naar boven geschoven als C #4000 is en het gehele schuifregister wordt ingesteld op nul als C #8000 is.

## 4.14 Data verplaatsen

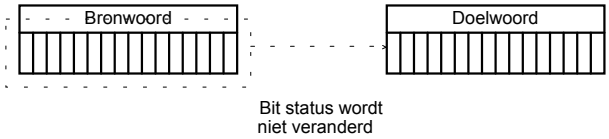
Hier wordt een omschrijving gegeven van de instructies die gebruikt worden om data tussen verschillende adressen in datagebieden te verplaatsen. Deze verplaatsingen kunnen binnen hetzelfde datagebied of tussen verschillende datagebieden plaatsvinden. Dataverplaatsing is essentieel voor het gebruik van alle datagebieden van de PLC. Effectieve communicatie in netwerken maakt vaak ook dataverplaatsing nodig. Al deze instructies veranderen alleen de inhoud van de woorden waarnaar de data verplaatst wordt, d.w.z. dat de inhoud van de bron woorden hetzelfde is voor en na de uitvoer van elke dataverplaatsing instructie. De actie die deze instructies uitvoeren kan beter kopiëren genoemd worden.

### *4.14.1 Verplaatsen - MOV(21)*




**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor D.

**Omschrijving** Wanneer de executieconditie uit is, wordt MOV(21) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert MOV(21) de inhoud van Br naar D.




**Voorzorgen**

TC nummers kunnen niet aan D worden toegewezen om de actuele waarde van een timer of counter te veranderen. TC nummers kunnen wel toegewezen worden aan het doelgebied van de BSET(71) instructie. Raadpleeg voor het kopiëren van waarden naar het TC gebied dus de BSET(71) instructie.

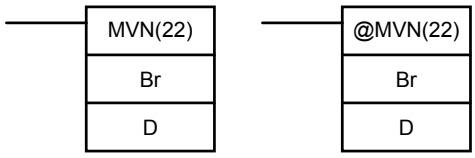
**Vlaggen**

**ER**: Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ**: Aan als alle bits in het woord dat naar D verplaatst wordt nul zijn.

### *4.14.2 Verplaats geïnverteerd - MVN(22)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Br: Bronwoord |
|---|
| IR, AR, DM, HR, TC, LR, # |

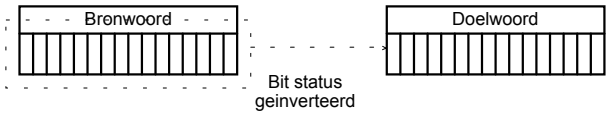
| D: Doelwoord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

DM6144 t/m DM6655 kan niet gebruikt worden voor D.

**Omschrijving**

Wanneer de executieconditie uit is, wordt MVN(22) niet uitgevoerd. Wanneer de executieconditie aan is, verplaatst MVN(22) de geïnverteerde inhoud van Br (het gespecificeerde woord of de vier cijferige hexadecimale constante) naar D. Voor elk bit in Br dat aan is wordt het corresponderende bit in D uitgezet en voor elk bit in Br dat uit is wordt het corresponderende bit in D aangezet.




**Voorzorgen**

TC nummers kunnen niet aan D worden toegewezen om de actuele waarde van een timer of counter te veranderen. TC nummers kunnen wel toegewezen worden aan het doelgebied van de BSET(71) instructie. Raadpleeg voor het kopiëren van waarden naar het TC gebied dus de BSET(71) instructie.

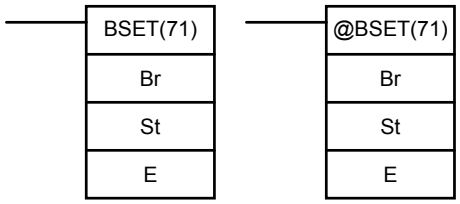
**Vlaggen**

**ER**: Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ**: Aan als alle bits in het woord dat naar D verplaatst wordt nul zijn.

### *4.14.3 Set blok - BSET(71)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Br: Brondata |
|---|
| IR, AR, DM, HR, TC, LR, # |

| St: Start woord |
|---|
| IR, AR, DM, HR, TC, LR |

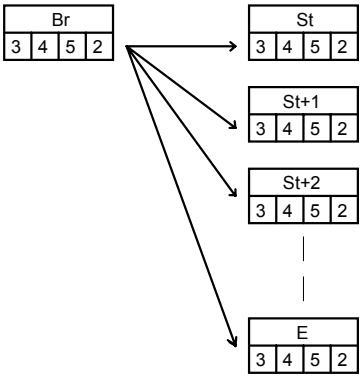
| E: Eind woord |
|---|
| IR, AR, DM, HR, TC, LR |

**Beperkingen**

St moet kleiner of gelijk zijn aan E en St en E moet in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor St en E.

**Omschrijving**

Wanneer de executieconditie uit is, wordt BSET (71) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert BSET(71) de inhoud van Br naar alle woorden van St t/m E.



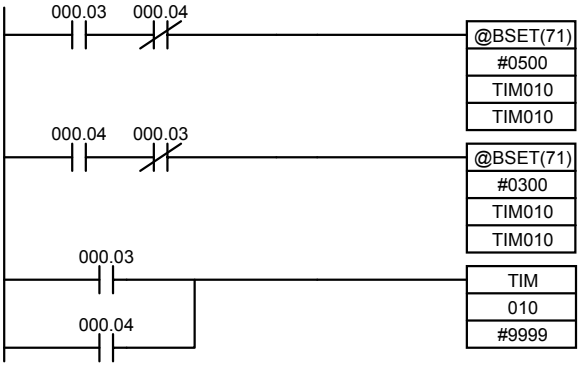

BSET(71) kan gebruikt worden om de actuele waarde van timers en counters te veranderen (dit kan niet worden gedaan met MOV(21) of MVN(22)). BSET(71) kan ook worden gebruikt om secties in een datagebied te wissen om het voor te bereiden voor de uitvoer van andere instructies.

**Vlaggen**

**ER:** St en E liggen niet in hetzelfde datagebied of St is groter dan E.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

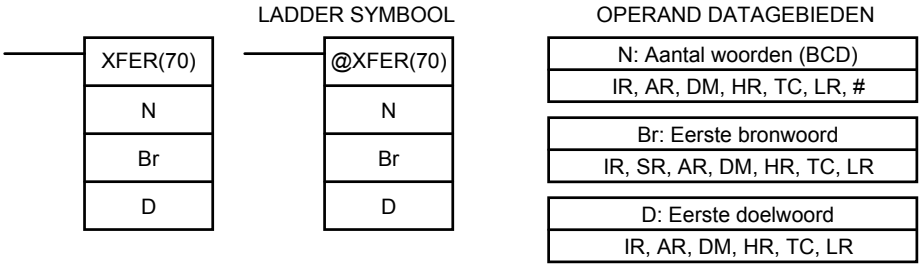
**Voorbeeld**

Het volgende voorbeeld toont hoe BSET(71) gebruikt wordt om de actuele waarde van een timer aan te passen, afhankelijk van de status van 000.03 en 000.04. Wanneer 000.03 aan is, werkt TIM010 als een 50 seconden timer, wanneer 000.04 aan is, werkt TIM010 als een 30 seconden timer.




| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.03 | | |
| 00001 | AND NOT | 000.04 | | |
| 00002 | @BSET(71) | #0500 | TIM010 | TIM010 |
| 00003 | LD | 000.04 | | |
| 00004 | AND NOT | 000.03 | | |
| 00005 | @BSET(71) | #0300 | TIM010 | TIM010 |
| 00006 | LD | 000.03 | | |
| 00007 | OR | 000.04 | | |
| 00008 | TIM | 010 | #9999 | |

## *4.14.4 Verplaats blok - XFER(70)*

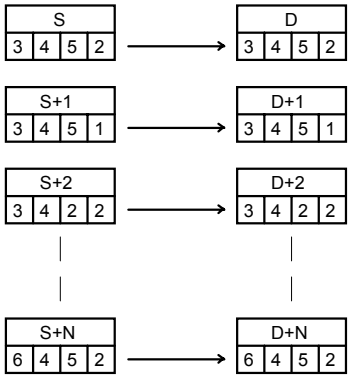



**Beperkingen**

Br en D mogen in hetzelfde datagebied liggen, maar de blokgebieden mogen niet overlappen. S en S+N moeten in hetzelfde datagebied liggen, zo ook D en D+N. DM6144 t/m DM6655 kan niet gebruikt worden voor D.

**Omschrijving**

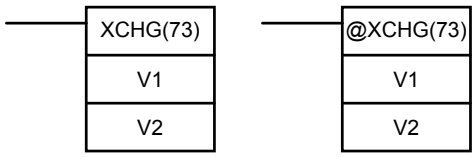
Wanneer de executieconditie uit is, wordt XFER(70) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert XFER(70) de inhoud van S, S+1, ..., S+N naar D, D+1, ..., D+N.




**Vlaggen** **ER:** N is niet opgegeven in BCD
S en S+N of D en D+N liggen niet in hetzelfde datagebied.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.14.5 Verwissel data - XCHG(73)*

LADDER SYMBOOL



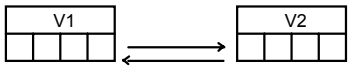

OPERAND DATAGEBIEDEN

| V1: Verwissel woord 1 |
|---|
| IR, AR, DM, HR, TC, LR |

| V2: Verwissel woord 2 |
|---|
| IR, AR, DM, HR, TC, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor V1 en V2.

**Omschrijving** Wanneer de executieconditie uit is, wordt XCHG(73) niet uitgevoerd. Wanneer de executieconditie aan is, verwisselt XCHG(73) de inhoud van V1 en V2.



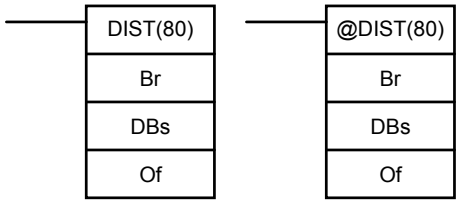

Wanneer u de inhoud van blokken die groter zijn dan 1 woord wilt verwisselen, kunnen werkwoorden als tijdelijke buffer gebruikt worden om één van de blokken te bewaren en kan XFER(70) drie keer gebruikt worden.

1) kopieer V1 naar de tijdelijke buffer.
2) kopieer V2 naar V1.
3) Kopieer de tijdelijke buffer naar V2.

**Vlaggen** **ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.14.6 Distribueer één woord - DIST(80)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

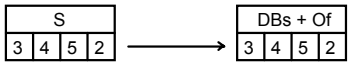
| Br: Brondata |
|---|
| IR, AR, DM, HR, TC, LR, # |

| DBs: Doel basiswoord |
|---|
| IR, AR, DM, HR, TC, LR |

| Of: Offset data (BCD) |
|---|
| IR, AR, DM, HR, TC, LR, # |

**Beperkingen** Of moet worden opgegeven in BCD. DBs moet in hetzelfde datagebied liggen als DBs+Of. DM6144 t/m DM6655 kan niet gebruikt worden voor Dbs of C.

**Omschrijving** Wanneer de executieconditie uit is, wordt DIST(80) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert DIST(80) de inhoud van Br naar DBs+Of. D.w.z. Of wordt opgeteld bij DBs om het doelwoord te bepalen.




**Stack bewerkingen** Wanneer de bits 12 t/m 15 van Of 9 (1001) bevatten, dan wordt DIST(80) gebruikt voor stack bewerkingen. De overige drie cijfers van Of geven in dit geval de grootte van de stack weer. De inhoud van DBs wordt gebruikt als stack pointer.

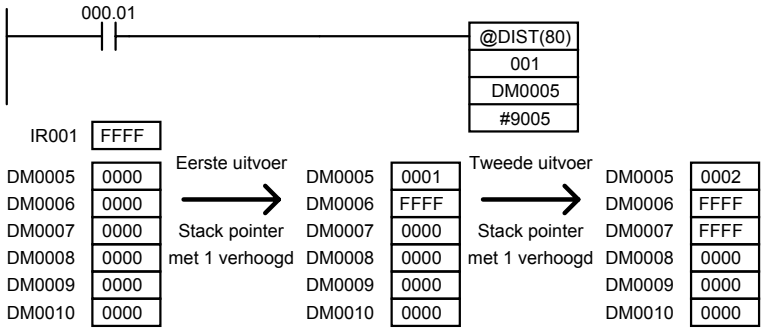
Wanneer de executieconditie uit is, wordt DIST(80) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert DIST(80) de inhoud van Br naar DBs+1+de inhoud van woord Dbs. Met andere woorden, 1 en de inhoud van adres DBs worden bij elkaar opgeteld. Dit wordt als offset gebruikt ten opzichte van Dbs om het woord te bepalen waar de data geplaatst moet worden. Vervolgens wordt de inhoud van DBs met 1 verhoogd.

**Opmerking**

1. DIST(80) wordt elke cyclus uitgevoerd, tenzij de gedifferentieerde uitvoering (@DIST(80)) wordt gebruikt of DIST(80) wordt toegepast in combinatie met DIFU(13) of DIFD(14).
2. Zorg ervoor dat de stack pointer wordt geïnitialiseerd voor DIST(80) wordt gebruikt voor stack bewerkingen.

**Voorbeeld**

Het volgende voorbeeld toont hoe DIST(80) wordt gebruikt om een stack te creëren op DM0006 t/m DM0010. DM0005 is de stack pointer.




**Vlaggen**

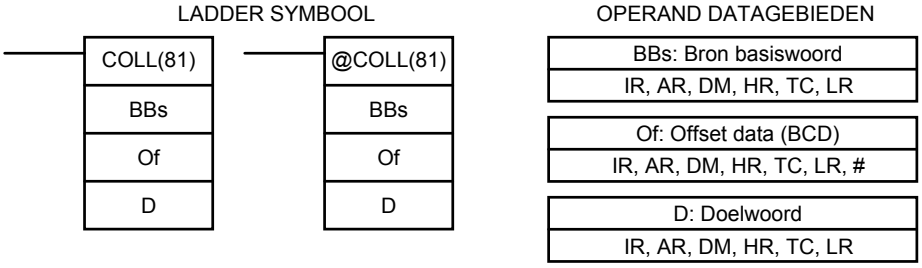
**ER:** De offset data is niet opgegeven in BCD of wanneer het opgeteld wordt bij DBs ligt het resultaat adres buiten het datagebied van DBs.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

Tijdens een stack bewerking is de waarde van de stack pointer groter geworden dan de lengte van de stack.

**EQ:** Aan wanneer de inhoud van Br nul is, anders uit.

## *4.14.7 Verzamel data - COLL(81)*

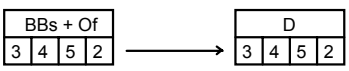



**Beperkingen**

Of moet worden opgegeven in BCD. BBs moet in hetzelfde datagebied liggen als BBs+Of. DM6144 t/m DM6655 kan niet gebruikt worden voor D.

**Omschrijving**

Wanneer de executieconditie uit is, wordt COLL(81) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert COLL(81) de inhoud van BBs+Of naar D. D.w.z. Of wordt opgeteld bij BBs om het doelwoord te bepalen.



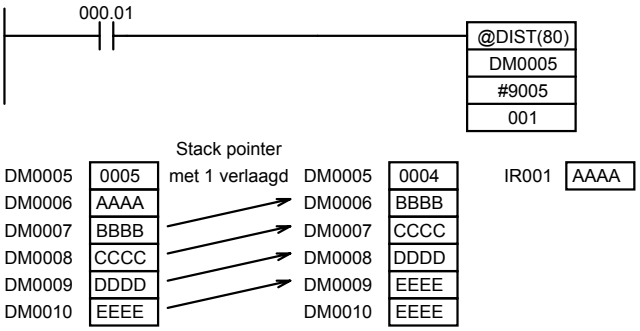

**FIFO Stack bewerkingen**

Wanneer de bits 12 t/m 15 van Of 9 (1001) bevatten, dan wordt COLL(81) gebruikt voor FIFO stack bewerkingen. De overige drie cijfers van Of geven in dit geval de grootte van de stack weer. De inhoud van BBs wordt gebruikt als stack pointer.

Wanneer de executieconditie uit is, wordt COLL(81) niet uitgevoerd. Wanneer de executieconditie aan is, schuift COLL(81) de inhoud van de woorden van de stack één positie naar beneden en schuift uiteindelijk de inhoud van BBs+1 (de waarde die het eerste op de stack gezet is) naar D. Vervolgens wordt de inhoud van BBs met 1 verlaagd.

**Opmerking** COLL(81) wordt elke cyclus uitgevoerd, tenzij de gedifferentieerde uitvoering (@COLL(81)) wordt gebruikt of COLL(81) wordt toegepast in combinatie met DIFU(13) of DIFD(14).

**Voorbeeld** Het volgende voorbeeld toont hoe COLL(81) wordt gebruikt om een stack te creëren op DM0006 t/m DM0010. DM0005 is de stack pointer.

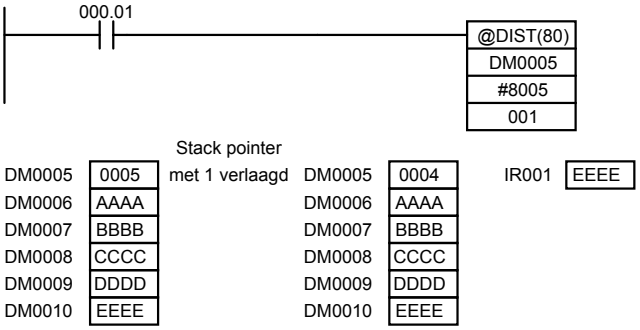



**LIFO Stack bewerkingen** Wanneer de bits 12 t/m 15 van Of 8 (1000) bevatten, dan wordt COLL(81) gebruikt voor LIFO stack bewerkingen. De overige drie cijfers van Of geven in dit geval de grootte van de stack weer. De inhoud van BBs wordt gebruikt als stack pointer.

Wanneer de executieconditie uit is, wordt COLL(81) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert COLL(81) de inhoud van het woord dat door de stackpointer wordt aangewezen (Bbs + de inhoud van BBs) naar D. Vervolgens wordt de inhoud van BBs met 1 verlaagd.

**Opmerking** COLL(81) wordt elke cyclus uitgevoerd, tenzij de gedifferentieerde uitvoering (@COLL(81)) wordt gebruikt of COLL(81) wordt toegepast in combinatie met DIFU(13) of DIFD(14).

**Voorbeeld** Het volgende voorbeeld toont hoe COLL(81) wordt gebruikt om een stack te creëren op DM0006 t/m DM0010. DM0005 is de stack pointer.




**Vlaggen**

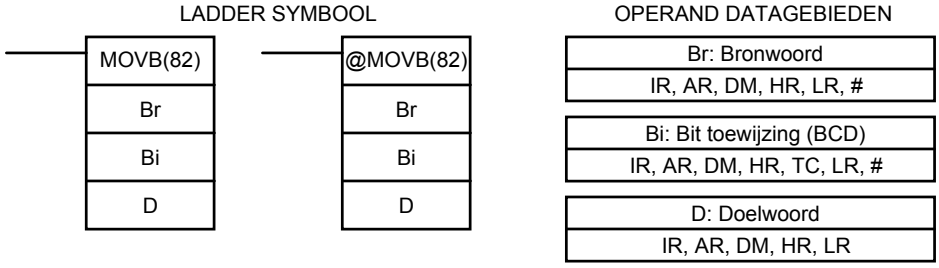
**ER:** De offset data is niet opgegeven in BCD of wanneer het opgeteld wordt bij BBs ligt het resultaat adres buiten het datagebied van BBs.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

Tijdens een stack bewerking is de waarde van de stack pointer groter geworden dan de lengte van de stack.

**EQ:** Aan wanneer de inhoud van D nul is, anders uit.

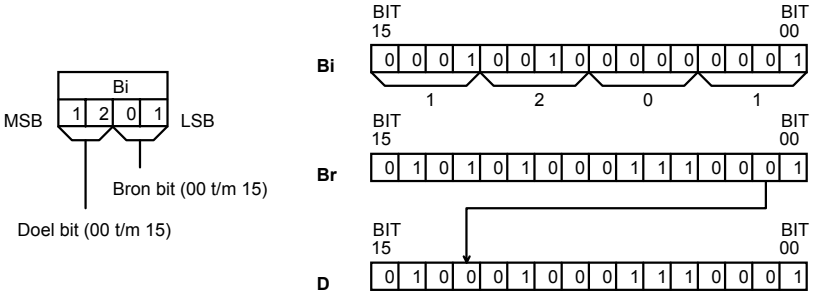
### *4.14.8 Verplaats bit - MOVB(82)*




**Beperkingen** De rechter en linker twee cijfers van Bi moeten elk tussen 00 en 15 liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor Bi en D.

**Omschrijving**

Wanneer de executieconditie uit is, wordt MOVB (82) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert MOVB(82) het gespecificeerde bit uit Br naar het gespecificeerd bit in D. De te gebruiken bits in S en D worden gespecificeerd door Bi. De rechter twee cijfers van Bi geven het bron bit aan en de twee linker cijfers het doel bit.




**Vlaggen**

**ER:** Bi is niet opgegeven in BCD of specificeert een niet bestaand bit (d.w.z. een bitnummer moet tussen 00 en 15 liggen).

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

## *4.14.9 Verplaats digit - MOVD(83)*

LADDER SYMBOOL

| MOVD(83) | @MOVD(83) |
|---|---|
| Br | Br |
| Di | Di |
| D | D |

OPERAND DATAGEBIEDEN

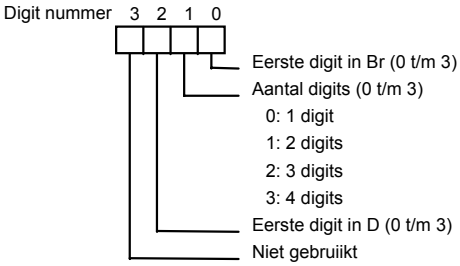
| |
|---|
| Br: Bronwoord |
| IR, AR, DM, HR, TC, LR, # |
| Di: Digit toewijzing (BCD) |
| IR, AR, DM, HR, TC, LR, # |
| D: Doelwoord |
| IR, AR, DM, HR, TC, LR |

**Beperkingen**

De drie meest rechter digits van Di moeten tussen 0 en 3 liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor Di of D.

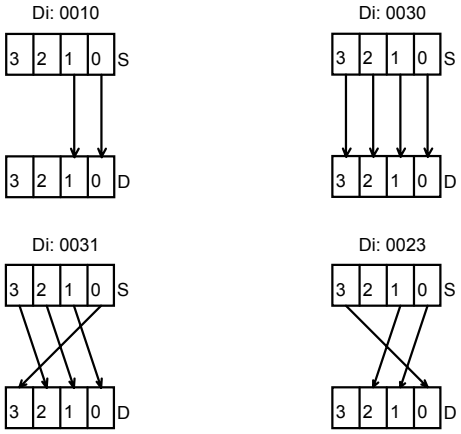
**Omschrijving**

Wanneer de executieconditie uit is, wordt MOVD(83) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert MOVD(83) de inhoud van de gespecificeerde digit(s) in S naar de gespecificeerde digits(s) in D. Tot aan vier digits kunnen in één keer worden verplaatst. Het eerste digit dat gekopieerd moet worden, het aantal digits dat gekopieerd moet worden en het eerste digit van het doelwoord waar de kopie ontvangen moet worden, wordt zoals hieronder getoond opgegeven op Di. Digits van Br worden gekopieerd naar opeenvolgende digits van D, waarbij gestart wordt op het eerst toegewezen digit en het opgegeven aantal digits aaneen wordt verplaatst. Als het laatste digit is bereikt in Br of D dan worden de volgende digits gekopieerd vanaf digit 0.




**Digit toewijzing**

Hieronder zijn een aantal voorbeelden getoond van dataverplaatsingen met verschillende waarden voor Di.




**Vlaggen**

**ER:** De waarde van minimaal één van de drie meest rechter digits van Di ligt niet tussen 0 en 3.

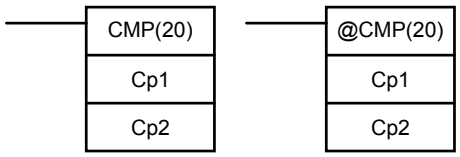
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

## 4.15 Datavergelijking

Hier vind u een beschrijving van de instructies die gebruikt kunnen worden voor het vergelijken van data. Bijvoorbeeld wordt CMP(20) gebruikt om de inhoud te vergelijken van twee woorden; BCMP(68) wordt gebruikt om te bepalen binnen welke van een aantal opgegeven bereiken de inhoud van één woord ligt; en TCMP(85) wordt gebruikt om te vergelijken of één woord gelijk is aan een aantal ingestelde waarden.

### *4.15.1 Vergelijken - CMP(20)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Cp1: Eerste te vergelijken woord |
|---|
| IR, AR, DM, HR, TC, TR, # |

| Cp2: Tweede te vergelijken woord |
|---|
| IR, AR, DM, HR, TC, LR, # |

**Beperkingen**

Wanneer de actuele waarde van een timer of counter wordt vergeleken, moet de andere waarde opgegeven worden in BCD.

**Omschrijving**

Wanneer de executieconditie uit is, wordt CMP(20) niet uitgevoerd. Wanneer de executieconditie aan is, vergelijkt CMP(20) Cp1 en Cp2 en zet het resultaat weg op de GR, EQ en LE vlaggen in het SR gebied.

**Voorzorgen**

Het plaatsen van instructies tussen de CMP(20) instructie en de verwerking van de EQ, LE en GR vlaggen kan de status van deze vlaggen veranderen. Wees er zeker van dat de status van deze vlaggen wordt uitgelezen voordat de status is veranderd.

De CMP(20) instructie vergelijkt de inhoud van twee woorden met elkaar. De inhoud van een woord kan een BCD of hexadecimaal getal zijn of een willekeurig patroon van zestien bits. De gebruiker zal zelf moeten controleren dat er bijvoorbeeld geen hexadecimale getallen met BCD getallen worden vergeleken.

**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan als Cp1 gelijk is aan Cp2.

**LE:** Aan als Cp1 kleiner is dan Cp2.

**GR:** Aan als Cp1 groter is dan Cp2.

**Opmerking**

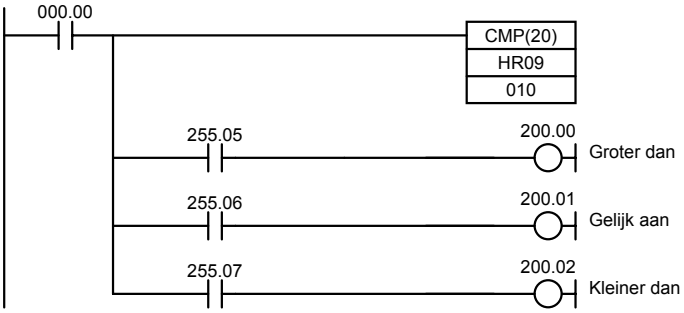
Door een geïnverteerd contact te programmeren van de drie resultaat vlaggen kunnen de vergelijkingen niet gelijk, kleiner of gelijk en groter of gelijk gemaakt worden.

| vlag | Adres | C1 < C2 | C1 = C2 | C1 > C2 |
|---|---|---|---|---|
| GR | 255.05 | UIT | UIT | AAN |
| EQ | 255.06 | UIT | AAN | UIT |

| vlag | Adres | C1 < C2 | C1 = C2 | C1 > C2 |
|---|---|---|---|---|
| LE | 255.07 | AAN | UIT | UIT |

**Voorbeeld 1: Opslaan van CMP(20) resultaten**

Het volgende voorbeeld toont hoe de resultaten van een vergelijking opgeslagen kunnen worden voor later gebruik. Als de inhoud van HR09 groter is dan die van 010, wordt 200.00 aan gezet; als de inhoud van beide woorden gelijk is, wordt 200.01 aan gezet; als de inhoud van HR09 kleiner is dan dat van 010, wordt 200.02 aan gezet. Dit natuurlijk zolang de voorwaarde voor de CMP(20) instructie aan is. In de meeste applicaties is maar één van de drie onderstaande out instructies nodig, hierdoor is het gebruik van een TR relais niet noodzakelijk.

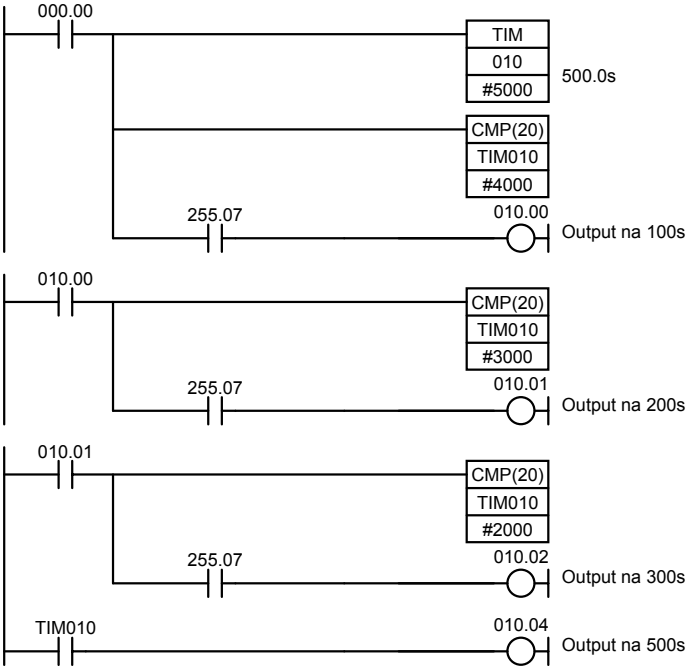



| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | OUT | TR0 | |
| 00002 | CMP(20) | HR09 | 10 |
| 00003 | AND | 255.05 | |
| 00004 | OUT | 200.00 | |
| 00005 | LD | TR0 | |
| 00006 | AND | 255.06 | |
| 00007 | OUT | 200.01 | |
| 00008 | LD | TR0 | |
| 00009 | AND | 255.07 | |
| 00010 | OUT | 200.02 | |

**Voorbeeld 2: Verkrijgen van indicaties tijdens de werking van een timer**

Het volgende voorbeeld gebruikt TIM, CMP(20) en de LE vlag (255.07) om outputs te generen op bepaalde intervallen tijdens het aftellen van de timer. De timer wordt gestart door 000.00 aan te zetten. Wanneer 000.00 uit is, wordt TIM010 gereset en de CMP(20) instructies niet uitgevoerd (d.w.z. ze worden wel uitgevoerd, maar met uit executiecondities). Output 010.00 wordt aangezet na 100 seconden (#5000-#4000); output 010.01 na 200 seconden; output 010.02 na 300 seconden; en output 010.04 na 500 seconden.

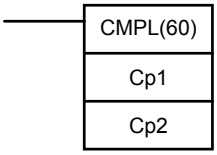
Omdat alle vergelijkingen de actuele waarde van de timer als referentie gebruiken moet de andere operand van de CMP(20) instructie in BCD opgegeven worden.



| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | TIM | 010 | #5000 |
| 00002 | CMP(20) | TIM010 | #4000 |
| 00003 | AND | 255.07 | |
| 00004 | OUT | 010.00 | |
| 00005 | LD | 010.00 | |
| 00006 | CMP(20) | TIM010 | #3000 |
| 00007 | AND | 255.07 | |
| 00008 | OUT | 010.01 | |
| 00009 | LD | 010.01 | |
| 00010 | CMP(20) | TIM010 | #2000 |
| 00011 | AND | 255.07 | |
| 00012 | OUT | 010.02 | |
| 00013 | LD | TIM010 | |
| 00014 | OUT | 010.04 | |

### *4.15.2 Dubbel vergelijken - CMPL(60)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Cp1: Eerste woord van eerste paar woorden |
|---|
| IR, AR, DM, HR, TC, TR |

| Cp2: Eerste woord van tweede paar woorden |
|---|
| IR, AR, DM, HR, TC, LR |

**Beperkingen**

Cp1 en Cp1+1 moeten in hetzelfde datagebied liggen.

**Omschrijving**

Wanneer de executieconditie uit is, wordt CMPL(60) niet uitgevoerd. Wanneer de executieconditie aan is, verbindt CMPL(60) de viercijferige hexadecimale inhoud van Cp1+1 met die van Cp1 en die van Cp2+1 met die van Cp2 om twee achtcijferige hexadecimale getallen te creëren, Cp+1,Cp1 en Cp2+1,Cp2. De twee achtcijferige getallen worden vervolgens vergeleken en het resultaat hiervan wordt op de GR, EQ en LE vlaggen in het SR gebied geplaatst.

**Voorzorgen**

Het plaatsen van instructies tussen de CMPL(60) instructie en de verwerking van de EQ, LE en GR vlaggen kan de status van deze vlaggen veranderen. Wees er zeker van dat de status van deze vlaggen wordt uitgelezen voordat de status is veranderd.

**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan als Cp1+1,Cp1 gelijk is aan Cp2+1,Cp2.

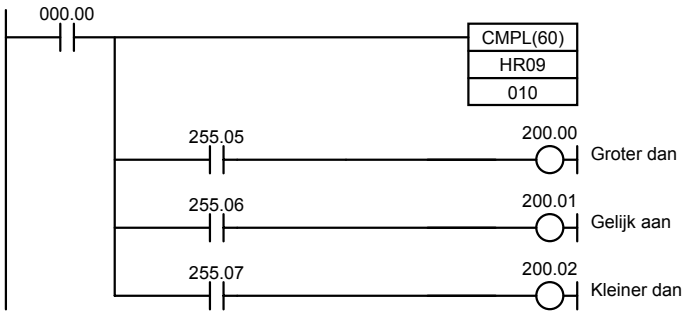
**LE:** Aan als Cp1+1,Cp1 kleiner is dan Cp2+1,Cp2.

**GR:** Aan als Cp1+1,Cp1 groter is dan Cp2+1,Cp2.

**Voorbeeld 1: Opslaan van CMPL(60) resultaten**

Het volgende voorbeeld toont hoe de resultaten van een vergelijking opgeslagen kunnen worden voor later hergebruik. Als de inhoud van HR09 en HR10 groter is dan dat van 010 en 011, wordt 200.00 aan gezet; als de inhoud van beide dubbel woorden gelijk is, wordt 200.01 aangezet; als de inhoud van HR09 en HR10 kleiner is dan dat van 010 en 011, wordt 200.02 aan gezet. Dit natuurlijk zolang de voorwaarde voor de CMPL(20) instructie aan is. In de meeste applicaties is maar één van de drie onderstaande out instructies nodig, hierdoor is het gebruik van een TR relais niet noodzakelijk.

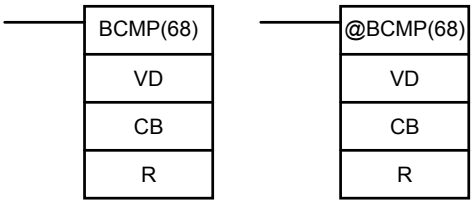
Voor de goede opletter is het onderstaande voorbeeld identiek aan dat van de CMP(20) instructie, aangezien de werking van beide instructies min of meer identiek is. Het enige verschil is, dat CMPL(60) met getallen van dubbele lengte werkt.



| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | OUT | TR0 | |
| 00002 | CMPL(60) | HR09 | 010 |
| 00003 | AND | 255.05 | |
| 00004 | OUT | 200.00 | |
| 00005 | LD | TR0 | |
| 00006 | AND | 255.06 | |
| 00007 | OUT | 200.01 | |
| 00008 | LD | TR0 | |
| 00009 | AND | 255.07 | |
| 00010 | OUT | 200.02 | |

### *4.15.3 Bereiken vergelijken - BCMP(68)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| |
|---|
| VD: Te vergelijken data |
| IR, DM, HR, TC, LR, # |
| CB: Eerste woord van reeks bereiken |
| IR, DM, HR, TC, LR |
| R: Resultaatwoord |
| IR, AR, DM, HR, TC, LR |

**Beperkingen**

Elke lage limiet in de te vergelijken reeks moet kleiner of gelijk zijn aan de bijbehorende hoge limiet. DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving**

Wanneer de executieconditie uit is, wordt BCMP(68) niet uitgevoerd. Wanneer de executieconditie aan is, vergelijkt BCMP(68) VD met de bereiken gedefinieerd in een blok dat bestaat uit CB, CB+1, CB+2, ..., CB+31. Elk bereik wordt gedefinieerd door twee woorden, het eerste woord voorziet in de lage limiet, het tweede woord voorziet in de hoge limiet. Wanneer VD binnen één of meerdere van deze bereiken ligt (inclusief de hoge en lage limiet zelf) dan wordt het corresponderende bit in R geset. De vergelijkingen die worden gemaakt en het corresponderende bit in R dat ervoor wordt geset (voor elke vergelijking die waar is) wordt hieronder getoond. De overige bits uit R (vergelijkingen die niet waar zijn) worden uit gezet.

| | |
|---|---|
| CB ≤ CD ≤ CB+1 | Bit 00 |
| CB+2 ≤ CD ≤ CB+3 | Bit 01 |
| CB+4 ≤ CD ≤ CB+5 | Bit 02 |
| CB+6 ≤ CD ≤ CB+7 | Bit 03 |
| CB+8 ≤ CD ≤ CB+9 | Bit 04 |
| CB+10 ≤ CD ≤ CB+11 | Bit 05 |
| CB+12 ≤ CD ≤ CB+13 | Bit 06 |
| CB+14 ≤ CD ≤ CB+15 | Bit 07 |
| CB+16 ≤ CD ≤ CB+17 | Bit 08 |
| CB+18 ≤ CD ≤ CB+19 | Bit 09 |
| CB+20 ≤ CD ≤ CB+21 | Bit 10 |
| CB+22 ≤ CD ≤ CB+23 | Bit 11 |
| CB+24 ≤ CD ≤ CB+25 | Bit 12 |
| CB+26 ≤ CD ≤ CB+27 | Bit 13 |
| CB+28 ≤ CD ≤ CB+29 | Bit 14 |
| CB+30 ≤ CD ≤ CB+31 | Bit 15 |

**Vlaggen**

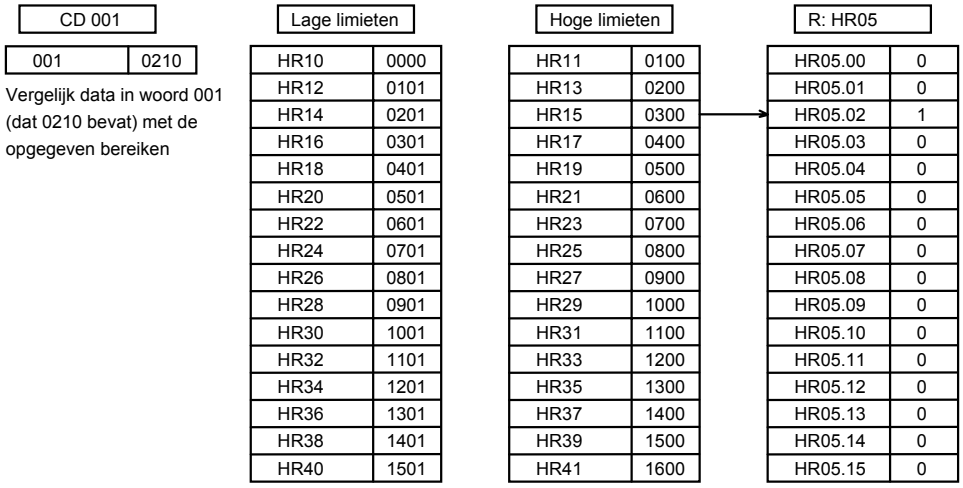
**ER:** De reeks bereiken (d.w.z. CB t/m CB+31) overschrijdt de grootte van het datagebied.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

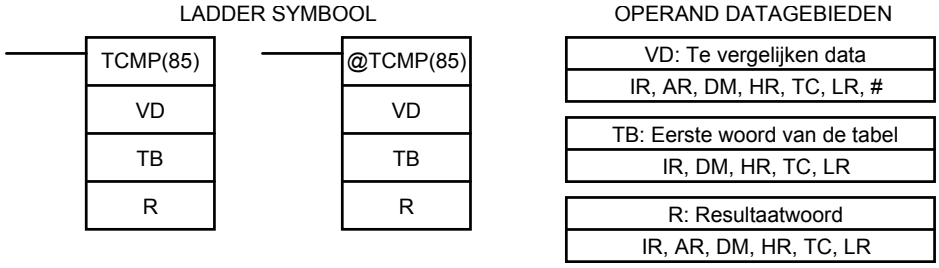
**Voorbeeld**

Het volgende voorbeeld toont de vergelijkingen die uitgevoerd zijn en de resultaten ervan voor een BCMP(68) instructie. In dit geval wordt de vergelijking elke scan uitgevoerd zolang de executie conditie (000.00) aan is.


| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | BCMP(68) | 001 | HR10 | HR05 |




## *4.15.4 Tabel vergelijken - TCMP(85)*



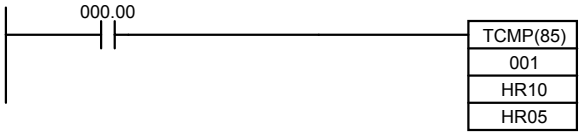

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R. TB t/m TB+15 moeten in hetzelfde datagebied liggen.

**Omschrijving** Wanneer de executieconditie uit is, wordt TCMP(85) niet uitgevoerd. Wanneer de executieconditie aan is, vergelijkt TCMP(85) VD met de inhoud van TB, TB+1, TB+2, ... en TB+15. Als VD gelijk is aan de inhoud van één van deze woorden, dan wordt het corresponderende bit in R geset. D.w.z., als VD gelijk is aan TB, dan wordt bit 00 aan gezet, als het gelijk is aan TB+1, dan wordt bit 01 aan gezet, etc. De rest van de bits in R zal uit gezet worden.

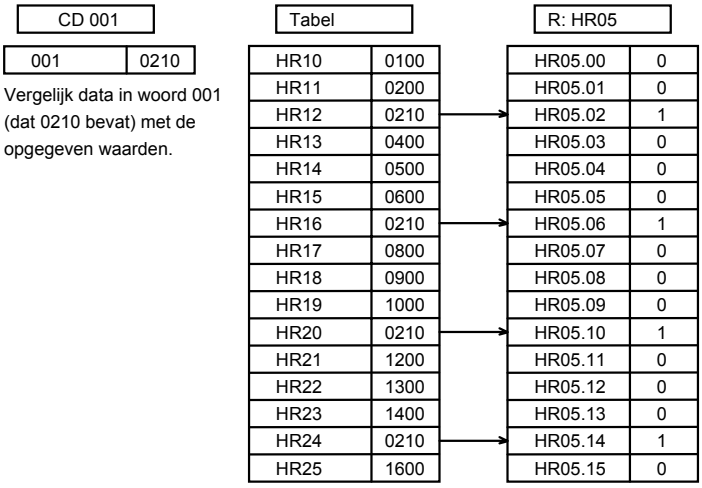
**Vlaggen**

**ER:** De vergelijkingstabel (d.w.z. TB t/m TB+15) overschrijdt het datagebied.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**Voorbeeld** Het volgende voorbeeld toont de vergelijkingen die uitgevoerd zijn en de resultaten ervan voor een TCMP(85) instructie. In dit geval wordt de vergelijking elke scan uitgevoerd zolang de executie conditie (000.00) aan is.



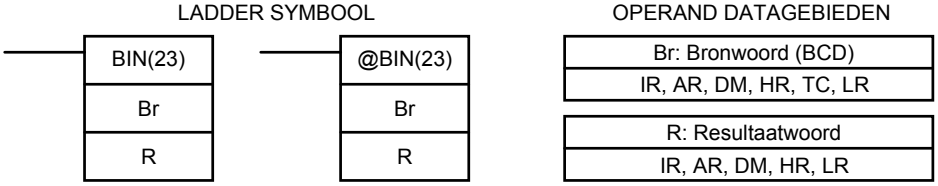

| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | TCMP(85) | 001 | HR10 | HR05 |




## 4.16 Dataconversie

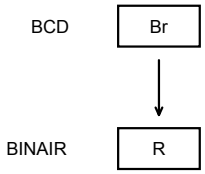
De conversie instructies zetten woorddata van het ene formaat om naar het andere en plaatsen de geconverteerde data op het gespecificeerde resultaatwoord. Er zijn instructies beschikbaar om tussen binair (hexadecimaal) en BCD, naar 7-segments display data, naar ASCII en tussen gemultiplexte en niet gemultiplexte data te converteren. Al deze instructies veranderen alleen de inhoud van de woorden waarnaar de omgezette data verplaatst wordt. D.w.z., de inhoud van bronwoorden is hetzelfde voor en na de uitvoer van een dataconversie instructie.

### *4.16.1 BCD naar binair - BIN(23)*




**Beperkingen** Br moet worden opgegeven in BCD. DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, wordt BIN(23) niet uitgevoerd. Wanneer de executieconditie aan is, zet BIN(23) de BCD inhoud van Br om naar het numerieke binaire equivalent en plaatst de binaire waarde op R. Alleen de inhoud van R wordt verandert; de inhoud van Br blijft onveranderd.



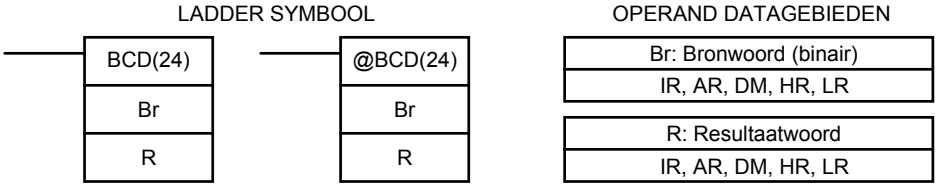

BIN(23) kan gebruikt worden om BCD getallen naar binair om te zetten. Binaire getallen worden door sommige rekenkundige instructies in de PLC gebruikt en de meeste analoge uitgangskaarten worden binair aangestuurd.

**Vlaggen**

**ER:** De inhoud van Br is niet in BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat van de omzetting gelijk is aan nul.

### *4.16.2 Binair naar BCD - BCD(24)*

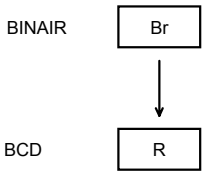


**Beperkingen**

Als de inhoud van Br boven de 270F komt zal de omgezette waarde ervan meer zijn dan 9999 en wordt BCD(24) niet uitgevoerd. Wanneer de instructie niet wordt uitgevoerd blijft de inhoud van R onveranderd. DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving**

BCD(24) zet de binaire (hexadecimale) inhoud van Br om naar het numerieke BCD equivalent en plaatst dit BCD resultaat op R. Alleen de inhoud van R wordt veranderd, de inhoud van Br blijft onveranderd.




BCD(24) kan gebruikt worden om binaire getallen naar BCD om te zetten zodat bijvoorbeeld de uitlezing in SYSWIN, op de programmingconsole of elk ander programmeerapparaat in decimaal verschijnt in plaats van in hexadecimaal. Het kan ook gebruikt worden om getallen naar BCD te om te zetten om rekenkundige bewerkingen in BCD uit te kunnen voeren i.p.v. in binair.

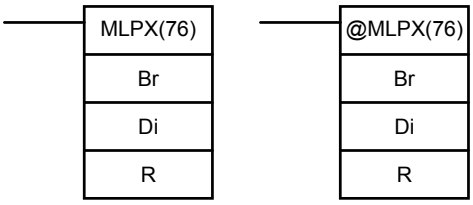
**Vlaggen**

**ER:** Br is groter dan 270F.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan nul.

### *4.16.3 4 naar 16 decoder - MLPX(76)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Br: Bronwoord |
|---|
| IR, AR, DM, HR, TC, LR |

| Di: Digit designator |
|---|
| IR, AR, DM, HR, TC, LR, # |

| R: Eerste resultaatwoord |
|---|
| IR, AR, DM, HR, LR |

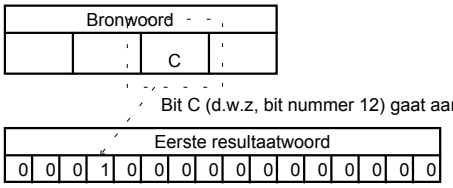
**Beperkingen**

De twee meest rechter digits van Di moeten beide een waarde bevatten die tussen de 0 en 3 ligt. Alle resultaat woorden moeten in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving**

Wanneer de executieconditie uit is, wordt MLPX(76) niet uitgevoerd. Wanneer de executieconditie aan is, zet MLPX(76) tot aan vier digits uit Br om naar een decimale waarde van 0 t/m 15 en elke omgezette decimale waarde wordt gebruikt om een bit positie aan te duiden. Het bit waarvan het nummer overeenkomt met de omgezette waarde wordt vervolgens aan gezet in een resultaatwoord. Als meer dan 1 digit is gespecificeerd op Di, dan wordt één bit aan gezet op een aantal opeenvolgende woorden, beginnend met R (Zie de voorbeelden hieronder).

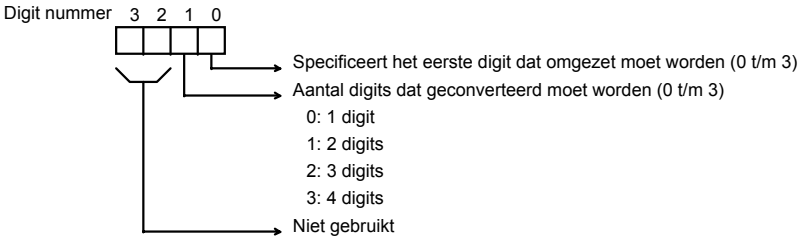
Het volgende is een voorbeeld van een één digit decodering van digit 1 uit Br. In dit geval zou Di 0001 zijn.



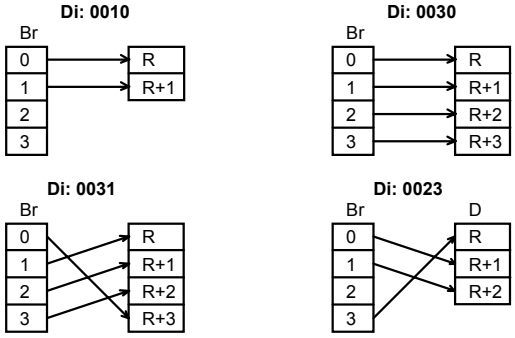

Het eerste digit en het aantal digits dat omgezet moet worden is vastgelegd op Di. Als meer digits worden toegewezen dan aanwezig zijn in Br (tel vanaf het door Di toegewezen start digit), dan worden de overige digits genomen vanaf het begin van Br. Het laatste woord dat nodig is om het geconverteerde resultaat op te slaan (R plus het aantal digits dat omgezet moet worden) moet in hetzelfde datagebied liggen als R. D.w.z. dat wanneer twee digits geconverteerd moeten worden het laatste woord adres uit een datagebied niet kan worden toegewezen aan R; als drie digits worden omgezet kunnen de laatste twee woorden uit een datagebied niet worden toegewezen aan R.

**Digit designator**

De digits van Di worden ingesteld zoals hieronder getoond is.




Een aantal voorbeeld waarden van Di en de conversies die geproduceerd worden zijn hieronder getoond.



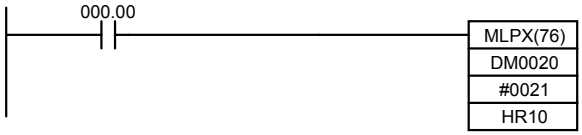

**Vlaggen**

**ER:** Een niet gedefinieerde digit designator is gebruikt of R plus het aantal digits overschrijdt de grootte van het datagebied.

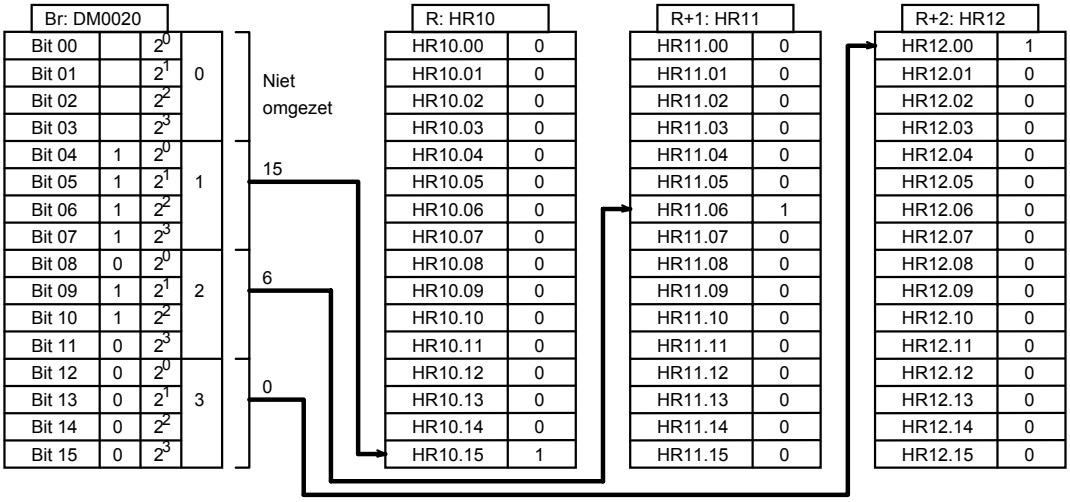
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**Voorbeeld**

Het volgende programma zet drie digits uit DM0020 om naar bitposities en zet de bijbehorende bits aan in drie opeenvolgende woorden, startend bij HR10.

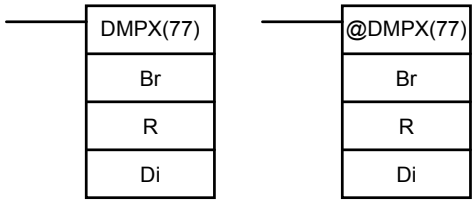



| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | MLPX(76) | DM0020 | #0021 | HR10 |



### *4.16.4 16 naar 4 encoder - DMPX(77)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| |
|---|
| Br: eerste bronwoord |
| IR, AR, DM, HR, TC, LR |
| R: Resultaatwoord |
| IR, AR, DM, HR, LR |
| Di: Digit designator |
| IR, AR, DM, HR, TC, LR, # |

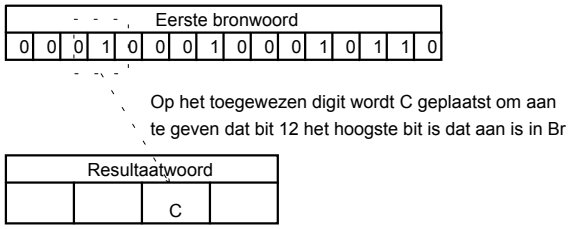
**Beperkingen**

De waarde van de twee meest rechter digits van Di moeten elke tussen 0 en 3 liggen. Alle bronwoorden moeten in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor Br, R of Di.

**Omschrijving**

Wanneer de executieconditie uit is, wordt DMPX(77) niet uitgevoerd. Wanneer de executieconditie aan is, bepaalt DMPX(77) de positie van het hoogste bit in Br dat aan is en codeert dit naar een één cijferige hexadecimale waarde die overeenkomt met het nummer van dit bit. De gecodeerde hexadecimale waarde wordt vervolgend op het in R gespecificeerde digit geplaatst. De digits waarop het resultaat geplaatst moet worden en het aantal woorden dat gecodeerd moet worden wordt gespecificeerd op Di.

Het volgende is een voorbeeld van een één digit codeerbewerking waarvan het resultaat op digit 1 van R wordt geplaatst. De waarde op Di moet in dit geval 0001 zijn.

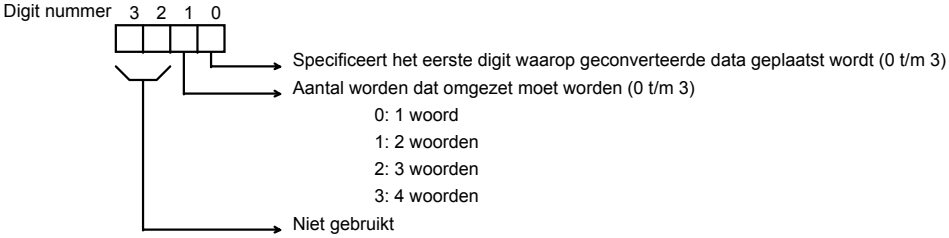



Tot aan vier digits kunnen gecodeerd worden uit maximaal vier opeenvolgende woorden, startend bij Br. De gecodeerde cijfers worden geplaatst op R, beginnend bij het in Di toegewezen digit. Het eerste digit en het aantal digits dat omgezet moet worden is vastgelegd op Di. Als meer digits worden benoemd dan aanwezig zijn in R (tel vanaf het door Di toegewezen start digit), dan worden de overige digits benoemd vanaf het begin van R.

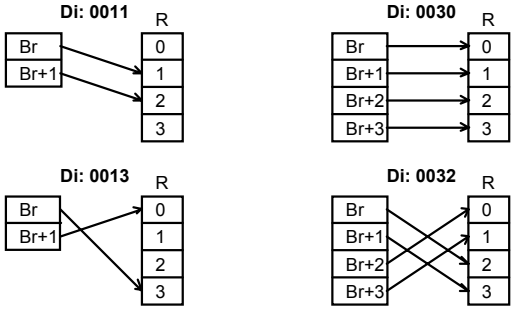
Het laatste woord dat geconverteerd moet worden (Br plus het aantal digits dat omgezet moet worden) moeten in hetzelfde datagebied liggen als Br.

**Digit designator**

De digits van Di moeten worden ingesteld zoals hieronder getoond is.




Een paar voorbeelden van Di waarden en bijbehorende woord naar digit conversies worden hieronder getoond.

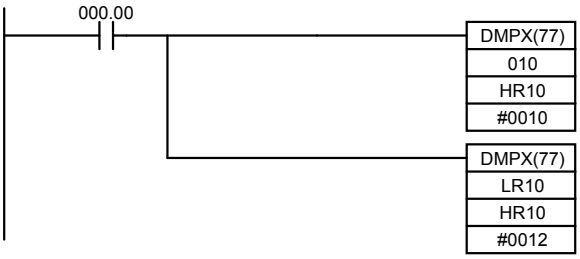


**Vlaggen**

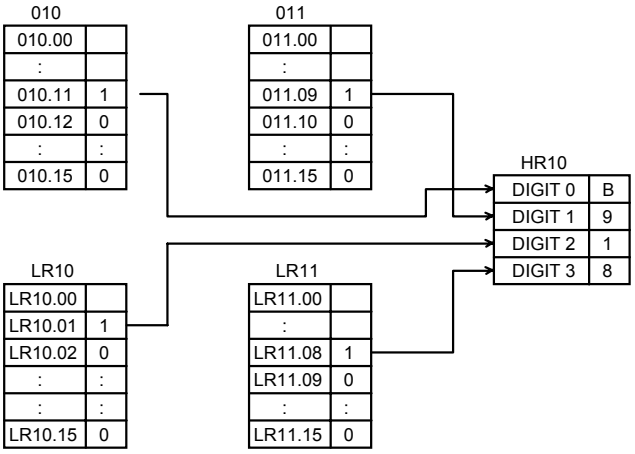
**ER:** Een niet gedefinieerde digit designator is gebruikt of Br plus het aantal digits overschrijdt de grootte van het datagebied.
De inhoud van het bronwoord is nul.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**Voorbeeld**

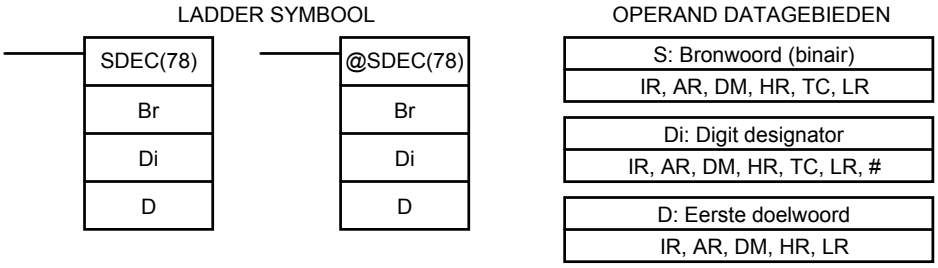
Wanneer 000.00 aan is, codeert het volgende diagram woord 010 en 011 naar de eerste twee digits van HR10 en vervolgens LR10 en LR11 naar de laatste twee digits van HR10. Alhoewel de gehele status van elke bronwoord niet getoond is kan ervan uitgegaan worden dat het bit met de status 1 dat gebruikt wordt in het voorbeeld het hoogste bit dat aan is in het woord.




| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | DMPX(77) | 010 | HR10 | #0010 |
| 00002 | DMPX(77) | LR10 | HR10 | #0012 |




## *4.16.5 7 segment decoder - SDEC(78)*




**Beperkingen**

Di moet binnen de toegestane grenzen liggen. Alle doelwoorden moeten in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor D.

**Omschrijving**

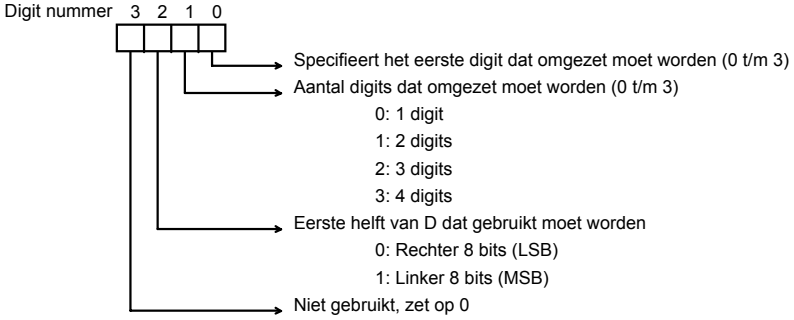
Wanneer de executieconditie uit is, wordt SDEC(78) niet uitgevoerd. Wanneer de executieconditie aan is, zet SDEC(78) de aangeduide digits uit Br naar het equivalente 8-bits 7-segment display code om en plaatst dit op de doelwoorden, beginnend bij D.

Eén of alle digits uit Br kunnen worden omgezet, op volgorde vanaf het eerste toegewezen digit. Het eerste digit, het aantal digits dat omgezet moet worden en de helft van D waarop de eerste 7 segment display code geplaatst moet worden (meest rechter of linker 8 bits of byte) wordt bepaald door Di. Wanneer meer dan één digit wordt omgezet dan worden de geconverteerde digits op opeenvolgende bytes geplaatst vanaf het toegewezen begin byte op D. Elke 7 segmentscode

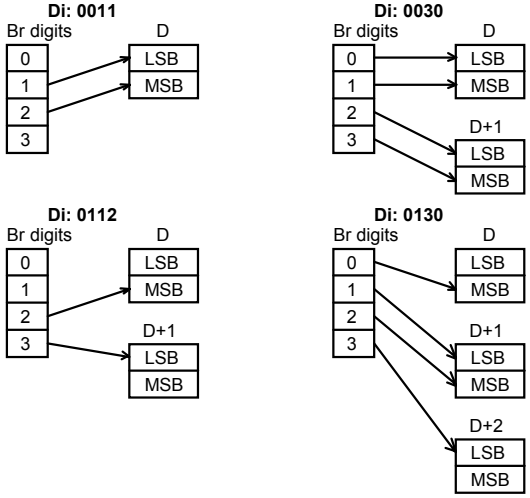
bestaat uit 1 byte. Als meer digits worden benoemd dan aanwezig zijn in R (tel vanaf het door Di toegewezen start digit), dan worden de overige digits benoemd vanaf het begin van R.

**Digit designator**

De digits van Di worden ingesteld zoals hieronder getoond is.

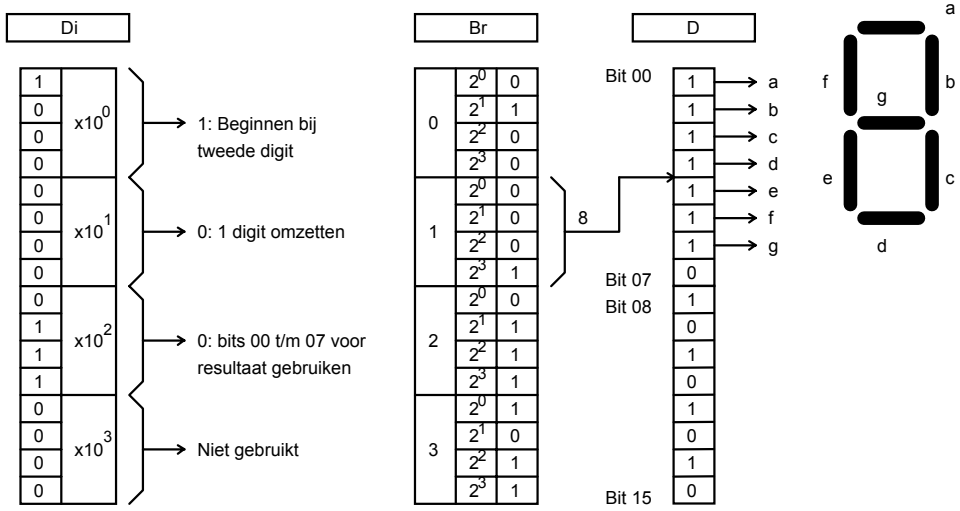



Een paar voorbeelden van Di waarden en bijbehorende 4 bit binair naar 7 segment code conversies worden hieronder getoond.




**Voorbeeld**

Het volgende voorbeeld toont de data om een 8 te genereren. De lower case letters tonen welk bit correspondeert met welk segment van het 7 segments display. De tabel hieronder toont de originele data en omgezette code voor alle hexadecimale cijfers.




| Originele data | | | | | Omgezette code (segments) | | | | | | | | Displ. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Digit | Bits | | | | - | g | f | e | d | c | b | a | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 2 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 3 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 4 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 5 |

| Originele data | | | | | Omgezette code (segments) | | | | | | | | Displ. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Digit | Bits | | | | - | g | f | e | d | c | b | a | |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 6 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 7 |
| 8 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| 9 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 9 |
| A | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | A |
| B | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | b |
| C | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | C |
| D | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | d |
| E | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | E |
| F | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | F |

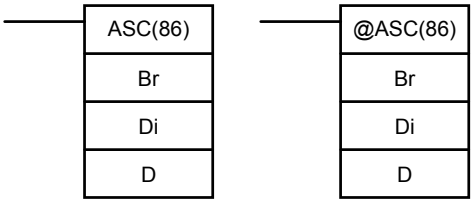
**Vlaggen**

**ER:** Incorrecte digit designator of grootte van het datagebied voor het resultaat is overschreden.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

### *4.16.6 ASCII conversie - ASC(86)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Br: Bronwoord |
|---|
| IR, AR, DM, HR, TC, LR |

| Di: Digit designator |
|---|
| IR, AR, DM, HR, TC, LR, # |

| D: Eerste doelwoord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

Alle bestemmingswoorden moeten in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor D.

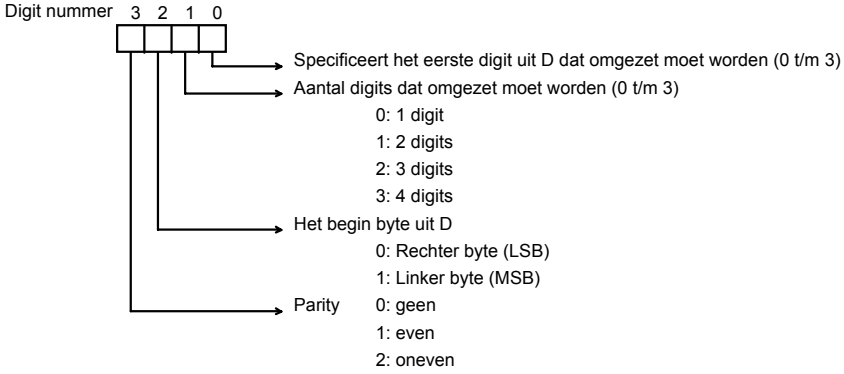
**Omschrijving**

Wanneer de executieconditie uit is, wordt ASC(86) niet uitgevoerd. Wanneer de executieconditie aan is, zet ASC(86) de aangeduide digits uit Br om naar de equivalente 8 bit ASCII code en plaatst dit op de doelwoorden beginnend bij D.

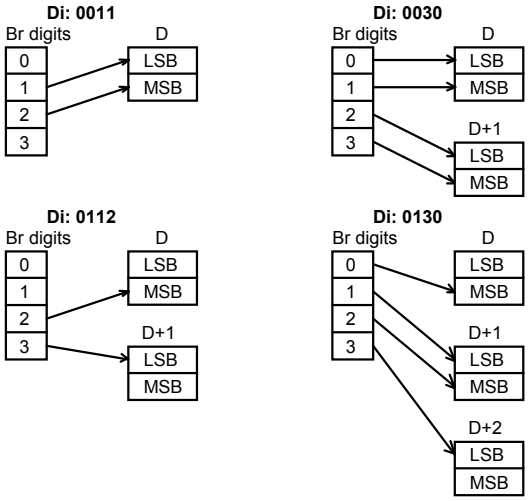
Eén of alle digits uit Br kunnen worden geconverteerd, op volgorde vanaf het eerste aangeduide digit. Het eerste digit, het aantal digits dat omgezet moet worden en het byte van D waarop de eerste ASCII code geplaatst moet worden, wordt bepaald door Di. Wanneer meer dan één digit wordt omgezet dan worden de geconverteerde digits op opeenvolgende bytes geplaatst vanaf het aangeduide begin byte op D. Elke ASCII code bestaat uit 1 byte. Als meer digits worden aangeduid dan aanwezig zijn in R (tel vanaf het door Di aangeduide start digit), dan worden de overige digits genomen vanaf het begin van D.

**Digit designator**

De digits van Di worden ingesteld zoals hieronder getoond is.




Een paar voorbeelden van Di waarden en bijbehorende 4 bit binair naar ASCII code conversies worden hieronder getoond.




**Pariteit**

Het linkerbit van elke ASCII character (2 digits) kan automatisch worden gecorrigeerd voor even of oneven pariteit. Als geen pariteit is aangeduid dan is dit linkerbit altijd nul.

Wanneer even pariteit wordt gekozen, dan zal het linkerbit worden aangepast zodat het aantal bits dat aan is even is. D.w.z., wanneer aangepast voor even pariteit, zal ASCII "31" (00110001) "B1" (10110001: Het pariteitsbit wordt aangezet om een even aantal aan bits te creëren) worden en ASCII "36" (00110110) zal "36" (00110110: pariteitsbit blijft uit aangezien het aantal AAN bits al even is) blijven. De status van het pariteitsbit beïnvloedt de betekenis van de ASCII code niet.

Wanneer oneven pariteit is gekozen, dan wordt het linkerbit van elk ASCII karakter aangepast zodat er een oneven aantal aan bits is.

**Vlaggen**

**ER:** Incorrecte digit designator of de grootte van het datagebied voor het resultaat is overschreden.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

## 4.17 BCD calculaties

De BCD calculatie instructies - INC(38), DEC(39), ADD(30), ADDL(54), SUB(31), SUBL(55), MUL(32), MULL(56), DIV(33), DIVL(57), FDIV(79) en ROOT(72) voeren allemaal rekenkundige bewerkingen uit op BCD data.

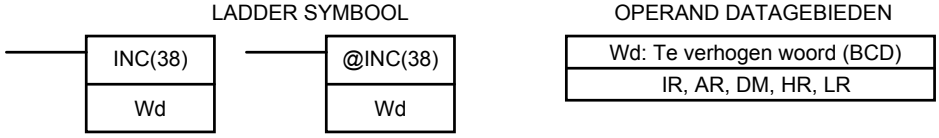
Voor INC(38) en DEC(39) zijn de bron- en resultaatwoorden hetzelfde. Dat wil zeggen, dat de inhoud van het bronwoord wordt overschreven door het resultaat van de instructie. De meeste andere instructies veranderen alleen de inhoud van de woorden waarop het resultaat wordt geplaatst. Bij de andere BCD calculatie instructies is de inhoud van bronwoorden hetzelfde voor en na de uitvoer van de instructie.

STC(40) en CLC(41), die worden gebruikt om de Carry vlag te setten en te wissen, worden eveneens in deze groep besproken omdat de meeste BCD bewerkingen gebruik maken van de Carry vlag (CY) in hun resultaat. Binaire berekeningen en schuifbewerkingen gebruiken ook CY.

**Opmerking**

De optel- en aftrekinstructies gebruiken CY zowel in de berekening als in het resultaat. Wees er zeker van dat CY gewist is als de voorgaande status niet van belang is in de berekening en gebruik het resultaat dat geplaatst is op CY, indien nodig, voor het wordt veranderd door de uitvoer van een andere instructie.

### *4.17.1 Increment - INC(38)*




**Beperkingen**

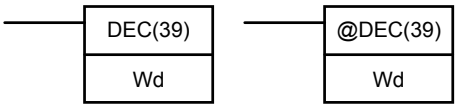
DM6144 t/m DM6655 kan niet gebruikt worden voor Wd.

**Omschrijving** Wanneer de executieconditie uit is, wordt INC(38) niet uitgevoerd. Wanneer de executieconditie aan is, verhoogt INC(38) het BCD getal op Wd met 1, zonder de Carry (CY) te beïnvloeden.

**Vlaggen**
**ER:** Wd is niet opgegeven in BCD
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
**EQ:** Aan wanneer het resultaat gelijk is aan 0 (na de bewerking 9999+1).

### *4.17.2 Decrement - DEC(39)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Wd: Te verlagen woord (BCD) |
|---|
| IR, AR, DM, HR, LR |

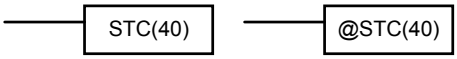
**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor Wd.

**Omschrijving** Wanneer de executieconditie uit is, wordt DEC(39) niet uitgevoerd. Wanneer de executieconditie aan is, verlaagt DEC(39) het BCD getal op Wd met 1, zonder de Carry (CY) te beïnvloeden. DEC(39) werkt op dezelfde manier als INC(38) alleen verlaagt het de waarde inplaats van verhogen.

**Vlaggen**
**ER:** Wd is niet opgegeven in BCD
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
**EQ:** Aan wanneer het resultaat gelijk is aan 0 (na de bewerking 1-1).

### *4.17.3 Zet Carry - STC(40)*

LADDER SYMBOOL

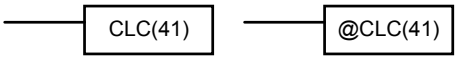



Wanneer de executieconditie uit is, wordt STC(40) niet uitgevoerd. Wanneer de executieconditie aan is, zet STC(40) het CY bit (255.04) aan.

**Vlaggen** **CY:** STC(40) zet het CY bit aan.

### *4.17.4 Wis Carry - CLC(41)*

LADDER SYMBOOL



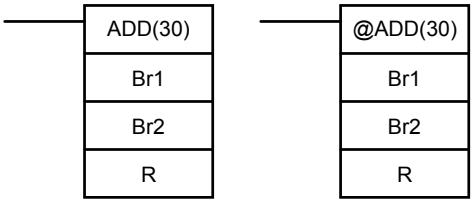

Wanneer de executieconditie uit is, wordt CLC(41) niet uitgevoerd. Wanneer de executieconditie aan is, wist CLC(41) het CY bit (255.04).

CLEAR CARRY wordt gebruikt om het Carry bit te resetten (uit zetten).

**Vlaggen** **CY:** CLC(41) zet het CY bit uit.

### *4.17.5 BCD optellen - ADD(30)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Br1: Bronwoord 1 (BCD) |
|---|
| IR, AR, DM, HR, TC, LR, # |

| Br2: Bronwoord 2 (BCD) |
|---|
| IR, AR, DM, HR, TC, LR, # |

| R: Resultaatwoord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R. Br1 en Br2 moeten in BCD worden opgegeven.

**Omschrijving** Wanneer de executieconditie uit is, wordt ADD(30) niet uitgevoerd. Wanneer de executieconditie aan is, telt ADD(30) de inhoud van Br1, Br2 en CY bij elkaar op en plaatst het resultaat op R. CY wordt geset als het resultaat groter is dan 9999.
Br1+Br2+CY→CY, R

**Carry**

CY wordt geset als het resultaat groter is dan 9999. Een optelling van 9999 + 9999 geeft als resultaat CY=1 en R=9998. Wanneer het resultaat groter is dan 9999 gaat ADD(30) verder met tellen vanaf 0.

**Vlaggen**

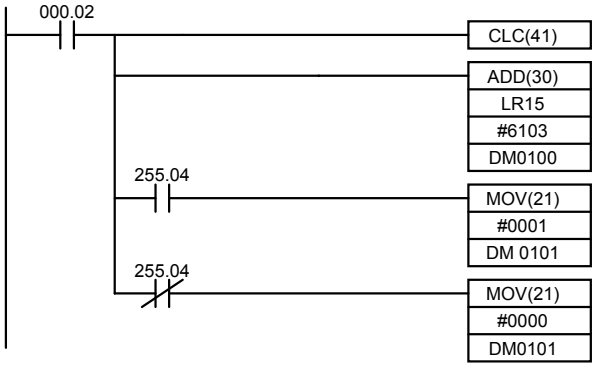
**ER:** Br1 en/of Br2 is niet BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY:** Aan wanneer er een Carry in resultaat is, anders uit.

**EQ:** Aan wanneer het resultaat 0 is.

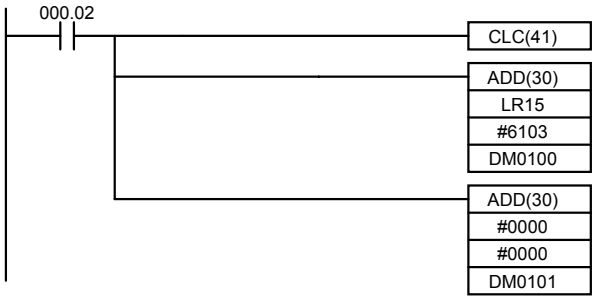
**Voorbeeld**

Als 000.02 aan is dan zal het programma, zodra het volgende diagram wordt bereikt, de Carry wissen met CLC(41), de inhoud van LR15 optellen bij een constante (#6103) en het resultaat op DM0100 plaatsen. Op DM101 wordt 0 of 1 geplaatst, afhankelijk van de status van CY (255.04). Dit zorgt ervoor dat de status van de Carry wordt bewaard en het resultaat later kan worden verwerkt als achtcijferige data.




| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LR | 000.02 | | |
| 00001 | OUT | TR0 | | |
| 00002 | CLC(41) | | | |
| 00003 | ADD(30) | LR15 | #6103 | DM0100 |
| 00004 | AND | 255.04 | | |
| 00005 | MOV(21) | #0001 | DM0101 | |
| 00006 | LD | TR0 | | |
| 00007 | AND NOT | 255.04 | | |
| 00008 | MOV(21) | #0000 | DM0101 | |

Aangezien de optelling die door ADD(30) wordt uitgevoerd Br1+Br2+CY→CY, R is, kan het bovenstaande diagram ook worden vervangen door het onderstaande.



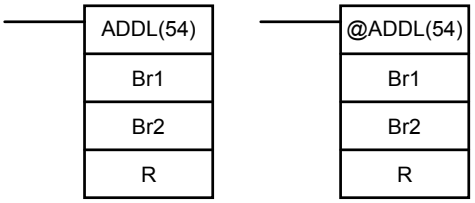

| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LR | 000.02 | | |
| 00001 | CLC(41) | | | |
| 00002 | ADD(30) | LR15 | #6103 | DM0100 |
| 00003 | ADD(30) | #0000 | #0000 | DM0101 |

Voor de eerste optelling wordt de Carry gewist. Mocht het resultaat van deze optelling groter zijn dan 9999, dan telt ADD(30) alles wat het resultaat groter is dan 10000 op bij 0 en plaatst dit op R en wordt de Carry geset. De tweede optelling zal in dit geval als optelling uitvoeren #0000 + #0000 + CY (=1) = 0001. Het resultaat 0001 wordt vervolgens op R (DM0101) geplaatst.

Alhoewel dankzij het bovenstaande principe twee ADD(30) instructies gebruikt kunnen worden om acht cijferige BCD optellingen uit te voeren is de instructie ADDL(54) speciaal toegevoegd voor dit doel.

### *4.17.6 Dubbel BCD optellen - ADDL(54)*

LADDER SYMBOOL

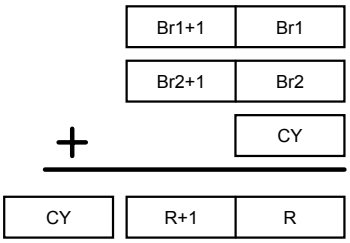



OPERAND DATAGEBIEDEN

| |
|---|
| Br1: Bronwoord 1 (BCD) |
| IR, AR, DM, HR, TC, LR |
| Br2: Bronwoord 2 (BCD) |
| IR, AR, DM, HR, TC, LR |
| R: Resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6143 t/m DM6655 kan niet gebruikt worden voor R. Br1 en Br2 moeten in BCD worden opgegeven.

**Omschrijving** Wanneer de executieconditie uit is, wordt ADDL(54) niet uitgevoerd. Wanneer de executieconditie aan is, telt ADDL(54) de inhoud van CY op bij de achtcijferige waarde in Br1+1, Br1 en de achtcijferige waarde in Br2+1, Br2 en plaatst het resultaat op R+1 en R. CY wordt geset als het resultaat groter is dan 99999999.



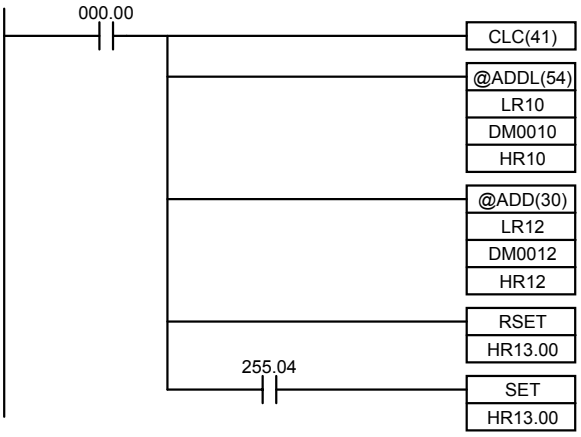

**Vlaggen**

**ER:** Br1 en/of Br2 is niet BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY:** Aan wanneer er een Carry in resultaat is, anders uit.

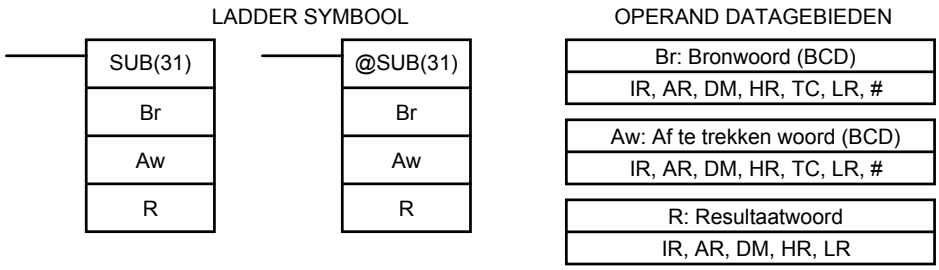
**EQ:** Aan wanneer het resultaat 0 is.

**Voorbeeld** Wanneer 00000 aan is, zal het volgende programma twee twaalfcijferige getallen bij elkaar optellen. Het eerste getal is opgeslagen in LR10 t/m LR12, het tweede getal in DM0010 t/m DM0012. Het resultaat wordt geplaatst op HR10 t/m HR13. De eerste optelling telt de lage acht van de twaalf cijfers bij elkaar op. Bij de tweede optelling worden de hoge vier cijfers bij elkaar opgeteld door gebruik te maken van ADD(30). Een Carry die eventueel gegenereerd kan worden door de eerste optelling wordt bij de tweede bij het resultaat opgeteld. De Carry van de tweede optelling wordt geplaatst op HR13.00 door gebruik van het carry (CY) bit en een RSET en SET instructie.




| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | CLC(41) | | | |
| 00002 | @ADDL(54) | LR10 | DM0010 | HR10 |
| 00003 | @ADD(30) | LR12 | DM0012 | HR12 |
| 00004 | @ADB(50) | #0000 | #0000 | HR13 |

## *4.17.7 BCD aftrekken - SUB(31)*




**Beperkingen**

DM6144 t/m DM6655 kan niet gebruikt worden voor R. Br en Aw moeten in BCD worden opgegeven.

**Omschrijving**

Wanneer de executieconditie uit is, wordt SUB(31) niet uitgevoerd. Wanneer de executieconditie aan is, trekt SUB(31) de inhoud van Aw en CY van Br af en plaatst het resultaat op R. Als het resultaat negatief is wordt CY geset en het 10's complement van het actuele resultaat geplaatst op R. Om het 10's complement resultaat te converteren naar een positieve waarde kan de inhoud van R van 0 afgetrokken worden.

Br-Aw-CY→CY, R

**Vlaggen**

**ER:** Br en/of Aw is niet BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY:** Aan wanneer het resultaat negatief is, dat is wanneer Br kleiner is dan AW plus CY.

**EQ:** Aan wanneer het resultaat 0 is.

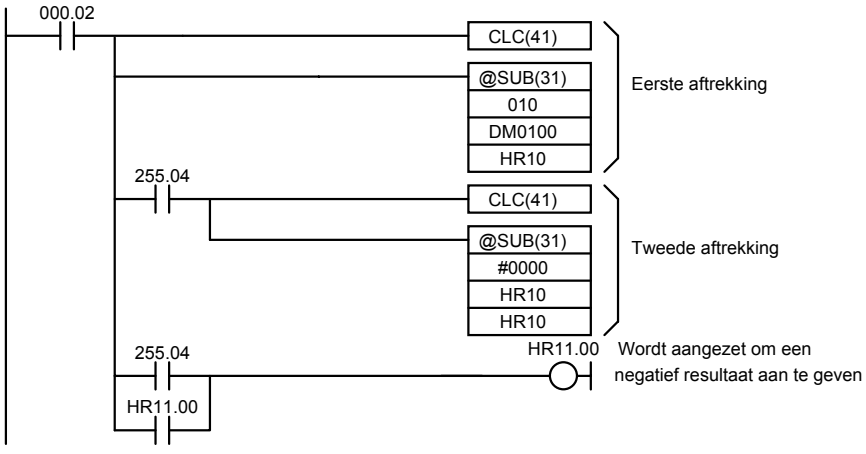
**Voorbeeld**

Wanneer 000.02 aan is, zal het volgende diagram CY wissen, de inhoud van DM0100 en CY van de inhoud van 010 aftrekken en het resultaat in HR10 opslaan.

Als CY wordt geset door de uitvoer van SUB(31), dan wordt het resultaat op HR10 van nul afgetrokken (merk op dat CLC(41) opnieuw gebruikt moet worden om een correct resultaat te krijgen) en wordt het resultaat terug geplaatst op HR10 en wordt HR11.00 aan gezet om een indicatie te bewaren van het negatieve resultaat.

Als CY niet wordt geset door de uitvoer van SUB(31), dan is het resultaat positief en wordt de tweede aftrekking niet uitgevoerd en HR11.00 niet aan gezet. HR11.00 is geprogrammeerd als een zelfhandhavend bit, zodat een verandering in de status van CY het niet uit zal zetten als het programma de volgende scan wordt uitgevoerd.

In dit voorbeeld is de gedifferentieerde uitvoering van SUB(31) zo gebruikt dat de aftrekking alleen wordt uitgevoerd op het moment dat 000.02 aangezet wordt (opgaande flank). Wanneer een andere aftrekking moet worden uitgevoerd zal 000.02 eerst voor minimaal één scan uit gezet moeten worden (waarbij ook HR11.00 gereset zal worden) en vervolgens weer aan.




| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.02 | | |
| 00001 | OUT | TR0 | | |
| 00002 | CLC(41) | | | |
| 00003 | @SUB(31) | 010 | DM0100 | HR10 |

```
00004    AND          255.04
00005    CLC(41)
00006    @SUB(31)     #0000        HR10        HR10
00007    LD           TR0
00008    AND          255.04
00009    LD           TR0
00010    AND          HR11.00
00011    OR LD
00012    OUT          HR11.00
```

De aftrekkingen uit dit diagram zijn hieronder getoond, waarbij willekeurige data is gebruikt voor 010 en DM0100.

**Opmerking** De actuele werking van SUB(31) is als volgt: Aw en CY worden van 10.000 afgetrokken en hier wordt vervolgens Br bij opgeteld. Voor positieve getallen wordt het meest linker cijfer (het vijfde) afgekapt, voor negatieve getallen wordt op deze wijze het 10s complement verkregen. De procedure voor het verkrijgen van het correcte resultaat is hieronder getoond.

**Eerste aftrekking**

| | | | |
|---|---|---|---|
| 010 | | 1029 | |
| DM0100 | - | 3452 | |
| CY | - | 0 | |
| HR10 | | 7577 | (1029+(10000-3452)) |
| CY | | 1 | (negatief resultaat). |

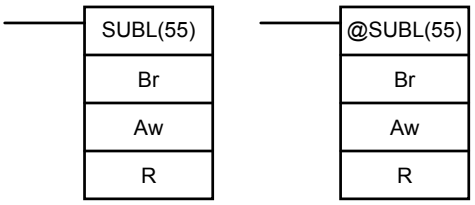
**Tweede aftrekking**

| | | | |
|---|---|---|---|
| # | | 0000 | |
| HR10 | - | 7577 | |
| CY | - | 0 | |
| HR10 | | 2423 | (0000+(10000-7577)). |
| CY | | 1 | (negatief resultaat). |

In het bovenstaande geval zal het programma HR11.00 aan zetten om aan te geven dat de waarde in HR10 negatief is.

### *4.17.8 Dubbel BCD aftrekken - SUBL(55)*

LADDER SYMBOOL

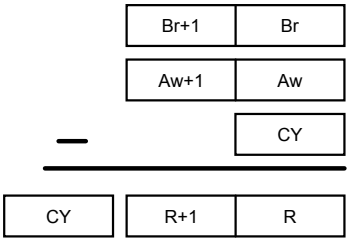



OPERAND DATAGEBIEDEN

| |
|---|
| Br: Eerste bronwoord (BCD) |
| IR, AR, DM, HR, TC, LR |
| Aw: Eerste af te trekken woord (BCD) |
| IR, AR, DM, HR, TC, LR |
| R: Eerste resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6143 t/m DM6655 kan niet gebruikt worden voor R. Br en Aw moeten in BCD worden opgegeven.

**Omschrijving** Wanneer de executieconditie uit is, wordt SUBL(55) niet uitgevoerd. Wanneer de executieconditie aan is, trekt SUBL(55) CY en de achtcijferige inhoud van Aw+1 en Aw af van de achtcijferige waarde op Br+1 en Br en plaatst het resultaat op R+1 en R. Als het resultaat negatief is, wordt CY geset en het 10's complement van het actuele resultaat opgeslagen op R+1, R. Om het 10's complement resultaat te converteren naar een positieve waarde kan de inhoud van R van 0 afgetrokken worden. Aangezien een achtcijferige constante niet direct kan worden ingevoerd, kan de BSET(71) instructie of de MOV(21) instructie (zie “verplaatsen - mov(21)” op pagina 108) gebruikt worden om een achtcijferige constante te creëren.




**Vlaggen** **ER:** Br+1, Br en/of Aw+1, Aw is niet BCD.

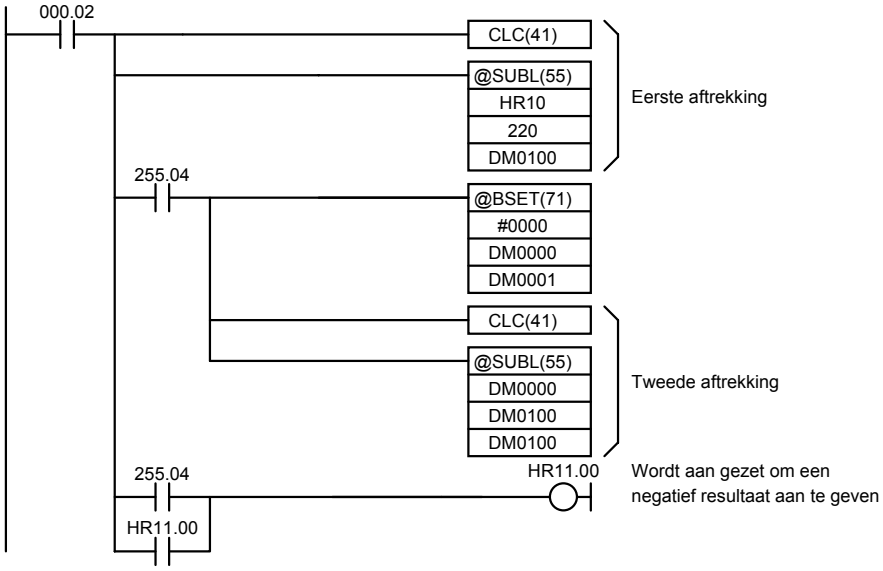
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY:** Aan wanneer het resultaat negatief is, dat is wanneer Br+1, Br kleiner is dan Aw+1, Aw plus CY.

**EQ:** Aan wanneer het resultaat 0 is.

**Voorbeeld**

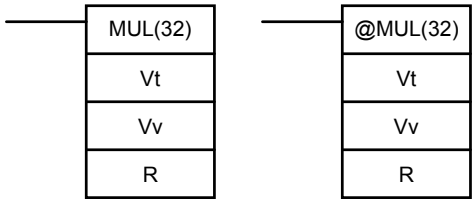
Het volgende voorbeeld werkt ongeveer zoals dat voor enkel woord aftrekkingen. In dit voorbeeld echter, moet BSET (71) gebruikt worden om de inhoud van DM0000 en DM0001 te wissen zodat het negatieve resultaat van 0 kan worden afgetrokken (het invoeren van achtcijferige constanten is niet mogelijk).




| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.02 | | |
| 00001 | OUT | TR0 | | |
| 00002 | CLC(41) | | | |
| 00003 | @SUBL(55) | HR10 | 220 | DM0100 |
| 00004 | AND | 255.04 | | |
| 00005 | @BSET(71) | #0000 | DM0000 | DM0001 |
| 00006 | CLC(41) | | | |
| 00007 | @SUBL(55) | DM0000 | DM0100 | DM0100 |
| 00008 | LD | TR0 | | |
| 00009 | AND | 255.04 | | |
| 00010 | LD | TR0 | | |
| 00011 | AND | HR11.00 | | |
| 00012 | OR LD | | | |
| 00013 | OUT | HR11.00 | | |

## *4.17.9 BCD vermenigvuldigen - MUL(32)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Vt: Vermenigvuldigtal (BCD) |
|---|
| IR, AR, DM, HR, TC, LR, # |

| Vv: Vermenigvuldiger (BCD) |
|---|
| IR, AR, DM, HR, TC, LR, # |

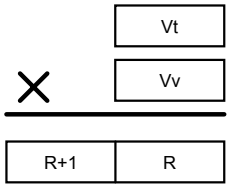
| R: Eerste resultaatwoord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

DM6143 t/m DM6655 kan niet gebruikt worden voor R. Vt en Vv moeten in BCD worden opgegeven.

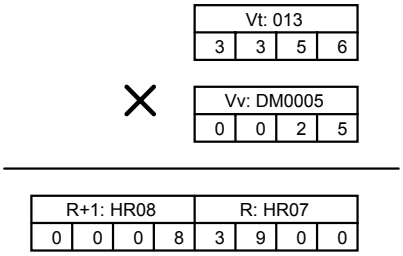
**Omschrijving**

Wanneer de executieconditie uit is, wordt MUL(32) niet uitgevoerd. Wanneer de executieconditie aan is, vermenigvuldigt MUL(32) Vt met de inhoud van Vv en plaatst het resultaat op R+1 en R.




**Voorbeeld** Wanneer 000.00 aan is, zal het volgende programma de inhoud van 013 en DM0005 met elkaar vermenigvuldigen en het resultaat op HR07 en HR08 plaatsen. Voorbeeld data en de berekeningen worden verderop getoond.


| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | MUL(32) | 013 | DM0005 | HR07 |



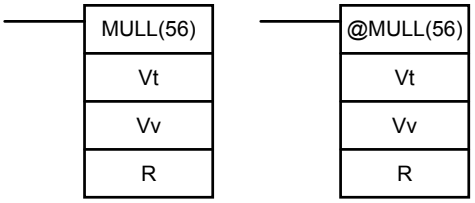

**Vlaggen**

**ER:** Vt en/of Vv is niet BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat 0 is.

### *4.17.10 Dubbel BCD vermenigvuldigen - MULL(56)*

LADDER SYMBOOL

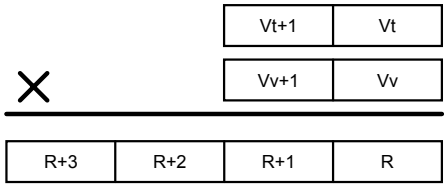



OPERAND DATAGEBIEDEN

| |
|---|
| Vt: Eerste vermenigvuldigtal (BCD) |
| IR, AR, DM, HR, TC, LR |
| Vv: Eerste vermenigvuldiger (BCD) |
| IR, AR, DM, HR, TC, LR |
| R: Eerste resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6141 t/m DM6655 kan niet gebruikt worden voor R. Br1 en Br2 moeten in BCD worden opgegeven.

**Omschrijving** Wanneer de executieconditie uit is, wordt MULL(56) niet uitgevoerd. Wanneer de executieconditie aan is, vermenigvuldigt MULL(56) de achtcijferige inhoud van Vt+1, Vt met de inhoud van Vv+1, Vv en plaatst het resultaat op R t/m R+3.



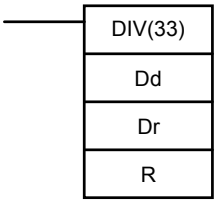

**Vlaggen**

**ER:** Vt+1, Vt en/of Vv+1, Vv is niet BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat 0 is.

## *4.17.11 BCD Delen - DIV(33)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Dd: Deeltal woord (BCD) |
|---|
| IR, AR, DM, HR, TC, LR, # |

| Dr: Deler woord (BCD) |
|---|
| IR, AR, DM, HR, TC, LR, # |

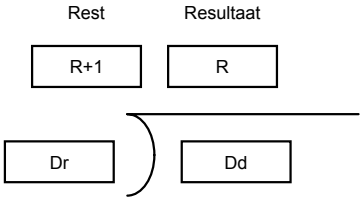
| R: Eerste resultaatwoord (BCD) |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

R en R+1 moeten in hetzelfde datagebied liggen. DM6143 t/m DM6655 kan niet gebruikt worden voor R. Dd en Dr moeten in BCD worden opgegeven.

**Omschrijving**

Wanneer de executieconditie uit is, wordt DIV(33) niet uitgevoerd en vervolgt het programma met de volgende instructie. Wanneer de executieconditie aan is, wordt Dd gedeeld door Dr en het resultaat geplaatst op R en R + 1. De deling die wordt uitgevoerd geeft een resultaat en een rest. Het resultaat wordt op R gezet en de rest op R + 1.




**Vlaggen**

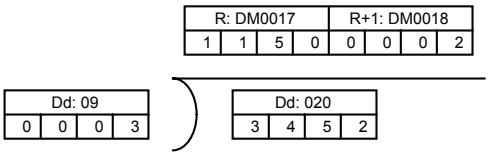
**ER:** Dd en/of Dr is niet BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat 0 is.

**Voorbeeld**

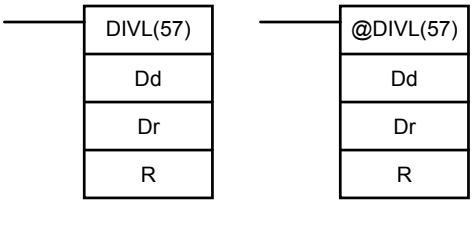
Wanneer 000.00 aan is bij het volgende programma, dan wordt de inhoud van 020 gedeeld door de inhoud van HR09 en het resultaat wordt geplaatst op DM0017 en DM0018. Voorbeeld data en de berekening worden onder het programma getoond.


| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | DIV(33) | 020 | HR09 | DM0017 |




## *4.17.12 Dubbel BCD delen - DIVL(57)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| Dd: Eerste deeltal woord (BCD) |
|---|
| IR, AR, DM, HR, TC, LR |

| Dr: Eerste deler woord (BCD) |
|---|
| IR, AR, DM, HR, TC, LR |

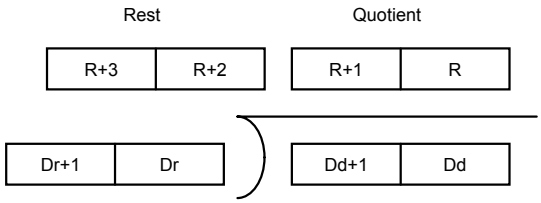
| R: Eerste resultaatwoord |
|---|
| IR, AR, DM, HR, LR |

**Beperkingen**

DM6141 t/m DM6655 kan niet gebruikt worden voor R. Dd en Dr moeten in BCD worden opgegeven.

**Omschrijving**

Wanneer de executieconditie uit is, wordt DIVL(57) niet uitgevoerd. Wanneer de executieconditie aan is, deelt DIVL(57) de achtcijferige inhoud van Dd en D+1 door de inhoud van Dr en Dr+1 en wordt het resultaat geplaatst op R t/m R+3: Het quotiënt (resultaat) op R en R+1, de rest op R+2 en R+3.




**Vlaggen**

**ER:** Dr en Dr+1 bevatten 0.
Dd, Dd+1, Dr, of Dr+1 is niet opgegeven in BCD.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

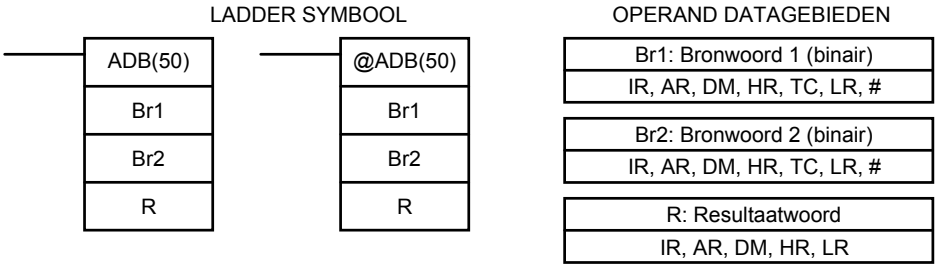
**EQ:** Aan wanneer het resultaat gelijk is aan 0.

## 4.18 Binaire berekeningen

De binaire calculatie instructies - ADB(50), SBB(51), MLB(52) en DVB(53) - voeren allemaal rekenkundige bewerkingen uit op hexadecimale data.

**Opmerking** De optelling en aftrekking gebruiken beide de Carry (CY) in de berekening en in het resultaat. Wees er zeker van dat CY gewist is als de voorgaande status niet van belang is in de berekening en gebruik het resultaat dat geplaatst is op CY, indien nodig, voor het wordt veranderd door de uitvoer van een andere instructie. STC(40) en CLC(41) kunnen gebruikt worden CY aan te sturen. Raadpleeg het hoofdstuk “bcd calculaties” op pagina 127 voor details over het gebruik van CLC(41) en STC(40).

### *4.18.1 Binair optellen - ADB(50)*




**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, wordt ADB(50) niet uitgevoerd. Wanneer de executieconditie aan is, telt ADB(50) de inhoud van Br1, Br2 en CY bij elkaar op en plaatst het resultaat op R. CY wordt geset wanneer het resultaat groter is dan FFFF.

Br1+Br2+CY→CY, R

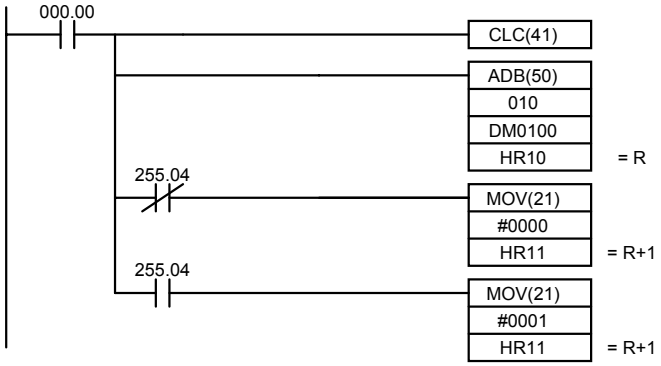
**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY:** Aan wanneer het resultaat groter is dan FFFF.

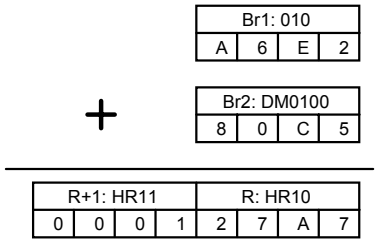
**EQ:** Aan wanneer het resultaat gelijk is aan 0.

**Voorbeelden** Het volgende voorbeeld toont een viercijferige optelling waarbij CY wordt gebruikt om #0000 of #0001 op R+1 te plaatsen om ervoor te zorgen dat de Carry bewaard blijft.

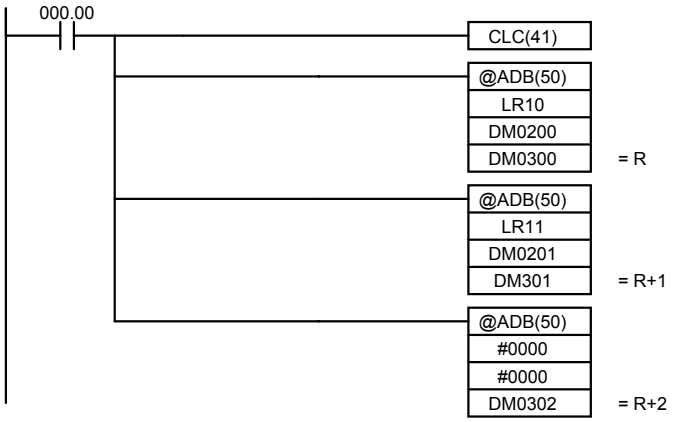


| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | OUT | TR0 | | |
| 00002 | CLC(41) | | | |
| 00003 | ADB(50) | 010 | DM0100 | HR10 |
| 00004 | AND NOT | 255.04 | | |
| 00005 | MOV(21) | #0000 | HR11 | |
| 00006 | LD | TR0 | | |
| 00007 | AND | 255.04 | | |
| 00008 | MOV(21) | #0001 | HR11 | |

In het onderstaande geval geeft de optelling A6E2 + 80C5 als resultaat 127A7. Het resultaat is en vijfcijferig getal, dus CY (255.04) = 1 en de inhoud van R+1 wordt #0001.

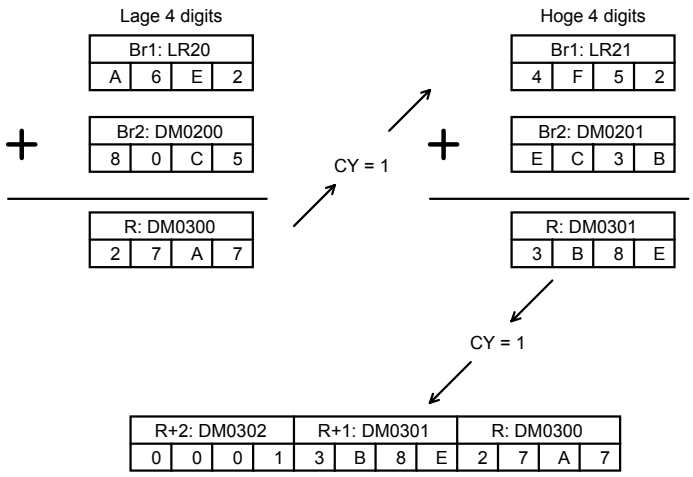



Het volgende voorbeeld voert een achtcijferige optelling uit door ADB(50) tweemaal te gebruiken. ADB(50) wordt ook gebruikt om de status van de Carry op DM0302 (een woord groter dan de rest van het antwoord) te plaatsen. Het complete antwoord staat dus op DM0300 t/m DM0302.



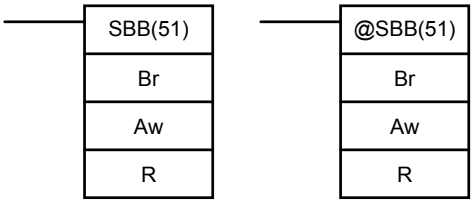

| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | CLC(41) | | | |
| 00002 | @ADB(50) | LR10 | DM0200 | DM0300 |
| 00003 | @ADB(50) | LR11 | DM0201 | DM0301 |
| 00004 | @ADB(50) | #0000 | #0000 | DM0302 |

In het volgende voorbeeld geeft de optelling 4F52A6E2 + EC3B80C5 als resultaat 13B8E27A7. De optellingen van de lage vier digits geeft als resultaat een vijfcijferig getal, dus CY (255.04) = 1 en de optelling van de hoge vier digits wordt verhoogd met 1.



### *4.18.2 Binair aftrekken - SBB(51)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| |
|---|
| Br: Bronwoord (Binair) |
| IR, AR, DM, HR, TC, LR, # |
| Aw: Af te trekken woord (Binair) |
| IR, AR, DM, HR, TC, LR, # |
| R: Resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, wordt SBB(51) niet uitgevoerd. Wanneer de executieconditie aan is, trekt SBB(51) de inhoud van Aw en CY van Br af en plaatst het resultaat op R. Als het resultaat negatief is, wordt CY geset en wordt het 2's complement van het actuele resultaat geplaatst op R.

Br-Su-CY→CY, R

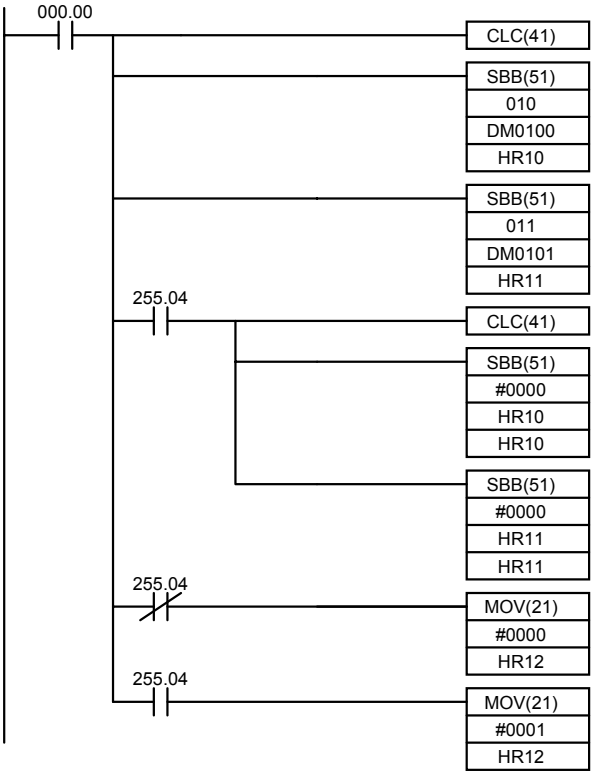
**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**CY:** Aan wanneer het resultaat negatief is, d.w.z. wanneer Br kleiner is dan Aw plus CY.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

**Voorbeeld** Het volgende voorbeeld toont een achtcijferige aftrekking. CY wordt na de eerste twee aftrekkingen getest om te controleren of het resultaat negatief is. Als het resultaat negatief is, wordt het eerste resultaat van nul afgetrokken om het om te zetten naar een positieve waarde die wordt geplaatst op HR10 en HR11. #0000 of #0001 wordt geplaatst op HR12, #0001 geeft aan dat het een negatief antwoord is.

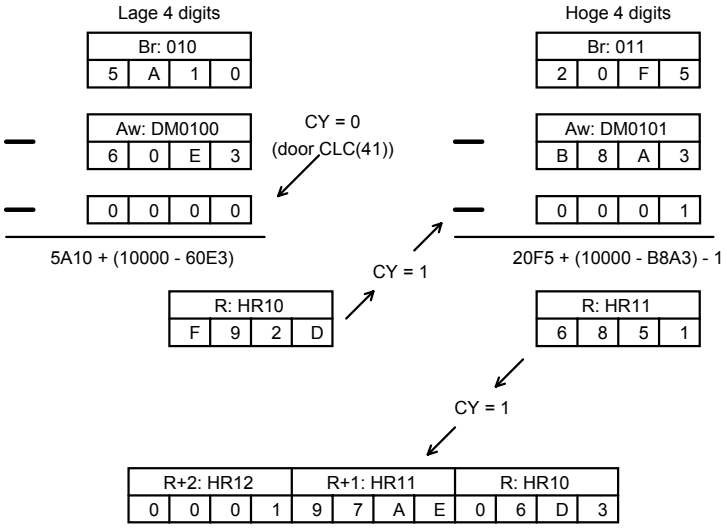



| Adres | Instructie | Operands | | |
|---|---|---|---|---|
| 00000 | LD | 000.00 | | |
| 00001 | OUT | TR0 | | |
| 00002 | CLC(41) | | | |
| 00003 | SBB(51) | 010 | DM0100 | HR10 |
| 00004 | SBB(51) | 011 | DM0101 | HR11 |
| 00005 | AND | 255.04 | | |
| 00006 | CLC(41) | | | |
| 00007 | SBB(51) | #0000 | HR10 | HR10 |
| 00008 | SBB(51) | #0000 | HR11 | HR11 |
| 00009 | LD | TR0 | | |
| 00010 | AND NOT | 255.04 | | |

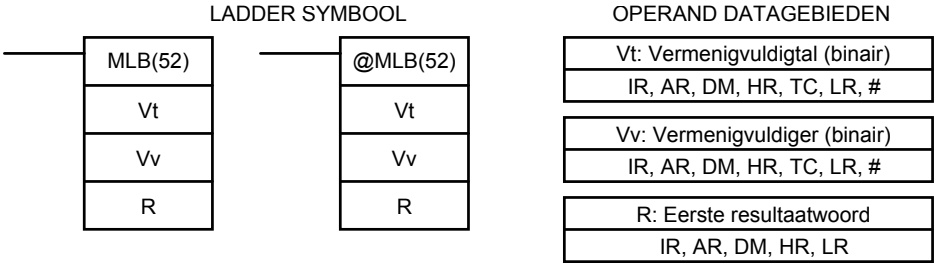
```
00011   MOV(21)     #0000       HR12
00012   LD          TR0
00013   AND         255.04
00014   MOV(21)     #0001       HR12
```

In het voorbeeld hieronder geeft de berekening 20F55A10 - B8A360E3 als resultaat 97AE06D3. Bij de aftrekking van de lage vier digits is Aw > Br, dus CY(255.04) wordt 1 en het resultaat van de aftrekking van de hoge vier digits wordt verlaagd met 1. De laatste berekening doet #0000 - F9D2 = 0000 + (10000 - F9D2) = 06D3.

#0000 - 6851 -1 (CY = 1) = 0000 + (10000 - 6851 - 1) = 97AE. De inhoud van HR12 is #0001 wat een negatief resultaat aangeeft.

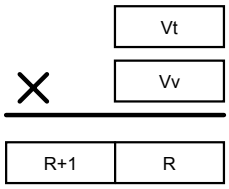



### *4.18.3 Binair vermenigvuldigen - MLB(52)*




**Beperkingen** DM6143 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, wordt MLB(52) niet uitgevoerd. Wanneer de executieconditie aan is, vermenigvuldigt MLB(52) de inhoud van Vt met de inhoud van Vv en plaatst de lage vier digits van het resultaat op R en de hoge vier digits op R+1.




**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

### *4.18.4 Binair delen - DVB(53)*

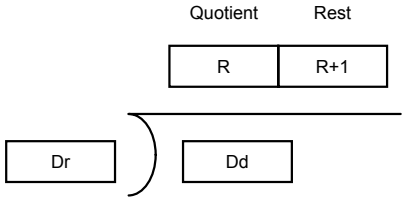
LADDER SYMBOOL


OPERAND DATAGEBIEDEN

| |
|---|
| Dd: Deeltal woord (binair) |
| IR, AR, DM, HR, TC, LR, # |
| Dr: Deler woord (binair) |
| IR, AR, DM, HR, TC, LR, # |
| R: Eerste resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6143 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, wordt DVB(53) niet uitgevoerd. Wanneer de executieconditie aan is, deelt DVB(53) de inhoud van Dd door de inhoud van Dr en het resultaat wordt geplaatst op R en R+1: Het quotiënt in R, de rest R+1.




**Vlaggen**

**ER:** Dr bevat 0.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

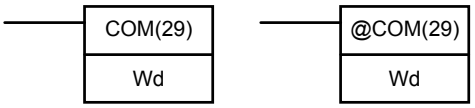
**EQ:** Aan wanneer het resultaat gelijk is aan 0.

## 4.19 Logische instructies

De logische instructies - COM(29), ANDW(34), ORW(35), XORW(36) en XNRW(37) - voeren logische bewerkingen uit op woord data.

### *4.19.1 Complement - COM(29)*

LADDER SYMBOOL



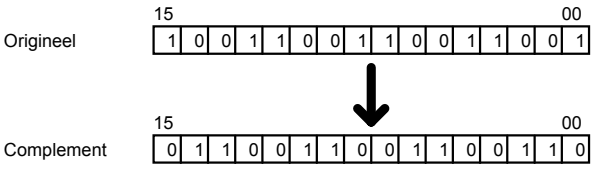

OPERAND DATAGEBIEDEN

| |
|---|
| Wd: Complement woord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor Wd.

**Omschrijving** Wanneer de executieconditie uit is, wordt COM(29) niet uitgevoerd. Wanneer de executieconditie aan is, wist COM(29) alle aan bits en set alle uit bits in Wd.

**Voorbeeld** Het onderstaande figuur toont hoe het complement over een woord bepaald wordt.




**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

### *4.19.2 Logische AND - ANDW(34)*

LADDER SYMBOOL

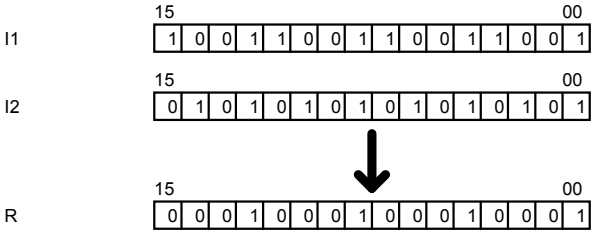

OPERAND DATAGEBIEDEN

| |
|---|
| I1: Input 1 |
| IR, AR, DM, HR, TC, LR, # |
| I2: Input 2 |
| IR, AR, DM, HR, TC, LR, # |
| R: Resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is wordt ANDW(34) niet uitgevoerd. Wanneer de executieconditie aan is voert ANDW(34) een logische AND uit tussen de inhoud van I1 en I2 (bit bij bit) en plaatst het resultaat op R.

**Voorbeeld** Het onderstaande figuur toont hoe een logische AND bewerking tussen twee woorden wordt uitgevoerd.



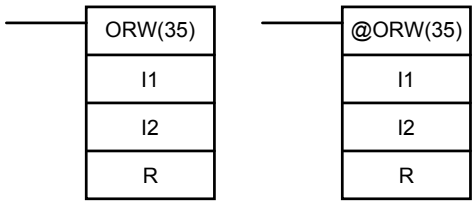

**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

### *4.19.3 Logische OR - ORW(35)*

LADDER SYMBOOL



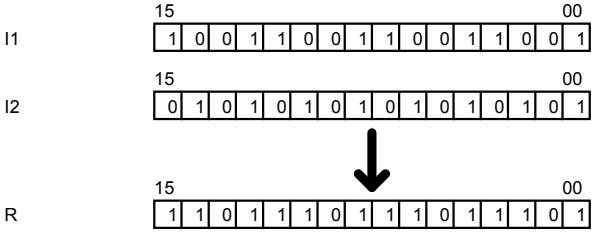

OPERAND DATAGEBIEDEN

| |
|---|
| I1: Input 1 |
| IR, AR, DM, HR, TC, LR, # |
| I2: Input 2 |
| IR, AR, DM, HR, TC, LR, # |
| R: Resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, ORW(35) niet uitgevoerd. Wanneer de executieconditie aan is, voert ORW(35) een logische OR uit tussen de inhoud van I1 en I2 bit-bij-bit en plaatst het resultaat op R.

**Voorbeeld** Het onderstaande figuur toont hoe een logische OR bewerking tussen twee woorden wordt uitgevoerd.



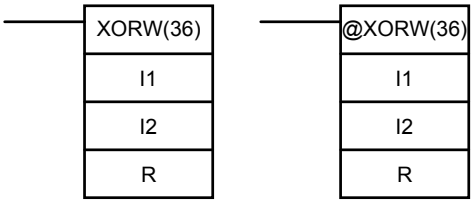

**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

### *4.19.4 Exclusive OR - XORW(36)*

LADDER SYMBOOL



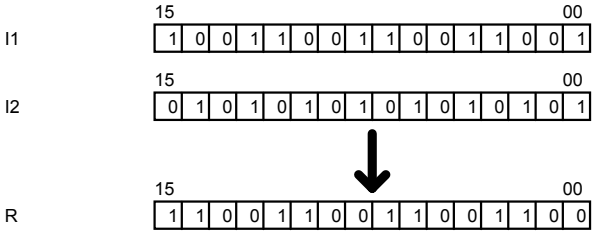

OPERAND DATAGEBIEDEN

| |
|---|
| I1: Input 1 |
| IR, SR, AR, DM, HR, TC, LR, # |
| I2: Input 2 |
| IR, SR, AR, DM, HR, TC, LR, # |
| R: Resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, wordt XORW(36) niet uitgevoerd. Wanneer de executieconditie aan is, voert XORW(36) de exclusive OR uit tussen de inhoud van I1 en I2 bit-bij-bit en plaatst het resultaat op R.

**Voorbeeld** Het onderstaande figuur toont hoe een logische XOR bewerking tussen twee woorden wordt uitgevoerd.



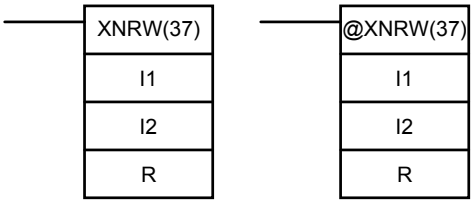

**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

### *4.19.5 Exclusive NOR - XNRW(37)*

LADDER SYMBOOL



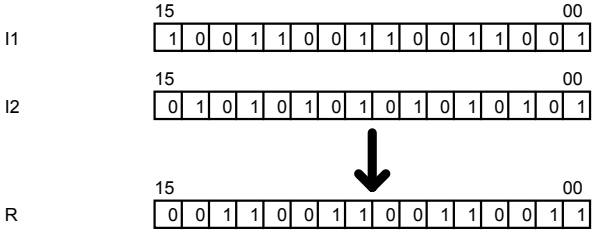

OPERAND DATAGEBIEDEN

| |
|---|
| I1: Input 1 |
| IR, AR, DM, HR, TC, LR, # |
| I2: Input 2 |
| IR, AR, DM, HR, TC, LR, # |
| R: Resultaatwoord |
| IR, AR, DM, HR, LR |

**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor R.

**Omschrijving** Wanneer de executieconditie uit is, wordt XNRW(37) niet uitgevoerd. Wanneer de executieconditie aan is, voert XNRW(37) een exclusive NOR uit tussen de inhoud van I1 en I2 bit bij bit en plaatst het resultaat op R.

**Voorbeeld** Het onderstaande figuur toont hoe een logische XOR bewerking tussen twee woorden wordt uitgevoerd.




**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

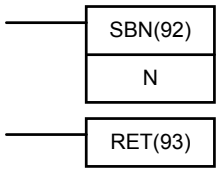
# 4.20 Subroutine en interrupt aansturing

## *4.20.1 Overzicht*

Subroutines breken lange programmataken in kleinere en maken het mogelijk om een geprogrammeerde reeks instructies opnieuw te gebruiken. Wanneer het hoofdprogramma een subroutine aanroept, dan wordt de programma-uitvoer voortgezet in die subroutine en worden dus de instructies in de subroutine uitgevoerd. De instructies in een subroutine worden op dezelfde manier geschreven als de gewone programma code. Wanneer alle instructies in de subroutine zijn uitgevoerd dan gaat het programma verder met het hoofdprogramma met de instructie die direct achter de instructie staat die de subroutine aanriep (tenzij anders gespecificeerd in de subroutine).

## *4.20.2 Subroutine definitie en return - SBN(92)/RET(93)*

LADDER SYMBOOL




DEFINER DATAGEBIEDEN

| N: Subroutinenummer |
|---|
| # (000 t/m 49) |

**Beperkingen**

Elke subroutinenummer kan maar bij één SBN(92) instructie gebruikt worden, d.w.z. tot aan 50 subroutines kunnen worden geprogrammeerd. Subroutinenummers 00 t/m 03 worden gebruikt door Interrupt Inputs.

**Omschrijving**

SBN(92) wordt gebruikt om het begin van een subroutine programma te markeren; RET(93) wordt gebruikt om het einde aan te geven. Elke subroutine wordt geïdentificeerd door een subroutinenummer, N, dat wordt geprogrammeerd als definer voor SBN(92). Dit zelfde subroutinenummer wordt gebruikt door de SBS(91) instructie die de subroutine aanroept. Voor de instructie RET(93) hoeft geen subroutinenummer ingevoerd te worden.

Alle subroutines moeten worden geprogrammeerd aan het einde van het hoofdprogramma. Wanneer één of meer subroutines zijn geprogrammeerd, dan zal het hoofdprogramma worden uitgevoerd tot aan de eerste SBN(92) instructie voordat terug wordt gegaan naar adres 00000 voor de volgende scan. Subroutines worden niet uitgevoerd tenzij deze worden aangeroepen door SBS(91) (of een andere instructie) of geactiveerd door een interrupt.

END (01) moet worden geprogrammeerd achter het laatste subroutine programma, d.w.z., achter de laatste RET(93) instructie. Het is niet nodig om op andere punten in het programma een END(01) te programmeren.

**Voorzorgen**

Als SBN(92) per ongeluk wordt geprogrammeerd in het hoofdprogramma, dan zullen instructies die erachter geprogrammeerd zijn niet uitgevoerd worden. De programma-uitvoer zal verder gaan bij de eerste instructie in het programma geheugen zodra SBN(92) is gevonden.

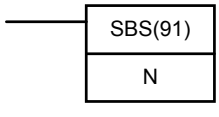
Wanneer DIFU(13) of DIFU(14) in een subroutine wordt geprogrammeerd, dan zal het operandbit niet uit gaan tot de subroutine voor de tweede keer wordt uitgevoerd. D.w.z., dat het operandbit langer dan één scan aan kan zijn of korter dan één scan wanneer de subroutine vaker binnen een scan wordt aangeroepen.

**Vlaggen**

Er worden geen vlaggen direct beïnvloed door deze instructies.

## *4.20.3 Subroutine aanroep - SBS(91)*

LADDER SYMBOOL



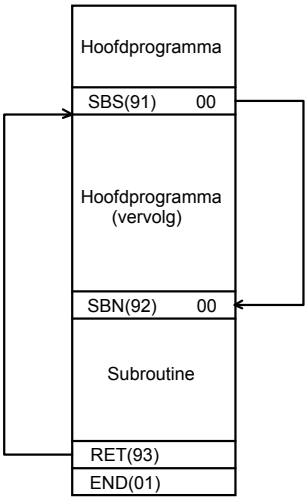

DEFINER DATAGEBIEDEN

| N: Subroutinenummer |
|---|
| # (000 to 049) |

**Omschrijving**

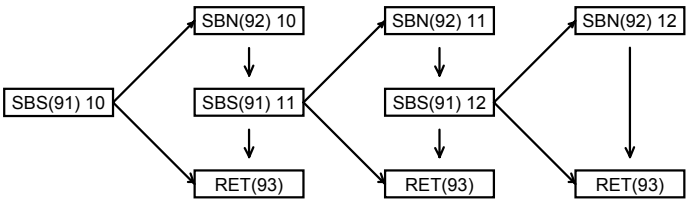
Een subroutine kan worden uitgevoerd door een SBS(91) instructie in het hoofdprogramma te plaatsen op het punt waar uitvoer van deze subroutine gewenst is. Het subroutinenummer dat gebruikt is door SBS(91) geeft het nummer aan van de gewenste subroutine. Wanneer SBS(91) wordt uitgevoerd (d.w.z.

wanneer de executieconditie ervoor aan is), worden de instructies tussen de SBN(92) instructie met hetzelfde nummer en de eerste RET(93) erachter uitgevoerd voordat het programma verder gaat met de instructie achter de SBS(91) die de aanroep deed.



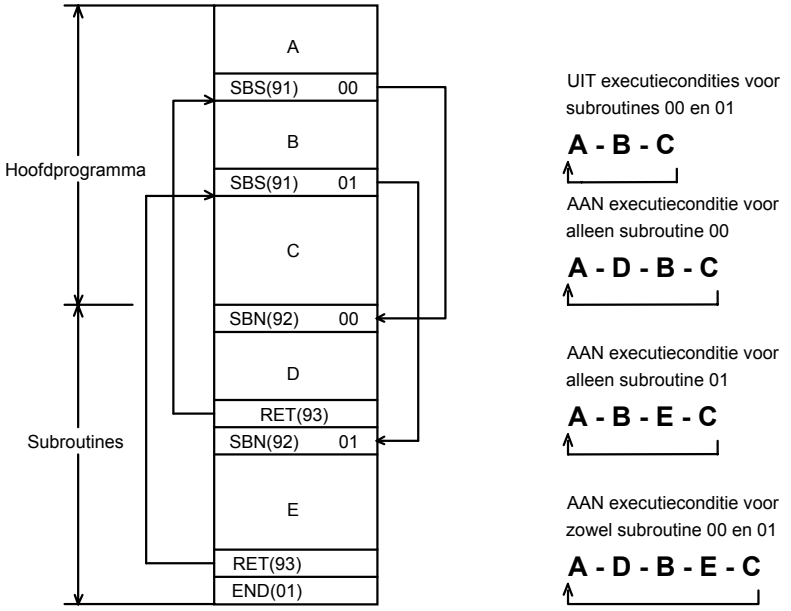

SBS(91) mag zo vaak als gewenst gebruikt worden in het programma, d.w.z. dezelfde subroutine mag op verschillende plaatsen in het programma worden aangeroepen.

SBS(91) kan ook in een subroutine worden gebruikt om vanuit de ene subroutine de andere aan te roepen, d.w.z. het nesten van subroutines is toegestaan. Wanneer de tweede subroutine is afgerond (d.w.z., RET(93) is bereikt), dan zal de programma-uitvoer worden vervolgd in de originele subroutine, die vervolgens verder wordt uitgevoerd, waarna weer terug gegaan wordt naar het hoofdprogramma. Nesten is mogelijk tot op zestien niveaus. Een subroutine mag zichzelf niet aanroepen. D.w.z. SBS(91) 00 mag niet geprogrammeerd worden in een subroutine die gedefinieerd wordt met SBN(92) 00. Het volgende figuur illustreert twee niveaus van nesten.




Alhoewel subroutine 00 t/m 31 kunnen worden aangeroepen met SBS(91), kunnen ze ook worden geactiveerd door interrupt signalen van de Interrupt Input Units. Subroutine 99, die ook kan worden aangeroepen door SBS(91) te gebruiken wordt gebruikt door de scheduled interrupt.

Het volgende figuur toont de programma-uitvoer voor verschillende executiecondities voor twee SBS(91) instructies.



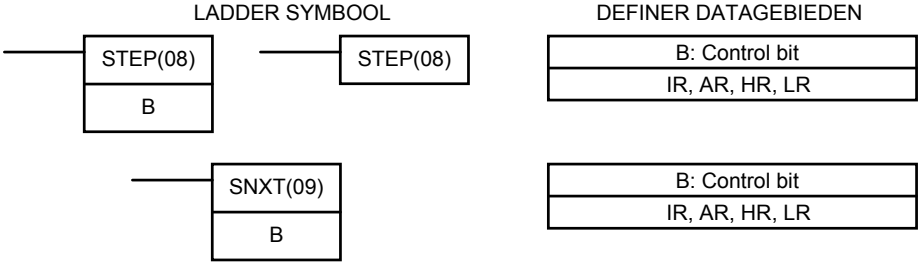

**Vlaggen** **ER:** Een subroutine bestaat niet voor het gespecificeerde subroutine nummer.
Een subroutine roept zichzelf aan.
Subroutines worden genest voor meer dan zestien niveaus.

**Opmerking** SBS(91) wordt niet uitgevoerd en de subroutine wordt niet aangeroepen wanneer ER aan is.

## 4.21 Step instructies

De stap instructies STEP(08) en SNXT(09) worden in combinatie gebruikt om een programma op te kunnen delen in kleiner, overzichtelijk stukken programma. Deze secties programma kunnen vervolgens onafhankelijk van elkaar uitgevoerd worden en gereset na uitvoer. Een programmasectie komt meestal overeen met een actueel proces in een applicatie of een deel van dat proces. Een stap is normale programmacode, behalve dat bepaalde instructies (d.w.z., IL(02) / ILC(03), JMP(04) / JME(05)) binnen een stappenprogramma niet gebruikt mogen worden.

### *4.21.1 Stap definitie en stap starten - STEP(08) / SNXT(09)*




**Beperkingen** Om een programma goed leesbaar te houden is het aan te raden om alle control bits uit hetzelfde woord te halen en de woorden en bits opeenvolgend te kiezen.

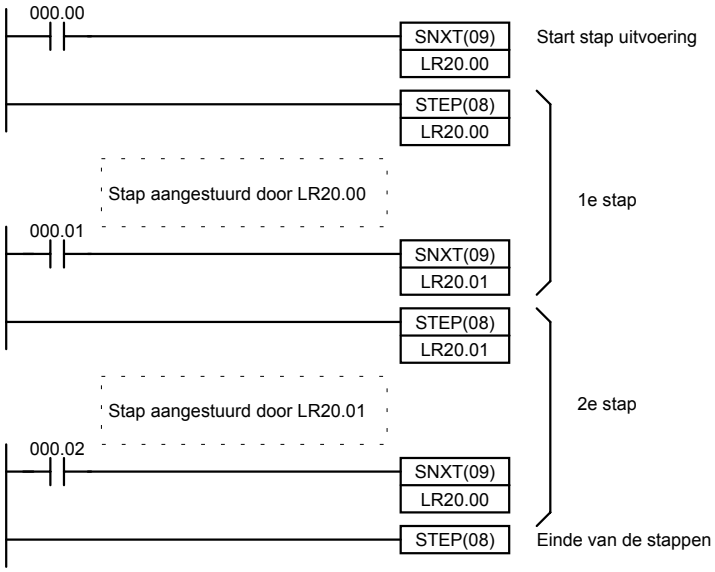
**Omschrijving** STEP(08) gebruikt een control bit in het IR of HR gebied om het begin van een programmasectie (een stap) te definiëren. De STEP(08) instructie heeft geen executieconditie, de werking van de instructie wordt bepaald door het control bit. Om een stap te activeren wordt SNXT(09) gebruikt met hetzelfde control bit als gebruikt is voor de stap (STEP(08) instructie) die geactiveerd moet worden. Wanneer SNXT(09) wordt uitgevoerd met een aan executieconditie, dan wordt de stap met hetzelfde control bit geactiveerd. Is de executieconditie uit dan wordt de stap niet geactiveerd. Wanneer de SNXT(09) instructie in het programma staat op een locatie voor de stap die de instructie activeert dan wordt deze stap dezelfde scan geactiveerd. Staat de SNXT(09) instructie verder in het programma dan de stap die deze instructie activeert dan wordt de bewuste stap pas de volgende scan uitgevoerd. SNXT(09) kan op verschillende plaatsen in het programma geprogrammeerd worden met verschillende executiecondities (zie voorbeeld 2

hieronder). Een stap in het programma die niet geactiveerd is met SNXT(09) wordt niet uitgevoerd.

Een stap is het programmagedeelte tussen twee STEP(08) instructies. Het control bit van de STEP(08) instructie die aan het begin van de stap geplaatst is, stuurt de stap aan. Een stappenprogramma wordt afgesloten met een STEP(08) instructie zonder control bit. Het programma voor en achter een stappenprogramma wordt normaal uitgevoerd.

Zodra SNXT(09) is gebruikt in het programma om een stap te starten zal deze stap geactiveerd blijven tot binnen deze stap een SNXT(09) instructie geactiveerd wordt. Om een stap uit te zetten is het ook mogelijk om een SNXT(09) instructie met een dummy control bit te activeren in deze stap. Het dummy control bit mag elk willekeurig niet gebruikte bit in het IR of HR geheugen zijn. Het mag geen bit zijn dat als controlebit wordt gebruikt door een STEP(08) instructie aangezien het dan deze stap zal activeren.

Uitvoer van een stap wordt geëindigd door een SNXT(09) te activeren binnen de stap of door het control bit van de stap uit te zetten (zie voorbeeld 3 hieronder). Wanneer een stap wordt gedeactiveerd worden alle bits die in deze stap met OUT worden aangestuurd gereset (uit gezet) en alle timers ingesteld op hun ingestelde waarde. Counters, schuifregisters en bits die bijvoorbeeld in KEEP(11) gebruikt worden houden hun status vast. Twee eenvoudige stappen worden hieronder getoond.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | SNXT(09) | LR20.00 |
| 00002 | STEP(08) | LR20.00 |

**Stap bestuurd door LR20.00**

| | | |
|---|---|---|
| 00100 | LD | 000.01 |
| 00101 | SNXT(09) | LR20.01 |
| 00102 | STEP(08) | LR20.01 |

**Stap bestuurd door LR20.01**

| | | |
|---|---|---|
| 00200 | LD | 000.02 |
| 00201 | SNXT(09) | LR20.02 |
| 00202 | STEP(08) | |

Stappen worden opeenvolgend geprogrammeerd. Elke stap begint met STEP(08) en eindigt over het algemeen met SNXT(09) en het begin van de volgende stap. (Zie voorbeeld 3 hieronder voor een uitzondering). In principe zijn er drie manieren waarop stappen uitgevoerd kunnen worden: sequentieel, aftakkend of parallel. De executiecondities ervoor en de plaats van SNXT(09) bepalen hoe de stappen worden uitgevoerd. De drie voorbeelden hieronder tonen deze drie typen van stapuitvoer.

**Voorzorgen**

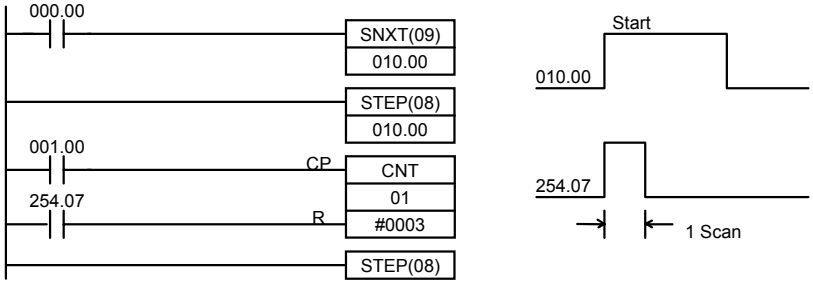
Interlocks, sprongen, SBN(92) en END (01) kunnen niet gebruikt worden in stappen programma's.

Bits die gebruikt worden als control bits moeten niet ergens anders in het programma aangestuurd worden, tenzij dit met het oogmerk is om de status van de betreffende stap te beïnvloeden. (Zie voorbeeld 3 hieronder). Het is aan te raden om alle control bits opeenvolgend uit hetzelfde woord te kiezen.

Wanneer IR of LR bits worden gebruikt als control bits, raakt hun status verloren tijdens spanningsinterrupties. Wanneer het noodzakelijk is dat stappen bij spanningsuitval hun status moeten behouden en programma-uitvoer doorgaat waar dit gebleven was moeten HR bits gebruikt worden.

**Vlaggen**

**25407:** Step Start Vlag: Gaat voor één scan aan wanneer STEP(08) wordt uitgevoerd en kan gebruikt worden om bijvoorbeeld counters te resetten zoals hieronder getoond is.




| Adres | Instructie | Operands | |
|---|---|---|---|
| 00000 | LD | 000.00 | |
| 00001 | SNXT(09) | 010.00 | |
| 00002 | STEP(08) | 010.00 | |
| 00003 | LD | 001.00 | |
| 00004 | LD | 254.07 | |
| 00005 | CNT | 01 | #0003 |
| 00006 | STEP(08) | | |

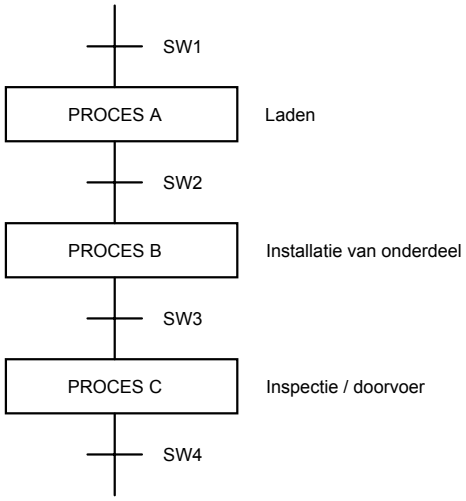
**Voorbeelden**

De volgende drie voorbeelden tonen de drie typen programma-uitvoer die mogelijk zijn met stappenprogramma's. Voorbeeld 1 demonstreert sequentiële uitvoer, voorbeeld 2 het maken van keuzen (programma takken) en voorbeeld 3 toont parallelle uitvoer.

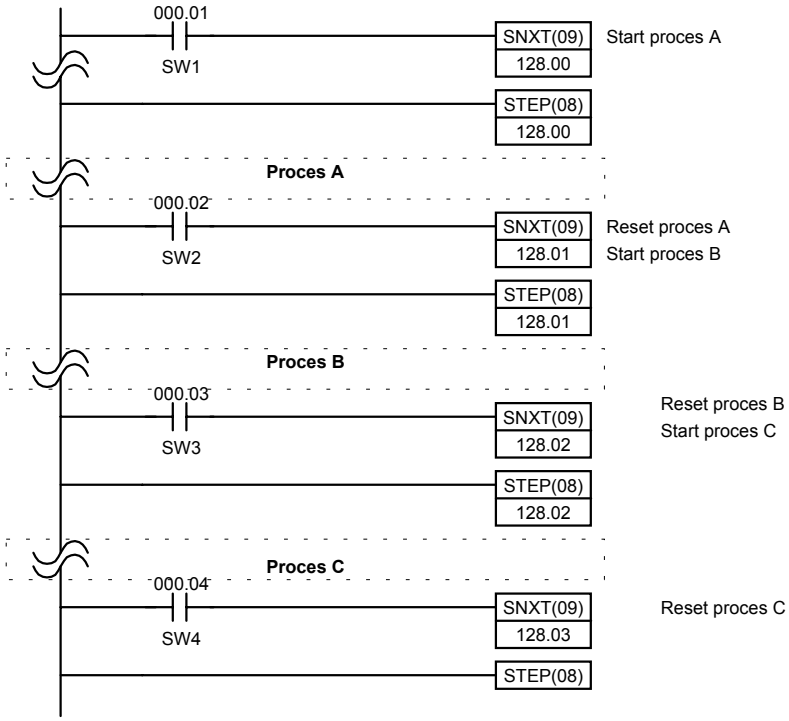
**Voorbeeld 1: Sequentiële uitvoer**

Stel een productieproces voor dat drie processen nodig heeft die sequentieel afgehandeld moeten worden, waarbij elk proces gereset moet worden voordat begonnen wordt met het volgende proces. Diverse sensoren (SW1, SW2, SW3 en SW4) worden gebruikt om te bepalen wanneer een proces is afgelopen en het volgende proces gestart kan worden.

Het volgende diagram toont schematisch de programmadoorloop en de voorwaarden die gebruikt zijn voor aansturing.




Het programma voor dit proces, dat hieronder getoond is, gebruikt het meest basistype stappenprogramma. Elke stap wordt afgesloten met een unieke SNXT(09) die de volgende stap start. Elke volgende step start wanneer de schakelaar die aangeeft dat de vorige stap klaar is aan gaat. In de laatste stap wordt het stappenprogramma beëindigd door een SNXT(09) uit te voeren met een dummy control bit.

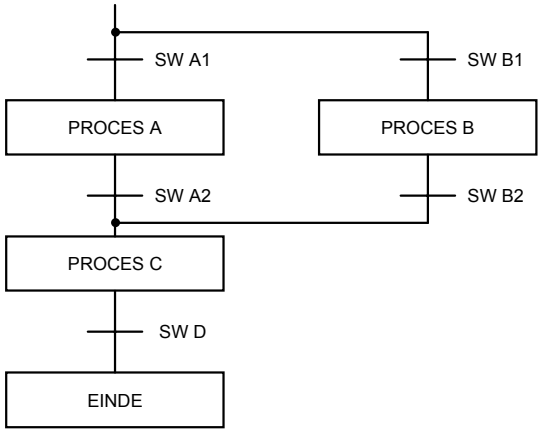



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.01 |
| 00001 | SNXT(09) | 128.00 |
| 00002 | STEP(08) | 128.00 |
| | **Proces A** | |
| 00100 | LD | 000.02 |
| 00101 | SNXT(09) | 128.01 |
| 00102 | STEP(08) | 128.01 |
| | **Proces B** | |
| 00200 | LD | 000.03 |
| 00201 | SNXT(09) | 128.02 |
| 00202 | STEP(08) | 128.02 |
| | **Proces C** | |
| 00300 | LD | 000.04 |
| 00301 | SNXT(09) | 128.03 |
| 00302 | STEP(08) | |

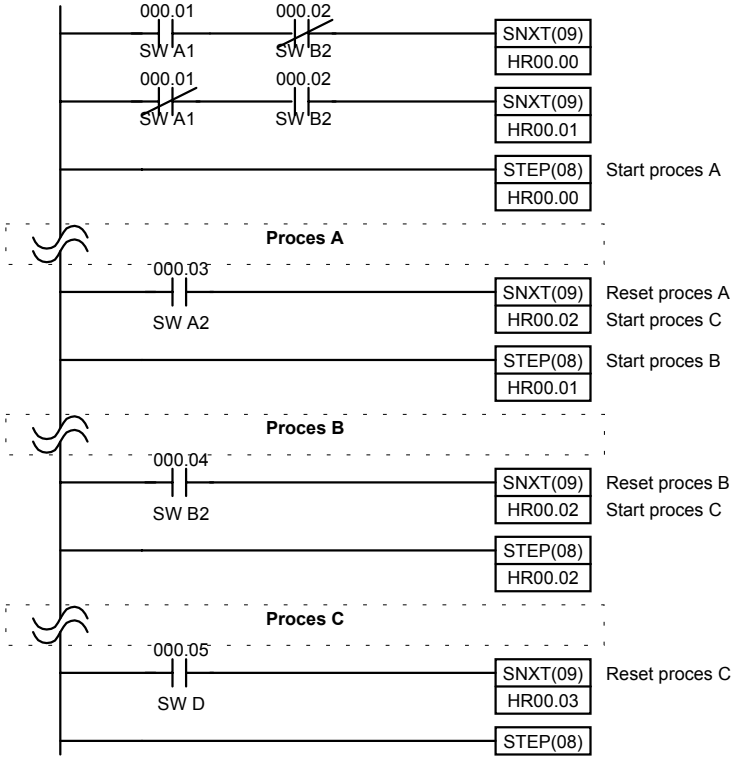
**Voorbeeld 2: Aftakkingen / keuzen**

Stel een proces voor waarin het noodzakelijk is dat een product op twee verschillende manieren behandeld kan worden, bijvoorbeeld afhankelijk van het gewicht van het product. Vervolgens moet een label op het product geprint worden, dat voor beide producten weer identiek is. het print proces is dus weer hetzelfde, onafhankelijk van welk proces ervoor gebruikt werd. Diverse sensoren worden gebruikt om te bepalen wanneer de processen moeten beginnen en stoppen.

De volgende figuur toont de programmadoorloop en de sensoren die gebruikt worden om het programma aan te sturen. In dit voorbeeld wordt de keuze voor proces A of B gemaakt met de sensoren SW A1 en SW B1.



Het programma voor dit proces, dat hieronder getoond wordt, begint met twee SNXT(09) instructies die proces A of B starten. Door de manier waarop 000.01 (SW A1) en 000.02 (SW B1) geprogrammeerd zijn kan maar één van deze SNXT(09) instructies een proces starten, dus proces A of B zal geactiveerd worden. Zowel proces A als B eindigt met een SNXT(09) instructie die proces C activeert.

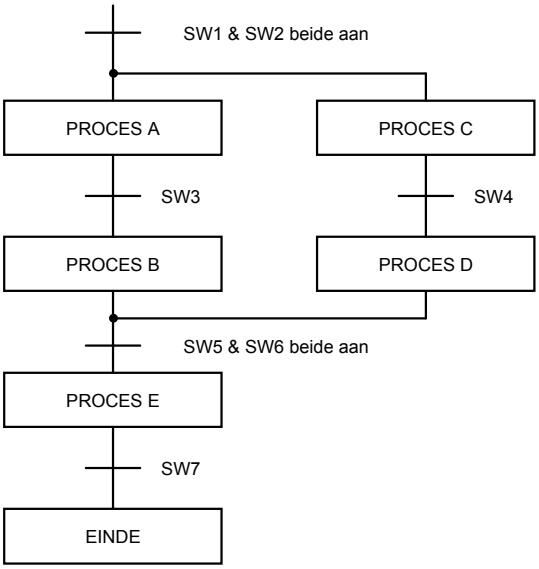



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.01 |
| 00001 | AND NOT | 000.02 |
| 00002 | SNXT(09) | HR00.00 |
| 00003 | LD NOT | 000.01 |
| 00004 | AND | 000.02 |
| 00005 | SNXT(09) | HR00.01 |
| 00006 | STEP(08) | HR00.00 |
| | **Proces A** | |
| 00100 | LD | 000.03 |
| 00101 | SNXT(09) | HR00.02 |
| 00102 | STEP(08) | HR00.01 |
| | **Proces B** | |
| 00200 | LD | 000.04 |
| 00201 | SNXT(09) | HR00.02 |
| 00202 | STEP(08) | HR00.02 |
| | **Proces C** | |
| 00300 | LD | 000.05 |
| 00301 | SNXT(09) | HR00.03 |
| 00302 | STEP(08) | |

**Voorbeeld 3: Parallelle uitvoer**

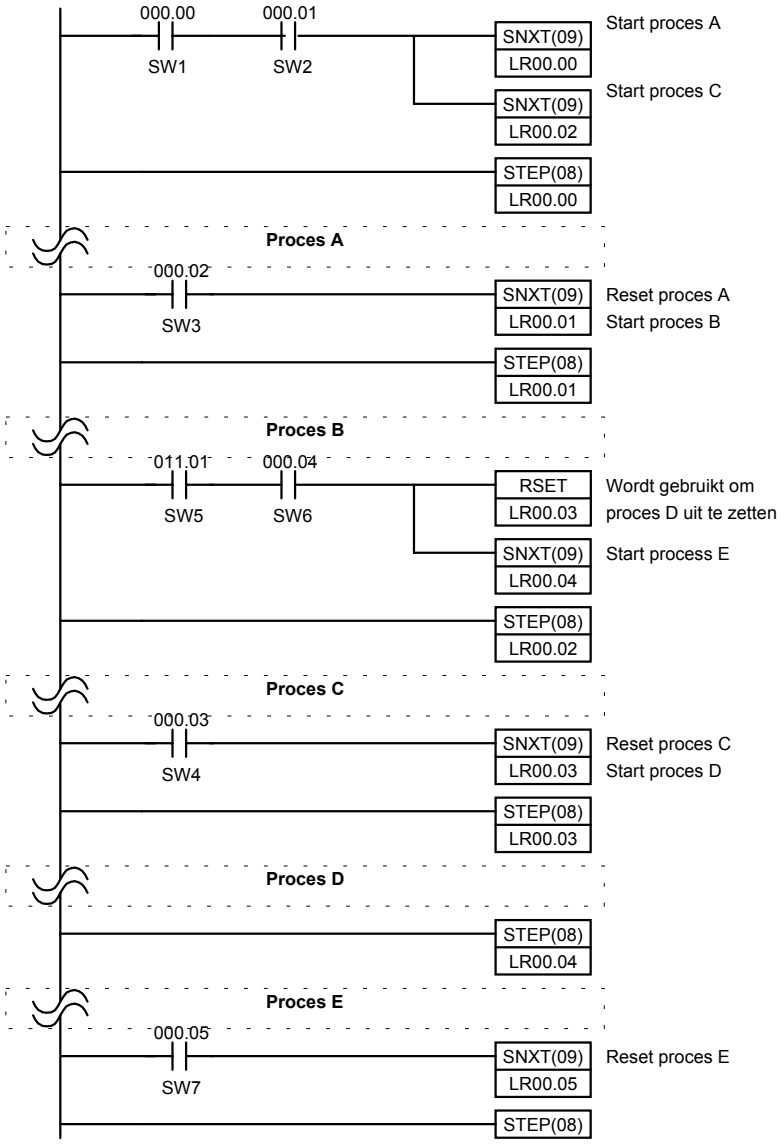
Stel een proces voor waarin het noodzakelijk is dat twee delen van een product tegelijkertijd door twee verschillende processen verwerkt worden voordat deze samen in een vijfde proces samen bewerkt worden. Diverse sensoren worden gebruikt om aan te geven wanneer een proces gestart of beëindigd moet worden.

Het volgende diagram toont de programmadoorloop en de schakelaars die gebruikt worden om het programma aan te sturen. In dit voorbeeld worden proces A en proces C tegelijkertijd gestart. Wanneer proces A eindigt, wordt proces B gestart; wanneer proces C eindigt, start proces D. Wanneer zowel proces B en D klaar zijn start proces E.




Het programma voor dit proces start met twee SNXT(09) instructies die proces A en C starten. Deze instructies takken af van dezelfde instructieregel en worden altijd samen uitgevoerd, waardoor stap A en C beide geactiveerd worden. Wanneer stap A is geëindigd start stap B onmiddellijk, wanneer stap C is geëindigd start stap D onmiddellijk.

Wanneer zowel proces B en proces D geëindigd zijn (d.w.z. wanneer SW5 en SW6 aan zijn), worden proces B en D samen gereset door de executieconditie voor de SNXT(09) aan het einde van het programma in proces B. Alhoewel er geen SNXT(09) is geprogrammeerd aan het einde van proces D, wordt het controlebit van deze stap toch uitgezet door het in stap B te resetten. Hiervoor is aan het einde van de executieconditie voor de SNXT(09) instructie in stap B een RSET instructie geprogrammeerd met het controlebit van stap D als operand. Proces B wordt dus direct gereset en proces D wordt indirect gereset voordat stap E wordt geactiveerd. Bij het indirect deactiveren van stappen is het aan te raden om zowel de stap die indirect gedeactiveerd wordt als de stap waarin deze actie gepleegd wordt, in het programma voor de stap te programmeren die vervolgens geactiveerd wordt. In dit voorbeeld houdt dat in dat stap D in de PLC voor stap E uitgevoerd moet worden. Daarna is het aan te raden om de stap die indirect gedeactiveerd wordt, na de stap te programmeren die deze stap indirect deactiveert. in dit voorbeeld, houdt dat in dat stap D na stap B uitgevoerd wordt.




| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | AND | 000.01 |
| 00002 | SNXT(09) | LR00.00 |
| 00003 | SNXT(09) | LR00.02 |
| 00004 | STEP(08) | LR00.00 |
| | **Proces A** | |
| 00100 | LD | 000.02 |
| 00101 | SNXT(09) | LR00.01 |
| 00102 | STEP(08) | LR00.01 |
| | **Proces B** | |
| 00200 | LD | 011.01 |
| 00201 | AND | 000.04 |
| 00201 | RSET | LR00.03 |
| 00201 | SNXT(09) | LR00.04 |
| 00202 | STEP(08) | LR00.02 |
| | **Proces C** | |
| 00300 | LD | 000.03 |
| 00301 | SNXT(09) | LR00.03 |
| 00302 | STEP(08) | LR00.03 |
| | Proces D | |
| 00400 | STEP(08) | LR0004 |
| | **Proces E** | |
| 00500 | LD | 000.05 |
| 00501 | SNXT(09) | LR00.05 |
| 00502 | STEP(08) | |

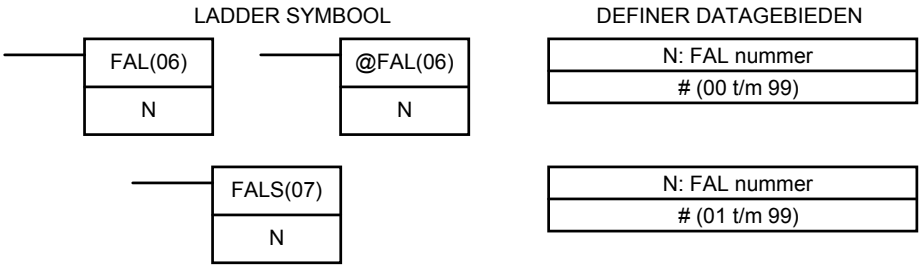
**Opmerking** Alhoewel in de bovenstaande voorbeelden steeds maar één van de mogelijke drie structuren is getoond, mogen deze natuurlijk ook gecombineerd gebruikt worden.

Daarnaast is het ook mogelijk om in een stappenprogramma terug te springen naar een voorgaande stap.

## 4.22 Speciale instructies

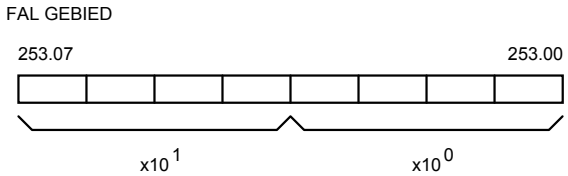
De volgende instructies worden gebruikt voor diverse bewerkingen zoals bijvoorbeeld: het genereren van user foutcodes en boodschappen, tellen van aan bit in woorden, instellen van de watchdog timer en refreshen van I/O tijdens programma-uitvoer.

### *4.22.1 Failure alarm en severe failure alarm - FAL(06) / FALS(07)*




**Omschrijving**

De FAL(06) en FALS(07) instructie kunnen door de programmeur gebruikt worden om foutcodes te creëren die tijdens bedrijf of voor onderhoud en debugging gebruikt kunnen worden. Wanneer de instructie wordt uitgevoerd met een aan executieconditie zullen beide instructies een FAL nummer naar de bits 00 t/m 07 van woord 253 schrijven. Het FAL nummer dat gebruikt kan worden ligt tussen 01 en 99 en wordt ingevoerd als definer voor de FAL(06) of FALS(07) instructie. FAL(06) met een definer 00 wordt gebruikt om dit gebied te resetten (zie hieronder).




FAL(06) produceert een niet fatale error en FALS(07) produceert een fatale error. Wanneer FAL(06) wordt uitgevoerd met een aan executieconditie, dan zal de ALARM / ERROR indicator op het front van de CPU gaan knipperen, maar de PLC zal doorgaan met de uitvoer van het programma. Wanneer FALS(07) wordt uitgevoerd met een aan executieconditie, dan zal de ALARM / ERROR indicator gaan branden en zal de PLC stoppen met programma uitvoer.

De PLC genereert zelf ook error codes op het FAL gebied.

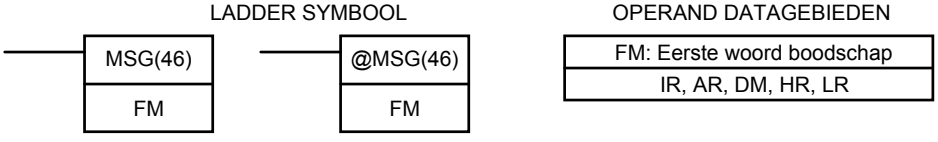
**Fouten resetten**

Een maximum van drie FAL errorcodes worden in het geheugen opgeslagen. Maar één van deze codes is beschikbaar op het FAL gebied. Om de andere codes uit te kunnen lezen moet het FAL gebied gereset worden door FAL(06) 00 uit te voeren. Elke keer als FAL(06) 00 wordt uitgevoerd, wordt de volgende FAL errorcode naar het FAL gebied verplaatst, waarbij de foutcode die er op dat moment staat gewist wordt. Het is natuurlijk niet mogelijk om foutcodes te wissen van fouten die nog actief zijn.

FAL(06) 00 wordt ook gebruikt om een boodschap te wissen die met de instructie MSG(46) geprogrammeerd is.

Wanneer het FAL gebied niet gewist kan worden, wat meestal het geval is wanneer FALS(07) wordt uitgevoerd, verwijder dan eerst de oorzaak van de fout en wis vervolgens het FAL gebied.

### *4.22.2 Toon boodschap - MSG(46)*



**Omschrijving**

Wanneer uitgevoerd met een aan executieconditie, leest MSG(46) acht woorden met extended ASCII code van FM t/m FM+7 en toont de boodschap op het handprogrammeerapparaat, GPC, FIT of in het status venster van SYSWIN. De getoonde boodschap kan tot aan 16 karakters lang zijn, elk ASCII karakter gebruikt acht bits (twee digits).

Wanneer niet alle acht woorden noodzakelijk zijn voor de boodschap, kan deze op een willekeurig punt gestopt worden door op een byte “0D” in te voeren. Wanneer OD wordt gevonden in de boodschap worden de karakters die er eventueel achter staan niet uitgelezen. Deze zijn beschikbaar voor andere doeleinden in het PLC programma.

**Boodschappen bufferen en prioriteiten**

Tot aan drie boodschappen kunnen worden gebufferd in het geheugen. Zodra boodschappen in de buffer staan worden deze getoond op first in, first out basis. Aangezien het mogelijk is dat meer dan drie MSG(46) instructies uitgevoerd kunnen worden in dezelfde scan, bestaat er een prioriteitsschema, gebaseerd op het gebied waarin de boodschappen zijn opgeslagen voor de selectie van de boodschappen die gebufferd moeten worden.

De prioriteit van de datagebieden is als volgt:
LR > IR(I/O) > IR(geen I/O) > HR > AR > TC > DM

Voor de afhandeling van boodschappen die in hetzelfde datagebied staan, hebben de boodschappen die op het laagste adres staan een hogere prioriteit.

Voor de afhandeling van indirect geadresseerde boodschappen (d.w.z. *DM) geld ook dat berichten met het laagste DM adres hogere prioriteit hebben.

**Boodschappen wissen**

Voer FAL(06) 00 uit in het programma om een boodschap te wissen.

Als de inhoud van de boodschap verandert terwijl deze wordt getoond dan zal de tekst die op dat moment getoond wordt ook veranderen.

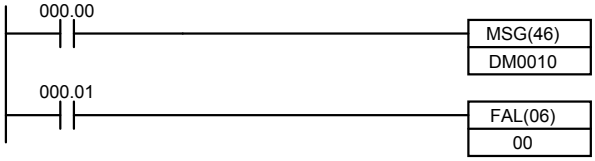
Om een string te tonen op het handprogrammeerapparaat moet deze in TERMINAL mode gezet worden. Alhoewel LMSG(47) normaal wordt uitgevoerd, verschijnt de string niet correct op het scherm van het handprogrammeerapparaat tenzij deze in TERMINAL mode staat.

**Vlaggen**

**ER:** Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**Voorbeeld**

Het volgende voorbeeld toont de tekst die op het handprogrammeerapparaat getoond wordt wanneer de executieconditie voor de MSG(46) instructie aan is, en de data die daarvoor in het geheugen van de PLC moet worden opgeslagen. Als 000.01 aan gaat wordt de boodschap gewist.

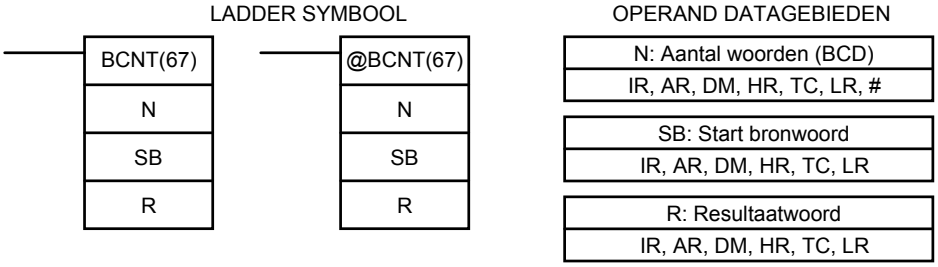



| Adres | Instructie | Operands |
|---|---|---|
| 00000 | LD | 000.00 |
| 00001 | MSG(46) | DM0010 |
| 00002 | LD | 000.01 |
| 00003 | FAL(06) | 00 |

| DM inhoud | | | ASCII equivalent | |
|---|---|---|---|---|
| DM0010 | 41 | 42 | A | B |
| DM0011 | 43 | 44 | C | D |
| DM0012 | 45 | 46 | E | F |
| DM0013 | 47 | 48 | G | H |
| DM0014 | 49 | 4A | I | J |
| DM0015 | 4B | 4C | K | L |
| DM0016 | 4D | 4E | M | N |
| DM0017 | 4F | 50 | O | P |

MSG
ABCDEFGHIJKLMNOP

### *4.22.3 Bit counter - BCNT(67)*




**Beperkingen** N mag niet nul zijn. SB en SB+(N-1) moeten in hetzelfde datagebied liggen. DM6144 t/m DM6655 kan niet gebruikt worden voor R.

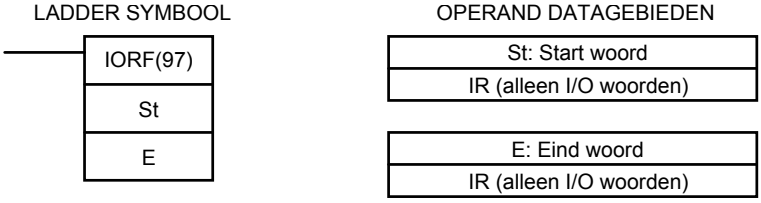
**Omschrijving** Wanneer de executieconditie uit is, wordt BCNT(67) niet uitgevoerd. Wanneer de executieconditie aan is, telt BCNT(67) het totale aantal bits dat aan is in de woorden tussen SB en SB+(N-1) en plaatst het resultaat op R.

**Vlaggen**

**ER:** N is niet opgegeven in BCD, N is 0 of SB en SB+(N-1) liggen niet in hetzelfde datagebied.
Het resultaat van de telling is groter dan 9999.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

**EQ:** Aan wanneer het resultaat gelijk is aan 0.

### *4.22.4 I/O Refresh - IORF(97)*

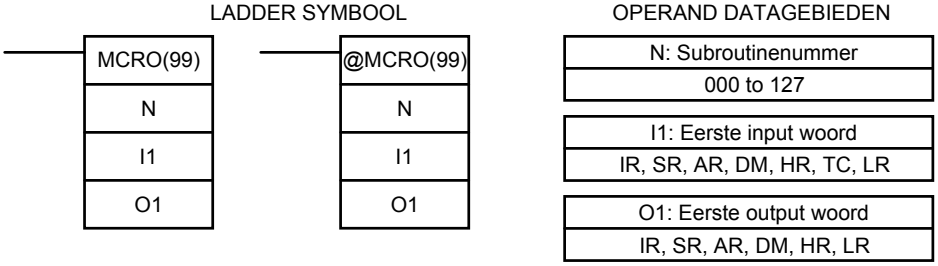



**Beperkingen** St moet kleiner of gelijk zijn aan E. IORF(97) kan gebruikt worden om I/O woorden die aan I/O Units zijn toegewezen te refreshen.

**Omschrijving** Om I/O woorden te refreshen die zijn toegewezen aan de CPU of een uitbreiding volstaat het om het eerste (St) en het laatste (E) van de te refreshen I/O woorden aan te geven. Wanneer de executieconditie voor IORF(97) aan is, worden alle I/O woorden tussen St en E gerefreshed. Dit zal gebeuren naast de normale I/O refresh tijdens de CPU's scan.

**Vlaggen** Er worden geen vlaggen beïnvloed door deze instructie.

### *4.22.5 Macro - MCRO(—)*




**Beperkingen** DM6144 t/m DM6655 kan niet gebruikt worden voor I1 en O1.

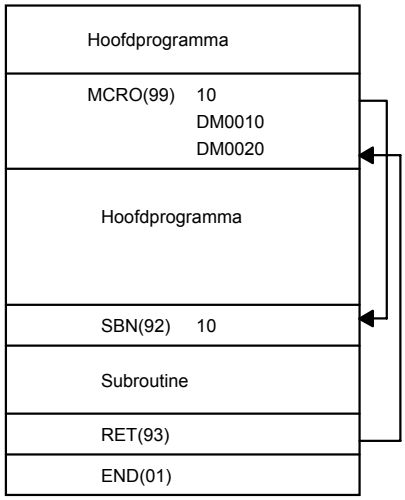
**Omschrijving** De MACRO instructie maakt het mogelijk om één enkele subroutine te gebruiken ter vervanging van een aantal subroutines die een identieke structuur hebben maar andere operands. Door deze instructie worden vier input woorden IR232 t/m IR235 en vier output woorden IR236 t/m IR239 gebruikt. Deze woorden kunnen worden gebruikt in de subroutine en nemen de inhoud over van I1 t/m I1+3 en O1 t/m O1+3 wanneer de subroutine wordt uitgevoerd.

Wanneer de executieconditie uit is, wordt MCRO(99) niet uitgevoerd. Wanneer de executieconditie aan is, kopieert MCRO(99) inhoud van I1 t/m I1+3 naar IR232 t/m IR235 en kopieert de inhoud van O1 t/m O1+3 naar IR236 t/m IR239 en roept

vervolgens de subroutine die op N gespecificeerd is aan. Wanneer de subroutine uitvoer klaar is wordt de inhoud van IR236 t/m IR239 terug gekopieerd naar O1 t/m O1+3 voordat MCRO(99) uitvoer af is.

**Voorbeeld**

In dit voorbeeld wordt de inhoud van DM0010 t/m DM0013 gekopieerd naar IR woord 232 t/m 235, de inhoud van DM0020 t/m DM0023 wordt gekopieerd naar IR woord 236 t/m 239 en subroutine 10 wordt vervolgens aangeroepen. Wanneer de subroutine uitgevoerd is, wordt de inhoud van IR woord 236 t/m 239 terug gekopieerd naar DM0020 t/m DM0023.

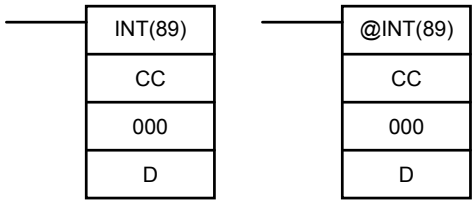



**Vlaggen**

**ER:** Een subroutine met het opgegeven nummer bestaat niet.
Een operand heeft de grenzen van een datagebied overschreden.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
Een subroutine heeft zichzelf aangeroepen.
Een actieve subroutine is aangeroepen.

### *4.22.6 Interrupt beheer - INT(89)*

LADDER SYMBOOL




OPERAND DATAGEBIEDEN

| |
|---|
| CC: Control code |
| # (000 t/m 003, 100 of 200) |
| 000: Geen functie |
| # (000 t/m 004) |
| D: Control data |
| IR, AR, DM, HR,TC, LR, TR, # |

**Beperkingen**

Wanneer CC 002 moet D een adres zijn waarop de CPM1(A) mag schrijven.

**Omschrijving**

Wanneer de executieconditie uit is, wordt INT(-) niet uitgevoerd. Wanneer de executieconditie aan is wordt INT(-) gebruikt om interrupts te beheren en één van de zes functies uit de onderstaande tabel uit te voeren. De gekozen functie is afhankelijk van de waarde van CC.

| CC | INT(-) functie |
|---|---|
| 000 | Maskeer / demaskeer input interrupts |
| 001 | Wis input interrupts |
| 002 | Lees de status van het huidige masker |
| 003 | Vernieuw het counter SP |
| 100 | Maskeer alle input interrupts |
| 200 | Demaskeer alle input interrupts |

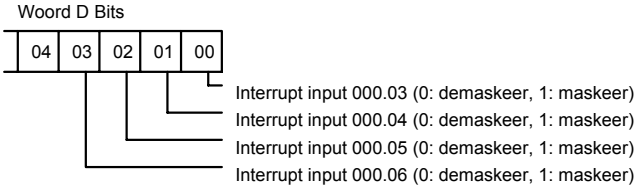
Deze zes functies worden hieronder meer gedetailleerd beschreven. Lees de sectie over interrupts “Instellen en gebruik van de CPM1(A) interrupt functies” op pagina 39 voor aanvullende informatie en voorbeelden.

**Maskeer / demaskeer input interrupts (CC = 000)**

De functie maskeer / demaskeer input interrupts wordt gebruikt om input interrupt 000.03 t/m 000.06 te activeren of de deactiveren. Gemaskeerde interrupts worden opgeslagen en genegeerd. Het interrupt programma dat bij een gemaskeerde

interrupt hoort wordt uitgevoerd zodra de interrupt gedemaskeerd is, tenzij natuurlijk de interrupt gewist is door INT(-) uit te voeren met een CC van 001.

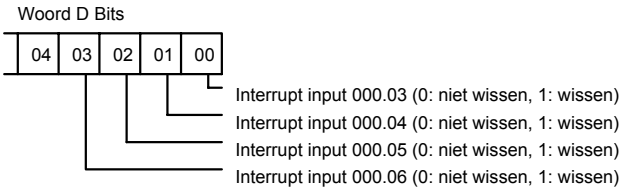
Zet het juiste bit op D op 0 of 1 om een input interrupt te demaskeren of te maskeren. De bits 0 t/m 3 corresponderen met de interrupt ingangen 000.03 t/m 000.06. De bits 04 t/m 15 moeten op 0 worden ingesteld.




**Wis input interrupts (CC = 001)**

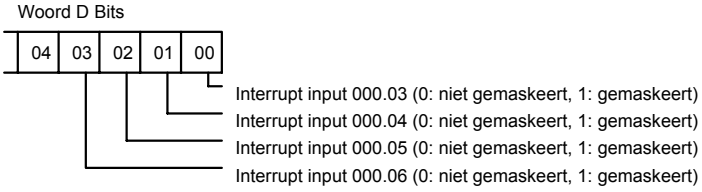
De functie wis input interrupts wordt gebruikt om een opgeslagen input interrupt voor ingang 000.03 t/m 000.06 te wissen. Aangezien gemaskeerde interrupts worden opgeslagen maar niet uitgevoerd, worden deze interrupts uitgevoerd zodra ze gedemaskeerd worden tenzij ze eerst gewist worden.

Zet het juiste bit op D op 0 of 1 om een input interrupt te wissen. De bits 0 t/m 3 corresponderen met de interrupt ingangen 000.03 t/m 000.06. De bits 04 t/m 15 moeten op 0 worden ingesteld.



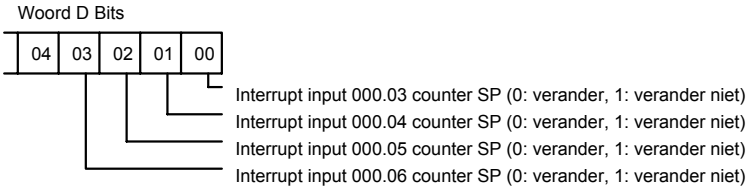

**Lees de status van het huidige masker (CC = 002)**

De functie lees de status van het huidige masker wordt gebruikt om deze status naar woord D van de CPM1(A) te schrijven. Het corresponderende bit zal aan zijn als de interrupt gemaskeerd is.




**Vernieuw het counter SP (CC = 003)**

De functie vernieuw het counter SP wordt gebruikt om een nieuw setpoint in te voeren wanneer de input interrupt in counter mode werkt. Het nieuwe setpoint is in het IR geheugen van de CPM1(A) opgeslagen. Met deze functie kan per interrupt ingang geselecteerd worden of het nieuwe setpoint gelezen moet worden of niet.




**Maskeer / demaskeer alle input interrupts (CC = 100/200)**

De functie maskeer / demaskeer alle input interrupts wordt gebruikt om in één keer alle interrupts te kunnen activeren of deactiveren. Gemaskeerde interrupts worden opgeslagen en niet uitgevoerd. Woord D wordt bij deze instelling niet gebruikt, stel het in op #0000.

**Vlaggen**

**ER:** Het SP van een counter is niet correct (alleen als CC = 003).

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

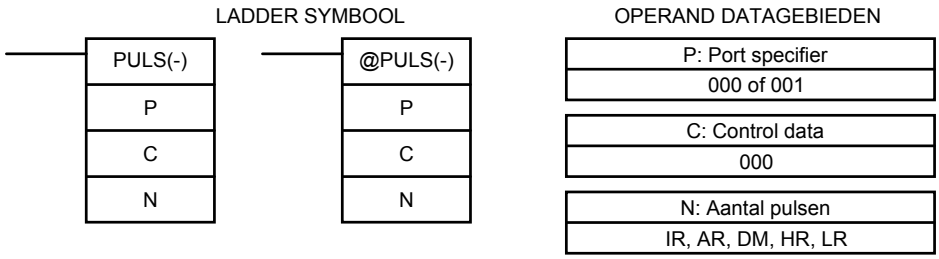
CC = 100 of 200 terwijl een interrupt programma wordt uitgevoerd.

CC = 100 terwijl alle interrupts gemaskeerd zijn.

CC = 200 terwijl alle interrupts gedemaskeerd zijn.

CC en / of D liggen niet binnen de gespecificeerde grenzen.

### *4.22.7 Puls - PULS(—)*




**Beperkingen** N en N+1 moeten in hetzelfde datagebied liggen. DM6143 t/m DM6655 kan niet worden gebruikt voor N.

**Omschrijving** PULS(—) wordt gebruikt om parameters in te stellen een pulsuitgang die later in het programma met SPED(—) geactiveerd wordt. De parameter die wordt ingesteld is het aantal pulsen dat wordt uitgestuurd in independent mode. Aangezien PULS(—) een relatief lange executietijd heeft is het aan te raden om deze instructie gedifferentieerd uit te voeren en alleen te activeren als dit nodig is. Lees de sectie “CPM1A pulsuitgang functie instellen en gebruik” op pagina 36 voor meer informatie over het gebruiken van pulsuitgangen.

**Poort Specifier (P)** De poort specifier geeft de locatie aan van de pulsuitgang. De parameters die bij de instructie zijn opgegeven op C en N zijn van toepassing op de volgende SPED(—) instructie waarvoor dezelfde poort output locatie is gespecificeerd.

| P | Pulsuitgang locatie |
|---|---|
| 000 | Output bit 010.00 |
| 001 | Output bit 010.01 |

**Control Data (C)** Deze operand moet bij de CPM1A altijd worden ingesteld op 000.

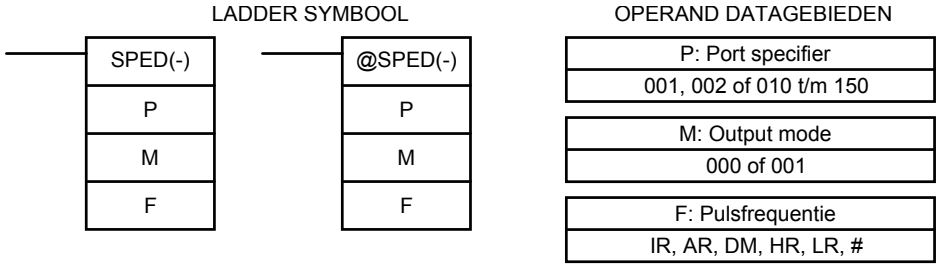
**Frequentie veranderingen** Het aantal pulsen dat ingesteld is zal worden uitgestuurd, zelfs als SPED(—) wordt gebruikt om de pulsfrequentie tijdens bedrijf te veranderen.

Als bijvoorbeeld het ingestelde aantal pulsen 2100 is en de frequentie wordt veranderd van 1KHz naar 100Hz, dan zal de pulsuitsturing stoppen na:
12 s als de pulsfrequentie wordt verander na 1s uitsturing op 1KHz.
3 s als de pulsfrequentie wordt verander na 2s uitsturing op 1KHz.

**Vlaggen**

**ER:** Er is een fout in de instructie instellingen.
De grenzen van een geheugengebied zijn overschreden.
Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.
PULS(—) wordt uitgevoerd in een interrupt subroutine terwijl een puls I/O of highspeed counter instructie wordt uitgevoerd in het hoofdprogramma of een andere interrupt routine.

### *4.22.8 Speed output - SPED(—)*




**Beperkingen** F moet worden ingevoerd in BCD#0000 of #0002 t/m #0200 wanneer een outputbit is gespecificeerd. DM6143 t/m DM6655 kan niet worden gebruikt voor F.

**Omschrijving** SPED(—) wordt gebruikt om de frequentie van de pulsuitsturing op de gespecificeerde uitgang in te stellen of te veranderen. Wanneer de executieconditie uit is, wordt SPED(—) niet uitgevoerd. Wanneer de executieconditie aan is, stelt SPED(—) de pulsfrequentie F in voor de poort of het uitgangsbit dat gespecificeerd wordt door P. M bepaalt de output mode. Aangezien SPED(—) een relatief grote executietijd heeft, kan de cyclustijd worden verminderd

door de gedifferentieerde uitvoering (@SPED(—)) van deze instructie alleen indien nodig uit te voeren. Lees de sectie “CPM1A pulsuitgang functie instellen en gebruik” op pagina 36 voor meer informatie over het gebruiken van pulsuitgangen.

**Poort Specifier (P)**

De poort specifier specificeert de poort of het outputbit waarop de pulsen worden uitgestuurd.

| P | Puls output location |
|---|---|
| 000 | Output bit 010.00 |
| 001 | Output bit 010.01 |

**Output Mode (M)**

De waarde op M bepaald de output mode. Een waarde van 000 geeft de independent mode aan en een waarde 001 geeft de continuous mode aan.

In independent mode zal de pulsuitsturing doorgaan tot één van de volgende situaties optreedt:

**1, 2, 3...**

1. Het aantal pulsen gespecificeerd door de PULS(—) instructie is bereikt. (Voer altijd PULS(—) uit voor SPED(—) wanneer de independent mode wordt gebruikt)
2. De INI(—) instructie wordt uitgevoerd met C=003.
3. SPED(—) wordt opnieuw uitgevoerd met een frequentie instelling (F) op 000.

Wanneer pulsen uitgestuurd worden in independent mode, specificeer het aantal pulsen dan met PULS(—) voordat SPED(—) wordt uitgevoerd. In continuous mode worden pulsen uitgestuurd tot de INI(—) instructie wordt uitgevoerd met C=002 of SPED(—) opnieuw wordt uitgevoerd met F=0000.

**Pulsfrequentie**

De waarde op F stelt de pulsfrequentie voor in units van 10 Hz. Deze waarde kan ingesteld qworden tussen 2 en 200, wat overeen komt met een frequentie van 20Hz tot 2kHz. Wanneer F op 0000 wordt ingesteld zal de pulsuitsturing op de gespecificeerde poort stoppen.

**Opmerking**

De puls uitgang kan niet worden gebruikt als interval timer 0 in werking is. Een pulsuitgang kan maar op één uitgang tegelijkertijd actief zijn.

**Vlaggen**

**ER:** Er is een fout in de instructie instellingen.

SPED(—) wordt uitgevoerd terwijl interval timer 0 actief is.

Indirect geadresseerd DM woord bestaat niet. Inhoud van *DM woord is niet in BCD opgegeven of de grootte van het DM gebied is overschreden.

SPED(—) wordt uitgevoerd in een interrupt subroutine terwijl een puls I/O of highspeed counter instructie wordt uitgevoerd in het hoofdprogramma of een andere interrupt routine.

# 5 Appendix

## 5.1 Conversietabel hexadecimaal, BCD, binair

| Decimaal | Hexadecimaal | BCD | Binair |
|---|---|---|---|
| 0 | 0 | 0000 | 0 |
| 1 | 1 | 0001 | 1 |
| 2 | 2 | 0010 | 10 |
| 3 | 3 | 0011 | 11 |
| 4 | 4 | 0100 | 100 |
| 5 | 5 | 0101 | 101 |
| 6 | 6 | 0110 | 110 |
| 7 | 7 | 0111 | 111 |
| 8 | 8 | 1000 | 1000 |
| 9 | 9 | 1001 | 1001 |
| 10 | A | 00010000 | 1010 |
| 11 | B | 00010001 | 1011 |
| 12 | C | 00010010 | 1100 |
| 13 | D | 00010011 | 1101 |
| 14 | E | 00010100 | 1110 |
| 15 | F | 00010101 | 1111 |
| 16 | 10 | 00010110 | 10000 |

## 5.2 Conversietabel hex, ASCII

De volgende codes kunnen worden gebuikt wanneer MSG(46) of FPD(-) worden gebruikt om data naar een programmeerapparaat te sturen.

| **Rechter Cijfer** | **Linker cijfer** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0, 1, 8, 9** | **2** | **3** | **4** | **5** | **6** | **7** | **A** | **B** | **C** | **D** | **E** | **F** |
| **0** | | | 0 | @ | P | ` | p | | - | @ | P | ` | p |
| **1** | | ! | 1 | A | Q | a | q | ! | 1 | A | Q | a | q |
| **2** | | “ | 2 | B | R | b | r | “ | 2 | B | R | b | r |
| **3** | | # | 3 | C | S | c | s | # | 3 | C | S | c | s |
| **4** | | $ | 4 | D | T | d | t | $ | 4 | D | T | d | t |
| **5** | | % | 5 | E | U | e | u | % | 5 | E | U | e | u |
| **6** | | & | 6 | F | V | f | v | & | 6 | F | V | f | v |
| **7** | | ‘ | 7 | G | W | g | w | ‘ | 7 | G | W | g | w |
| **8** | | ( | 8 | H | X | h | x | ( | 8 | H | X | h | x |
| **9** | | ) | 9 | I | Y | i | y | ) | 9 | I | Y | i | y |
| **A** | | * | : | J | Z | j | z | * | : | J | Z | j | z |
| **B** | | + | ; | K | [ | k | { | + | ; | K | [ | k | { |
| **C** | | , | < | L | ¥ | l | \| | , | < | L | ¥ | l | \| |
| **D** | | - | = | M | ] | m | } | - | = | M | ] | m | } |
| **E** | | . | > | N | ^ | n | → | . | > | N | ^ | n | |
| **F** | | / | ? | O | _ | o | ← | / | ? | O | _ | o | ← |

## 5.3 INDEX

**OMRON ELECTRONICS B.V.**
Wegalaan 61, Postbus 582, 2130 AN  HOOFDDORP
Tel. (023) 568 11 00
Fax (023) 568 11 88 / 568 11 50 (verkoop)
E-mail: omron-nl@eu.omron.com

Uw leverancier:

Cat. No. W353-NL2-02