

Block programming

**CS1/CJ1**

**Quick guide**

**OMRON**

## Indholdsfortegnelse

<b>1. SPECIFIKATION</b> .....	<b>3</b>
<b>2. PROGRAMMERINGS INSTRUKTIONER</b> .....	<b>4</b>
<b>3. BLOK PROGRAMMERING</b> .....	<b>5</b>
3.1. BPRG OG BEND .....	5
<b>4. BETINGELESER PROGRAMMERING</b> .....	<b>6</b>
4.1. PROGRAMMERING MED IF, ELSE OG IEND.....	6
<b>5. EXIT BLOCK PROGRAM</b> .....	<b>7</b>
5.1. HOP UD AF ET PROGRAM VED HJÆLP AF EXIT.....	7
<b>6. CYKLUS STOP OG VENT</b> .....	<b>8</b>
6.1. INSTRUKTIONEN WAIT OG WAIT (NOT).....	8
<b>7. TIMER</b> .....	<b>9</b>
7.1. INSTRUKTIONEN TIMW OG TIMWX .....	9
<b>8. TÆLLER</b> .....	<b>10</b>
8.1. INSTRUKTIONEN CNTW OG CNTWX.....	10
<b>9. LOOP</b> .....	<b>11</b>
9.1. INSTRUKTIONEN LOOP OG LEND .....	11
<b>10. BETINGELSER</b> .....	<b>12</b>

## 1. Specifikation

Denne Quick guide giver en kort beskrivelse af Blok programmerings funktioner i CS1/CJ1 plc serien.

Der kan laves op til 128 blokke i et program (alle taske). Fordelen ved at bruge blok programmering er bla. at alle instruktioner mellem BPRG og BEND kun bliver udført når betingelsen for BPRG er sand.

Blok programmering gør det muligt at lave et program som er vanskeligt at lave i ladder, f.eks. Branches og Step programmering, samt at optimere hastigheden på sit program væsentligt, specielt med WAIT og TIMW instruktionerne.

Hver blok er startet med instruktionen BPRG, alle instruktioner derefter skal skrives i mnemonic.

## 2. Programmerings instruktioner.

Følgende instruktioner findes til blok programmering.

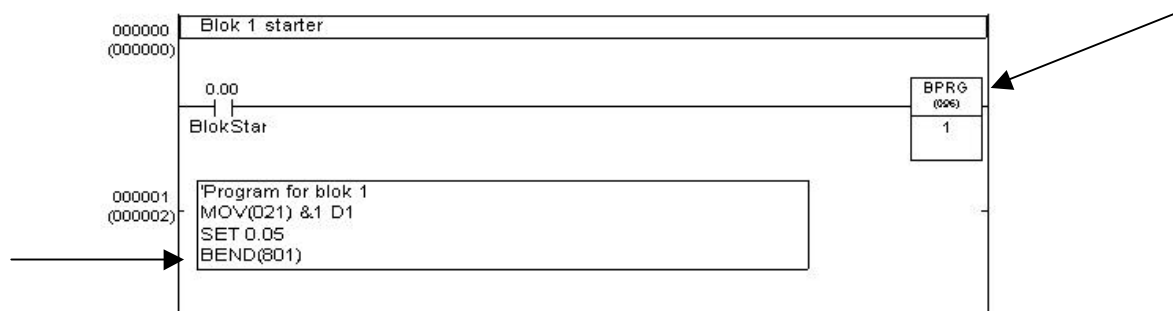
Instruktioner	Mnemonic	Funktions kode
BLOCK PROGRAM BEGIN	BPRG	096
BLOCK PROGRAM END	BEND	801
BLOCK PROGRAM PAUSE	BPPS	811
BLOCK PROGRAM RESTART	BPRS	812
CONDITIONAL BLOCK EXIT (NOT)	EXIT (NOT)	806
IF (NOT)	IF (NOT)	802
ELSE	ELSE	803
IF END	IEND	804
ONE CYCLE AND WAIT (NOT)	WAIT (NOT)	805
TIMER WAIT	TIMW (BCD)	813
	TIMWX (binary)	816
COUNTER WAIT	CNTW (BCD)	814
	CNTWX (binary)	818
HIGH-SPEED TIMER WAIT	TMHW (BCD)	8147
	TMHWX (binary)	815
LOOP	LOOP	809
LOOP END (NOT)	LEND (NOT)	810

### 3. BLOK Programmering

#### 3.1. BPRG og BEND

Blok programmer kan laves alle steder i en Task, for at opdele sit program i yderligere enheder.

En Blok startes med Instruksen BPRG, efterfølgende instruktioner skal være i mnemonic, og programet skal afsluttes med BEND

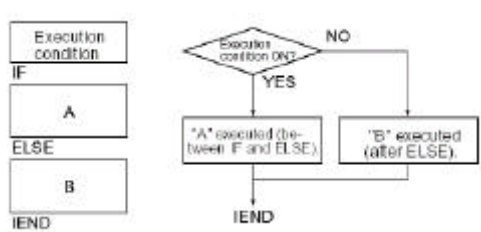


## 4. Betingelses programmering.

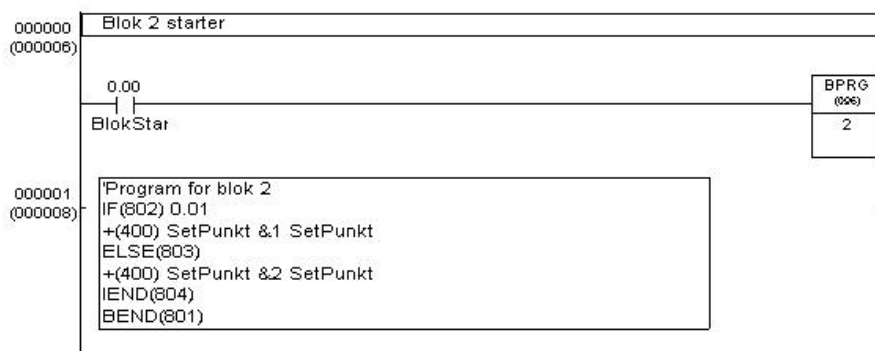
### 4.1. Programmering med IF, ELSE og IEND.

Betingelses programmering kan bruges i program blokken til at hoppe til det næste step, eller afgøre et valg i programmeringen.

Hvis man ønsker at lave et valg på baggrund af en betingelse kan det se ud som følgende:



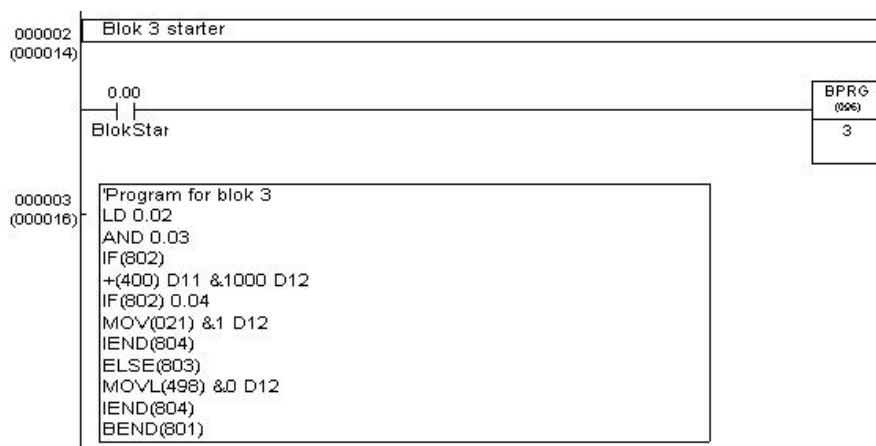
Et eksempel på plc program med betingelses programmering kunne være at an ønsker at lægge værdien 1 til et setpunkt hvis der trykkes på 0.01, hvis der derimod ikke trykkes på 0.01 lægges der i stedet for værdien 2 til setpunkt.



Det er også mulig at have en IF betingelse inde i en IF betingelse.

Følgende blok bliver udført hvis indgang 0.00 er aktiv.

Hvis indgang 0.02 og 0.03 er aktiv, lægges værdien 1000 til D11, og resultatet lægges i D12, hvis indgang 0.04 bliver aktiv lægges værdien 1 i D12. Ellers resettes D12.



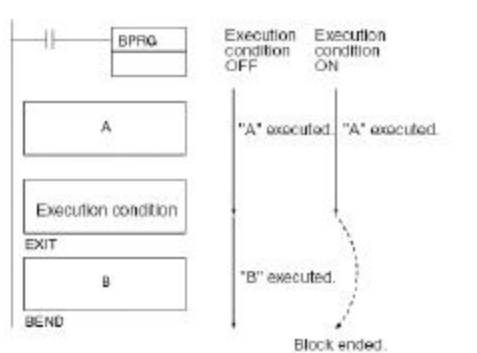
## 5. Exit Block program

### 5.1. Hop ud af et program ved hjælp af EXIT

Det er muligt at afbryde et program forløb, ved hjælp af EXIT eller EXIT (NOT).

Hvis betingelserne før EXIT er opfyldt (ON), vil program forløbet afbrydes og der hoppes til BEND.

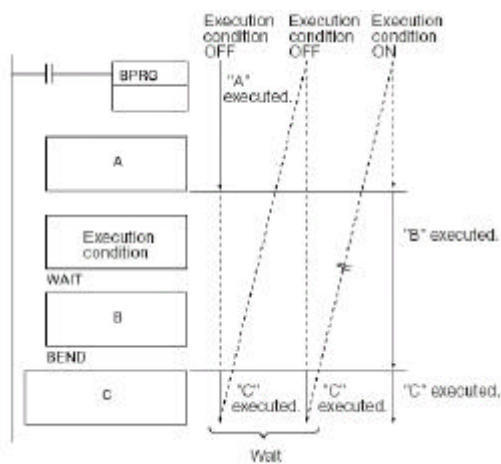
EXIT (NOT), er det samme, blot at betingelserne foran instruktionen være OFF.



## 6. Cyklus stop og vent.

### 6.1. Instruksen WAIT og WAIT (NOT).

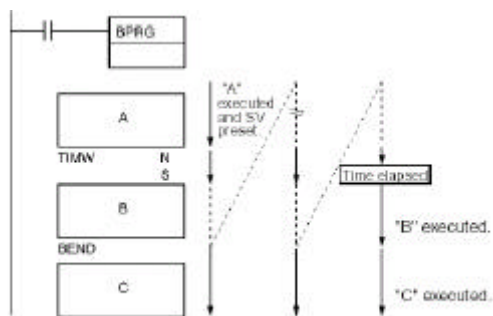
Når et program læser en WAIT instruktion stopper afviklingen af resten af blokken, efterfølgende vil blokken kun bruge tid på at læse status på WAIT instruktionen, og først afvikle programmet når instruktionen aktiveres.



## 7. Timer

### 7.1. Instruktionen TIMW og TIMWX

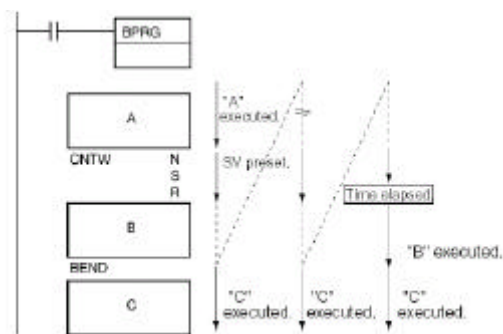
Når et program læser en TIMW (BCD) eller TIMWX (Binary) instruktion stopper afviklingen af resten af blokken, efterfølgende vil blokken kun bruge tid på at læse status på timer instruktionen, og først afvikle blokken igen når timeren er udløbet.



## 8. Tæller

### 8.1. Instruktionen CNTW og CNTWX

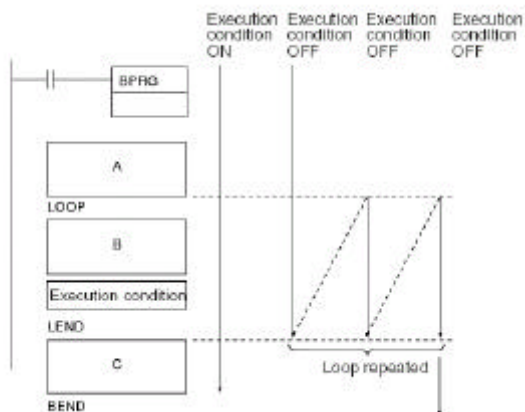
Når et program læser en CNTW (BCD) eller CNTWX(Binary) instruktion stopper afviklingen af resten af blokken, efterfølgende vil blokken kun bruge tid på at læse status på tæller instruktionen, og først afvikle blokken igen når tælleren er udløb.



## 9. Loop

### 9.1. Instruksen LOOP og LEND

Et loop kan bruges til at hoppe tilbage i programmet, og loop programmet ligger mellem LOOP og LEND. Vær opmærksom på at I/O data ikke bliver opdateret i Loopet, så læses der data fra I/O skal man bruge IORF instruksen til at opdatere I/O data.



## 10. Betingelser

Alle software eksempler, program forslag samt principdiagrammer kan og bør ikke opfattes som direkte implementerer i endelig applikationer.

Dersom der ændres i standard menuer samt prædefinerede opsætninger indestår OMRON ikke for ansvar.

Der gøres opmærksom på at Omron Electronics A/S ikke kan holdes ansvarlig for eventuelle tab af data.

Visse programeksempler er udviklet til at bruge bestemte hukommelses område. Dette medfører at der skal tages backup af de hukommelses områder som ikke må gå tabt.

Brugen af Omron Electronics A/S programeksempler er på eget ansvar.

Tabel instruktioner til bl.a.  
Receipt styring

**CS1/CJ1**

**Quick guide**

**OMRON**

## Indholdsfortegnelse

<b>1. SPECIFIKATION</b> .....	<b>3</b>
<b>2. TABEL LAYOUT</b> .....	<b>4</b>
<b>3. OPRET TABEL I PLC'EN</b> .....	<b>5</b>
3.1. OPSÆTNING AF DIM INSTRUKTION.....	5
<b>4. HENT EN BESTEMT RECORD</b> .....	<b>6</b>
4.1. HENT RECORD MED SETR INSTRUKTIONEN. ....	6
<b>5. INDSÆT DATA I EN RECORD</b> .....	<b>7</b>
5.1. INDSÆT RECORD MED SETR INSTRUKTIONEN. ....	7
<b>6. SØG EFTER ET VARE NR.</b> .....	<b>8</b>
6.1. INSTRUKTIONEN SRCH (SØG).....	8
6.2. LOOP INSTRUKTIONER. ....	9
6.2.1. <i>Instruktionen FOR og NEXT</i> .....	9
6.2.2. <i>Instruktionen BREAK</i> . ....	9
6.3. INSTRUKTIONEN MOVR.....	9
6.4. PROGRAM EKSEMPEL. ....	10
6.4.1. <i>Rung 1 – Initialisering af søgning</i> .....	10
6.4.2. <i>Rung 2 – Start Loop søgning</i> .....	11
<b>7. BETINGELSER</b> .....	<b>12</b>

## 1. Specifikation

Denne Quick guide giver en kort beskrivelse af Tabel instruktionerne i CS1/CJ1 plc serien.

Tabel instruktionerne gør det muligt at lave et plc program som selv holder styr på pointer og registrere, så det bliver væsentligt lettere at genbruge eller udvide et program.

Der kan laves op til 16 tabeller i et program (alle taske).

Efterfølgende laves et eksempel på at oprette en tabel, hente data, indsætte en ny record, samt at søge efter et vare nr.

## 2. Tabel layout.

Følgende layout bruges som udgangspunkt for efterfølgende program eksempler.  
Tabellen har 10 Records, med hver 4 kolonner. Endvidere besluttes det at tabellen skal kaldes Tabel nr 1.

Tabel 1

Record nr	Vare nr	Højde	Bredde	Længde
0	1000	11	11	11
1	1001	11	12	11
2	2002	12	13	11
3	2003	12	14	12
4	3004	13	15	12
5	3005	13	16	12
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0

### 3. Opret tabel i plc'en

#### 3.1. Opsætning af DIM instruktion

En tabel oprettes i plc'en ved brug af DIM instruktionen. Tabellen tildeles et nummer, der skal angives hvor mange kolonner og records der skal være i tabellen, og endeligt skal der angives en tabel start adresse.

For at lave "et billed" i plc'en af tabellen beskrevet under punkt 2, opsættes DIM instruktionen på følgende måde:



DIM instruktionen indeholder nu:

1. Tabel nr. = 1
2. Antal kolonner = 4
3. Antal recods = 10
4. Start adresse for tabellen = D1000

Det betyder at denne tabel nu bruger 4x10 ord, fra D1000 til D1039.

Nu skal data lægges ned i plc'en så de ligger på følgende måde:

**Tabel 1**

Record nr	Vare nr	Højde	Bredde	Længde
0	1000 D1000	11 D1001	11 D1002	11 D1003
1	1001 D1004	11 D1005	12 D1006	11 D1007
2	2002 D1008	12 D1009	13 D1010	11 D1011
3	2003 D1012	12 D1013	14 D1014	12 D1015
4	3004 D1016	13 D1017	15 D1018	12 D1019
5	3005 D1020	13 D1021	16 D1022	12 D1023
6	0 D1024	0 D1025	0 D1026	0 D1027
7	0 D1028	0 D1029	0 D1030	0 D1031
8	0 D1032	0 D1033	0 D1034	0 D1035
9	0 D1036	0 D1037	0 D1038	0 D1039

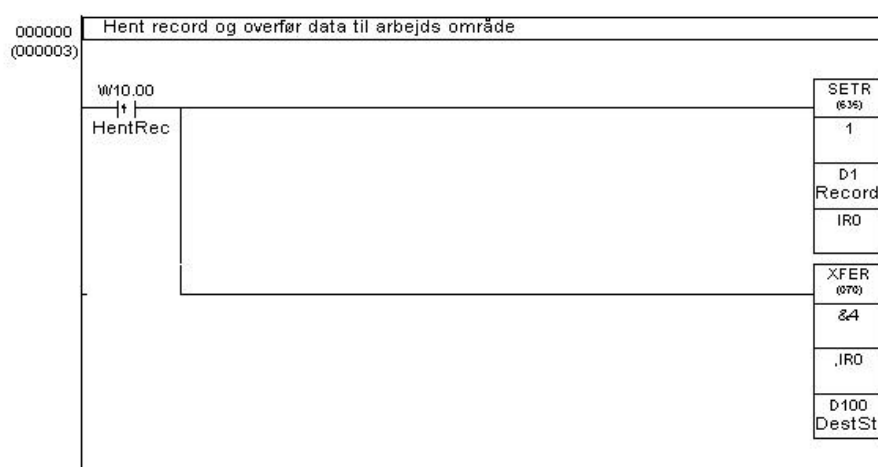
## 4. Hent en bestemt record.

### 4.1. Hent record med SETR instruktionen.

For at hente data i en tabel bruges SETR instruktionen. Her fortæller man hvilken tabel der skal søges i, hvilken record man ønsker overført til sin pointer, samt en pointer adresse.

Her i eksemplet, tabel 1, record nummeret indtastes i D1 og pointeren overføres til Indexregister IRO.

Når pointer adressen er fundet, overflyttes recorden til arbejdsadressen. I eksemplet er hver record 4 ord lang, pointeren ligger i Indexregister IRO og arbejdes området er her D100.



Tabel 1

Vare nr		Højde		Bredde		Længde	
1000	D1000	11	D1001	11	D1002	11	D1003
1001	D1004	11	D1005	12	D1006	11	D1007
2002	D1008	12	D1009	13	D1010	11	D1011
2003	D1012	12	D1013	14	D1014	12	D1015
3004	D1016	13	D1017	15	D1018	12	D1019
3005	D1020	13	D1021	16	D1022	12	D1023
0	D1024	0	D1025	0	D1026	0	D1027
0	D1028	0	D1029	0	D1030	0	D1031
0	D1032	0	D1033	0	D1034	0	D1035
0	D1036	0	D1037	0	D1038	0	D1039

Record nr

3 | D1

Resultat (arbejdsområde)

2003	D100	12	D101	14	D102	12	D103
------	------	----	------	----	------	----	------

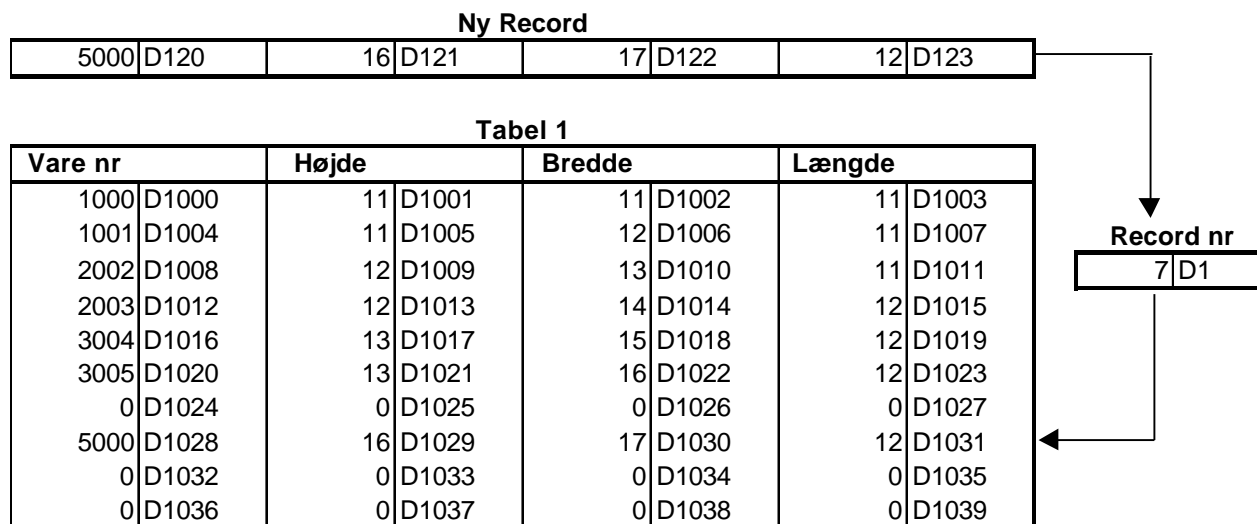
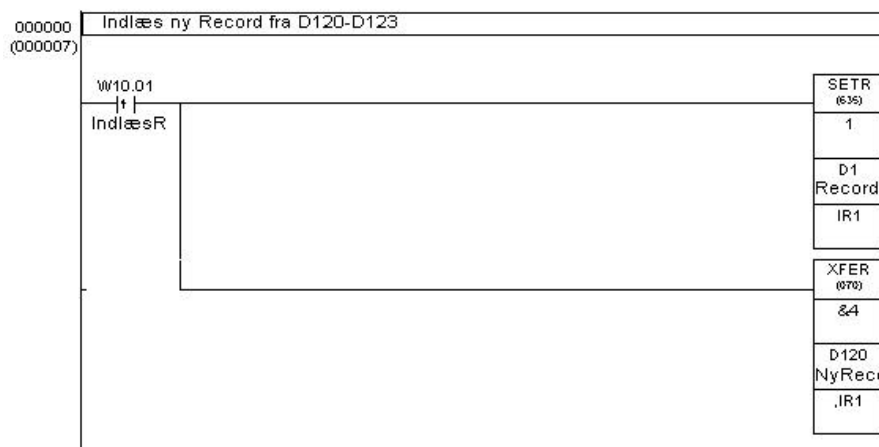
## 5. Indsæt data i en record.

### 5.1. Indsæt record med SETR instruktionen.

For at indlæse data i en tabel bruges SETR instruktionen. Her fortæller man hvilken tabel der skal indlæses i, hvilken record man ønsker overført data til, samt en pointer adresse.

Her i eksemplet, tabel 1, record nummeret indtastes i D1 og pointeren overføres til Indexregister IR1.

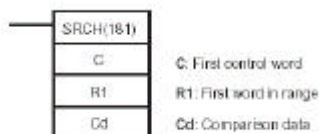
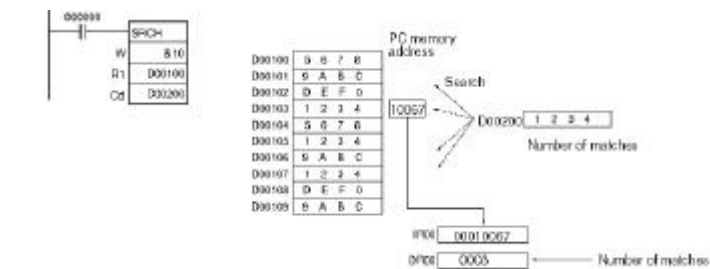
Når pointer adressen er fundet, overflyttes data fra D120-D123 til recorden.



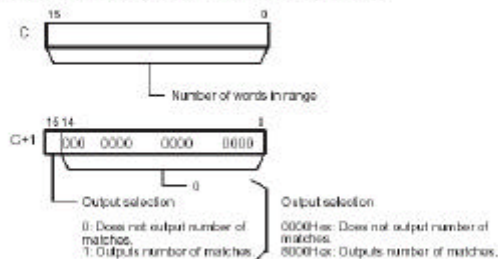
## 6. Søg efter et vare nr.

### 6.1. Instruksen SRCH (søg).

SCRH instruksen kan bruges til at søge efter data i et område af ord. Findes det søgte, skriver instruksen adressen på resultatet i IR0. Hvis der findes flere adresser med samme resultat skriver instruksen antallet i DR0.



**C and C+1: Control words**  
C specifies the number of words in the range and bit 15 of C+1 indicates whether or not to output the number of matches to DR0.



Note: C and C+1 must be in the same data area.

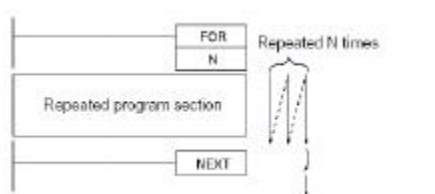
## 6.2. Loop instruktioner.

For at opnå en hurtig og effektiv søgning, bruges der i dette eksempel et LOOP til at lave søgningen i. Det betyder at programmet hopper tilbage i scanet det antal gange der er defineret.

Efterfølgende en kort forklaring på de instruktioner der skal bruges for at lave et loop.

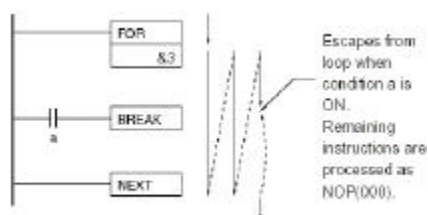
### 6.2.1. Instruktionen FOR og NEXT.

FOR instruktionen bruges til at starte et loop. Der må ikke være nogle betingelser foran instruktionen, og der indtastes i instruktionen hvor mange gange loopet skal gennem løbes. FOR instruktionen holder selv styr på hvor mange gange den skal loope, hver gang den når NEXT hopper den tilbage til FOR.



### 6.2.2. Instruktionen BREAK.

BREAK instruktionen kan bruges til at afbryde et Loop. F.eks hvis der laves en søgning, og der efter 5 loops findes en ok søgning, kan man jo lige så godt afbryde loopet.



## 6.3. Instruktionen MOVR

For at bruge de "nye" pointer i CS1/CS1 bør man vide hvordan MOVR instruktionen fungerer. MOVR bruges til af flytte en start adresse til en pointer. Bem. Det er ikke værdien af ordet men kun adressen der har betydning.

Her vises et eksempel på at man initialisere en pointer til at starte fra CIO0020.

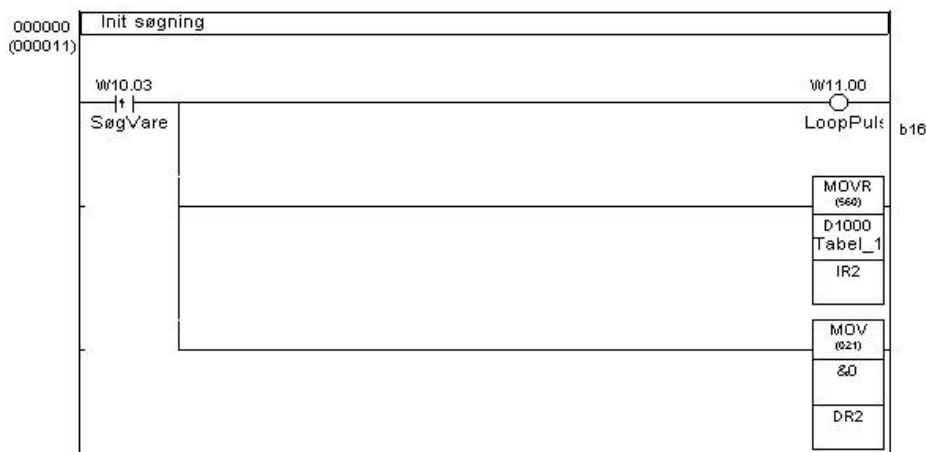


### 6.4. Program eksempel.

Dette program eksempel søger efter vare nr. i tabel 1 som tidligere oprettet. Det der ønskes søgt efter indtastes i D3, søgningen startes med W10.03, resultatet af søgningen kan læses i D140-D143. Hvis D140-D143 er lig med 0, fandt søgningen ikke det pågældende vare nr.

#### 6.4.1. Rung 1 – Initialisering af søgning.

Når søgningen startes sættes et LoopPuls bit. Pointeren IR2 initialiseres til at starte i D1000 (Der hvor tabel 1 starter), DR0 sættes til 0, den bruges til at flytte pointeren i hver søgning.



**Tabel 1**

Vare nr		Højde		Bredde		Længde			
1. gang	2003 D3	1000	D1000	11	D1001	11	D1002	11	D1003
2. gang		1001	D1004	11	D1005	12	D1006	11	D1007
3. gang		2002	D1008	12	D1009	13	D1010	11	D1011
4. gang		2003	D1012	12	D1013	14	D1014	12	D1015
		3004	D1016	13	D1017	15	D1018	12	D1019
		3005	D1020	13	D1021	16	D1022	12	D1023
		0	D1024	0	D1025	0	D1026	0	D1027
		5000	D1028	16	D1029	17	D1030	12	D1031
		0	D1032	0	D1033	0	D1034	0	D1035
		0	D1036	0	D1037	0	D1038	0	D1039

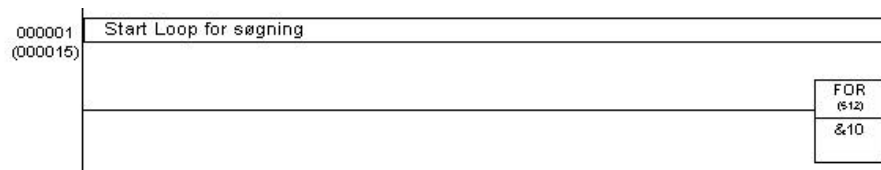
  

**Resultat af søgning**

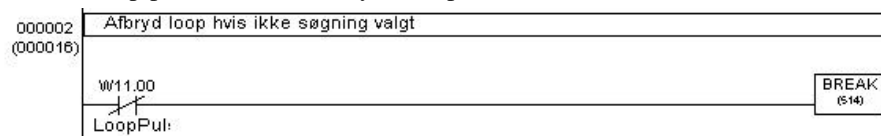
2003	D140	12	D141	14	D142	12	D143
------	------	----	------	----	------	----	------

## 6.4.2. Rung 2 – Start Loop søgning

Loopet gennem løbes 10 gange pga. tabel 1, som indeholder 10 records.

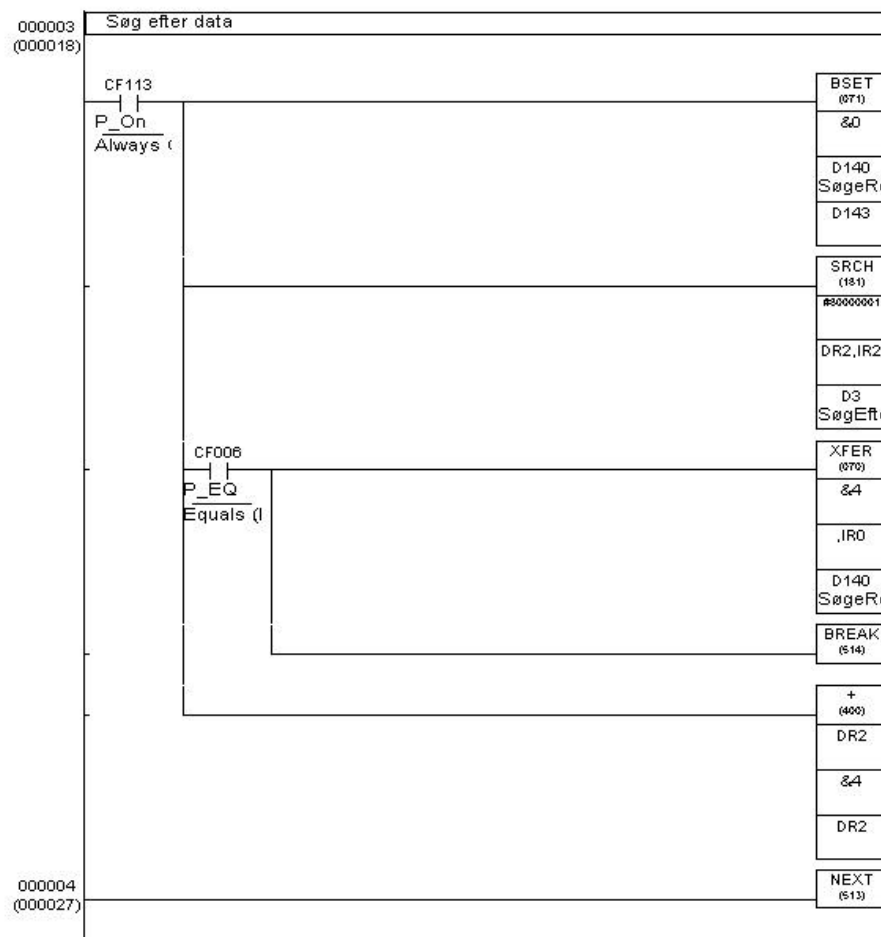


Hvis ikke loop pulsen er aktiv afbrydes loopet med BREAK.



Søgningen starter med at nul stille resultat ordet D140-D143, derefter søges i første record med SRCH, hvis søgningen gav resultat sættes Equal P\_EQ bittet og søgningen overføres med XFER, hvorefter loopet afbrydes.

Hvis ikke søgningen gav resultat flyttes pointeren frem til næste record og loopet starter forfra, i alt 10 gange.



## 7. Betingelser

Alle software eksempler, program forslag samt principdiagrammer kan og bør ikke opfattes som direkte implementerer i endelig applikationer.

Dersom der ændres i standard menuer samt prædefinerede opsætninger indestår OMRON ikke for ansvar.

Der gøres opmærksom på at Omron Electronics A/S ikke kan holdes ansvarlig for eventuelle tab af data.

Visse programeksempler er udviklet til at bruge bestemte hukommelses område. Dette medfører at der skal tages backup af de hukommelses områder som ikke må gå tabt.

Brugen af Omron Electronics A/S programeksempler er på eget ansvar.

Block programming

**CS1/CJ1**

**Quick guide**

**OMRON**

## Indholdsfortegnelse

<b>1. SPECIFIKATION</b> .....	<b>3</b>
<b>2. PROGRAMMERINGS INSTRUKTIONER</b> .....	<b>4</b>
<b>3. BLOK PROGRAMMERING</b> .....	<b>5</b>
3.1. BPRG OG BEND .....	5
<b>4. BETINGELESER PROGRAMMERING</b> .....	<b>6</b>
4.1. PROGRAMMERING MED IF, ELSE OG IEND.....	6
<b>5. EXIT BLOCK PROGRAM</b> .....	<b>7</b>
5.1. HOP UD AF ET PROGRAM VED HJÆLP AF EXIT.....	7
<b>6. CYKLUS STOP OG VENT</b> .....	<b>8</b>
6.1. INSTRUKTIONEN WAIT OG WAIT (NOT).....	8
<b>7. TIMER</b> .....	<b>9</b>
7.1. INSTRUKTIONEN TIMW OG TIMWX .....	9
<b>8. TÆLLER</b> .....	<b>10</b>
8.1. INSTRUKTIONEN CNTW OG CNTWX.....	10
<b>9. LOOP</b> .....	<b>11</b>
9.1. INSTRUKTIONEN LOOP OG LEND .....	11
<b>10. BETINGELSER</b> .....	<b>12</b>

## 1. Specifikation

Denne Quick guide giver en kort beskrivelse af Blok programmerings funktioner i CS1/CJ1 plc serien.

Der kan laves op til 128 blokke i et program (alle taske). Fordelen ved at bruge blok programmering er bla. at alle instruktioner mellem BPRG og BEND kun bliver udført når betingelsen for BPRG er sand.

Blok programmering gør det muligt at lave et program som er vanskeligt at lave i ladder, f.eks. Branches og Step programmering, samt at optimere hastigheden på sit program væsentligt, specielt med WAIT og TIMW instruktionerne.

Hver blok er startet med instruktionen BPRG, alle instruktioner derefter skal skrives i mnemonic.

## 2. Programmerings instruktioner.

Følgende instruktioner findes til blok programmering.

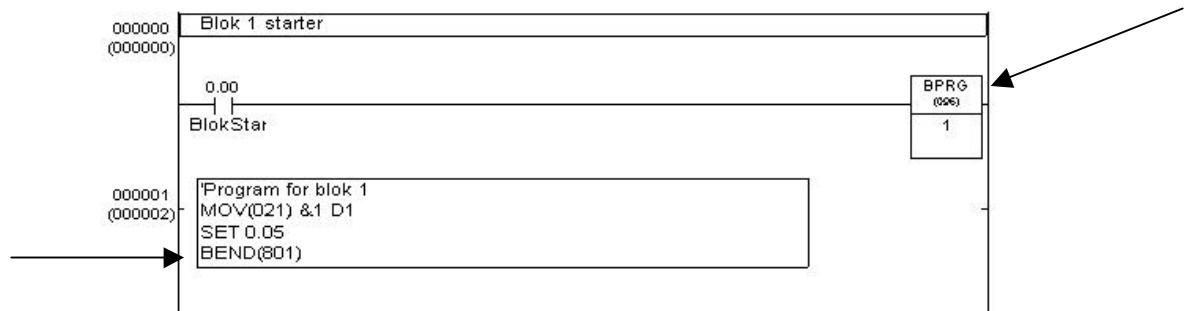
Instruktioner	Mnemonic	Funktions kode
BLOCK PROGRAM BEGIN	BPRG	096
BLOCK PROGRAM END	BEND	801
BLOCK PROGRAM PAUSE	BPPS	811
BLOCK PROGRAM RESTART	BPRS	812
CONDITIONAL BLOCK EXIT (NOT)	EXIT (NOT)	806
IF (NOT)	IF (NOT)	802
ELSE	ELSE	803
IF END	IEND	804
ONE CYCLE AND WAIT (NOT)	WAIT (NOT)	805
TIMER WAIT	TIMW (BCD)	813
	TIMWX (binary)	816
COUNTER WAIT	CNTW (BCD)	814
	CNTWX (binary)	818
HIGH-SPEED TIMER WAIT	TMHW (BCD)	8147
	TMHWX (binary)	815
LOOP	LOOP	809
LOOP END (NOT)	LEND (NOT)	810

### 3. BLOK Programmering

#### 3.1. BPRG og BEND

Blok programmer kan laves alle steder i en Task, for at opdele sit program i yderligere enheder.

En Blok startes med Instruksen BPRG, efterfølgende instruktioner skal være i mnemonic, og programet skal afsluttes med BEND

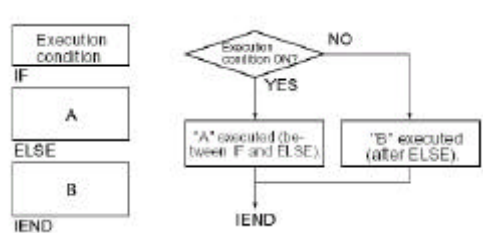


## 4. Betingelses programmering.

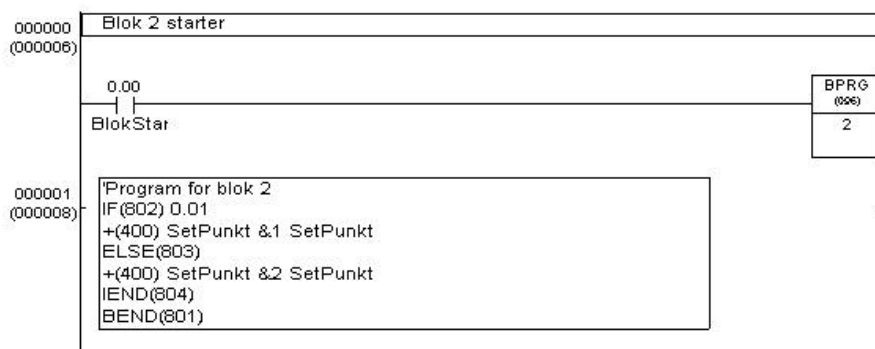
### 4.1. Programmering med IF, ELSE og IEND.

Betingelses programmering kan bruges i program blokken til at hoppe til det næste step, eller afgøre et valg i programmeringen.

Hvis man ønsker at lave et valg på baggrund af en betingelse kan det se ud som følgende:



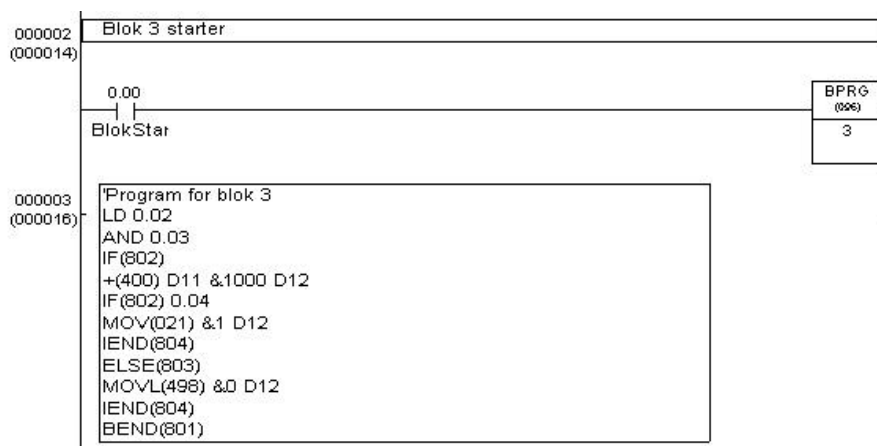
Et eksempel på plc program med betingelses programmering kunne være at an ønsker at lægge værdien 1 til et setpunkt hvis der trykkes på 0.01, hvis der derimod ikke trykkes på 0.01 lægges der i stedet for værdien 2 til setpunkt.



Det er også mulig at have en IF betingelse inde i en IF betingelse.

Følgende blok bliver udført hvis indgang 0.00 er aktiv.

Hvis indgang 0.02 og 0.03 er aktiv, lægges værdien 1000 til D11, og resultatet lægges i D12, hvis indgang 0.04 bliver aktiv lægges værdien 1 i D12. Ellers resettes D12.



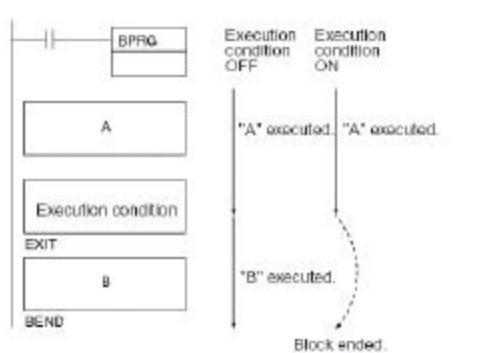
## 5. Exit Block program

### 5.1. Hop ud af et program ved hjælp af EXIT

Det er muligt at afbryde et program forløb, ved hjælp af EXIT eller EXIT (NOT).

Hvis betingelserne før EXIT er opfyldt (ON), vil program forløbet afbrydes og der hoppes til BEND.

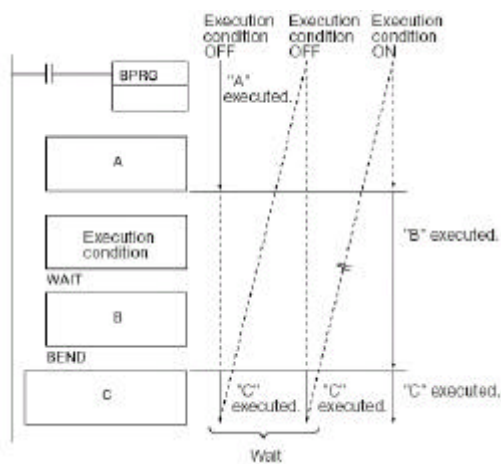
EXIT (NOT), er det samme, blot at betingelserne foran instruktionen være OFF.



## 6. Cyklus stop og vent.

### 6.1. Instruksen WAIT og WAIT (NOT).

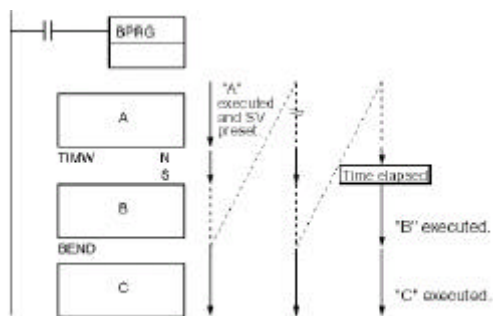
Når et program læser en WAIT instruktion stopper afviklingen af resten af blokken, efterfølgende vil blokken kun bruge tid på at læse status på WAIT instruktionen, og først afvikle programmet når instruktionen aktiveres.



## 7. Timer

### 7.1. Instruktionen TIMW og TIMWX

Når et program læser en TIMW (BCD) eller TIMWX (Binary) instruktion stopper afviklingen af resten af blokken, efterfølgende vil blokken kun bruge tid på at læse status på timer instruktionen, og først afvikle blokken igen når timeren er udløbet.







## 10. Betingelser

Alle software eksempler, program forslag samt principdiagrammer kan og bør ikke opfattes som direkte implementerer i endelig applikationer.

Dersom der ændres i standard menuer samt prædefinerede opsætninger indestår OMRON ikke for ansvar.

Der gøres opmærksom på at Omron Electronics A/S ikke kan holdes ansvarlig for eventuelle tab af data.

Visse programeksempler er udviklet til at bruge bestemte hukommelses område. Dette medfører at der skal tages backup af de hukommelses områder som ikke må gå tabt.

Brugen af Omron Electronics A/S programeksempler er på eget ansvar.