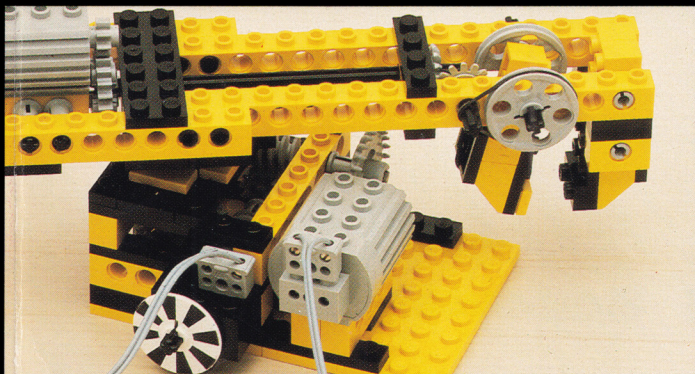
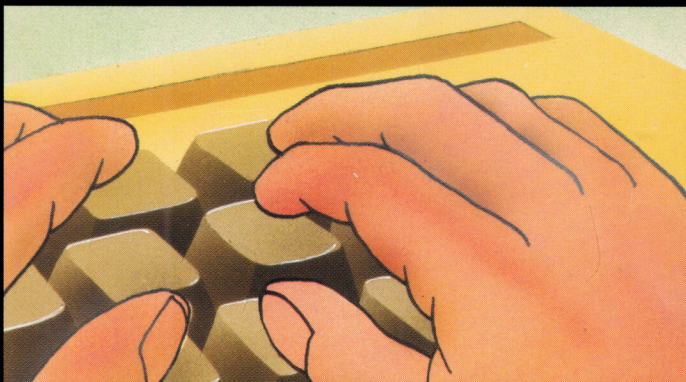
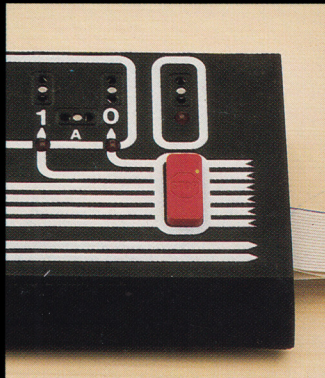


Hold styr på robotterne

- med
Comal 80

Børge R. Christensen

systeme



Børge R. Christensen

Hold styr på robotterne - med Comal 80

sys^{time}

Hold styr på robotterne
– med Comal 80

© 1987 by Børge R. Christensen og forlaget systime a/s, Herning
LEGO Gruppen har copyright på BOX programmet på den medfølgende diskette.

Sat med 9/11 Times hos corner sats, Herning

Trykt hos kn-offset, Herning

Billedredaktion: Børge R. Christensen. Billeder og modeller er anvendt med tilladelse fra LEGO Gruppen.

Omslag: Hans Møller

® LEGO er et registreret varemærke, som ejes af LEGO Gruppen, og som her er brugt med tilladelse herfra.

Fotografisk, mekanisk eller anden gengivelse eller mangfoldiggørelse af denne bog eller dele heraf er ikke tilladt ifølge gældende dansk lov om ophavsret.

LEGO produkterne er specielt udviklet til undervisningsbrug og kan kun købes via specielle skoleforhandlere. Salgsselskabet LEGO Danmark A/S, tlf. 05 33 19 44 oplyser om nærmeste forhandler!

Printed in Denmark 1987
ISBN 87-7251-544-2

forlaget systime a/s
Klokkebakken 20
7400 Herning
Tlf. 07 11 90 11

LEGO Danmark A/S
Nordmarksvej 6
7190 Billund
Tlf. 05 33 19 44

Forord

COMAL-80 har udviklet sig til et programmeringssprog, som har mange anvendelsesmuligheder. Denne bog handler om, hvordan man kan bruge COMAL til styring af robotter. Der er ganske vist tale om små modelrobotter, bygget af LEGO klodser, men sådanne modeller kan programmeres efter helt de samme principper, som bruges ved programmering af de robotter, man bruger i industrien.

Ved at bruge modellerne kan man lære om grundideerne i robotstyring uden at man behøver at udsætte sig selv eller sine omgivelser for de farer eller udgifter, som kunne være en følge af eksperimenter med store robotter.

I bogen benyttes kun de dele af COMAL, som er nødvendige til styring af modellerne. Der er altså ikke tale om nogen egentlig lærebog i COMAL, og man kan anvende kapitel 1-8 uden nogen forkundskaber overhovedet i dette sprog. Det, man har brug for at vide, er forklaret i teksten.

I det sidste afsnit, som består af kapitlerne 9-10, introduceres nogle mere avancerede styringsprincipper, og for at få fuldt udbytte af dette afsnit må man kende de grundlæggende dele af COMAL. Det skal dog understreges, at nævnte afsnit er at betragte som en slags tillæg til de øvrige dele af bogen. Bogens egentlige afsnit består af kapitlerne 1-8 og i disse gives en grundig gennemgang af programmeringen af de tre modeller, som jeg har udvalgt til brug i forbindelse med bogen.

Tønder, januar 1987

Børge R. Christensen

Indholdsfortegnelse

Indledning, hvori vi gør klar til at køre	7
Det dobbelte transportbånd	11
1. Alle skal frem	13
2. Nogen skal til højre, andre til venstre	18
3. En dag kom en vovse	22
4. Vi gør klar til at levere	28
Pariserhjulet	35
5. Her er plads til alle	36
6. Ikke så hastigt i starten	39
Robotarm og transportbånd	44
7. Man må have greb om tingene - og kunne sno sig	45
8. Madsens maleservice	47
Meget mere styr på robotterne	51
9. Om fordelingen ved at have et øje på hver finger	52
10. To robotter laver en aftale	65
Appendix	69
Om at gemme og hente programmer	69
Stikordsregister	72

Indledning, hvori vi gør klar til at køre



Med denne bog kan du lære at holde styr på nogle små *robotter*. De er bygget af LEGO klodser. Du skal bruge dette byggesæt:



Altså det, der hedder *LEGO Technic Control I*, art. nr. 1090.

Til det meste af bogen skal der bruges *1 sæt 1090*. I sidste del af den er der beskrevet et projekt. Til det skal der bruges *2 sæt 1090*.

Sammen med byggesættet følger et *oversigts hæfte* og nogle *byggevejledninger*, som ligger i et papomslag.

Øvelse 0.1

Find oversigts hæftet. På forsiden af det står der *1090* i øverste, venstre hjørne. Slå op på side 2 og 3. Der er billeder af de elementer, som findes i sættet. Find på *side 2* et billede af et hjul med sorte og hvide sektorer og ovenover det en sort klods med et hul i. Find også disse elementer i byggesættet.

/**/

Hjulet kaldes et *tællehjul*. På dets ene side findes 4 sorte og 4 hvide sektorer. På dets anden side er der 8 sorte og 8 hvide sektorer.

Klodsens kaldes en *føler* (LEGO optosensor). Hvis man ser på den fra den ene side, kan man få øje på to huller, et stort og et lille. Det store hul kan bruges som leje for en aksel. Det lille hul kaldes *lys-hullet* og er i virkeligheden et vindue ind til et "elektronisk øje", dvs. et *lysfølsomt element*.

Disse to elementer spiller en meget vigtig rolle, når robotterne optræder, og du kommer til at høre meget om dem i det følgende.

På *side 6* er vist, hvordan tællehjulet kan sættes på en aksel, der enten går gennem føleren eller en klods under føleren. Lyshullet kan ikke ses, da det er gemt bag ved tællehjulet. *Det er vigtigt, at tællehjulet anbringes på den side af føleren, hvor lys-hullet findes.*

Robotterne styres af en datamat - en computer - og i denne bog bruges en COMMODORE 64 eller 128 med en diskteststation. Men datamaten kan ikke uden videre holde styr på robotterne. Den skal have hjælp af en styrekasse (interfacebox, LEGO Interface A, art.nr. 9750 og LEGO kabel, art.nr. 9765) med strømforsyning og et styreprogram.

Et billede af styrekassen findes øverst på *side 7* i oversigtshæftet. Strømforsyningen er en lille, sort kasse med to ledninger. Den ene ledning har et stik til en almindelig stikkontakt, den anden ledning er forsynet med et lille rundt stik, som passer til et hul i styrekassens højre ende.

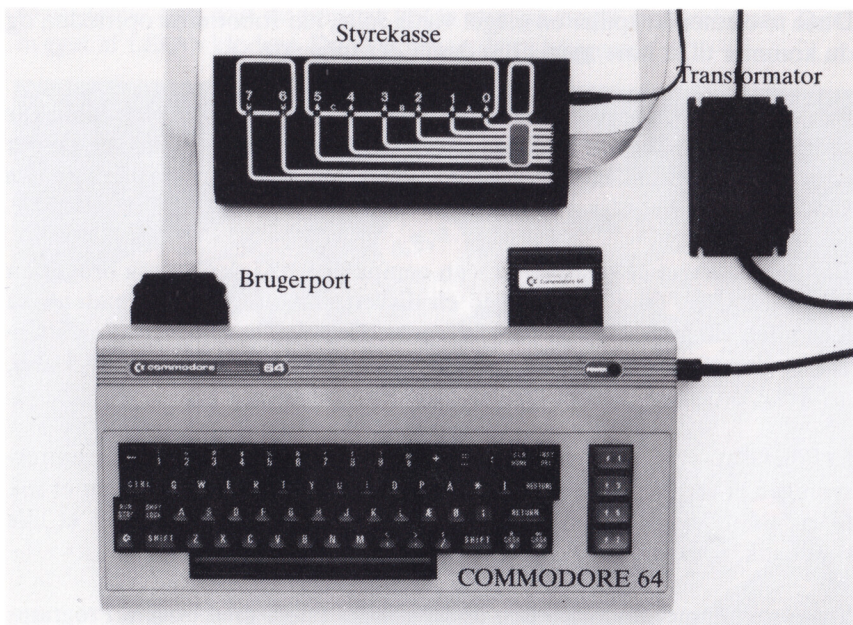
Styreprogrammet findes på den diskette, som følger med bogen. Programmet er skrevet i COMAL-80. Datamaten skal altså være udstyret med en COMAL-kapsel (UniCOMAL 2.01 til C64 eller 2.02 til C128).

Øvelse 0.2

I denne øvelse får du brug for at trykke på nogle af datamatens taster. Når der fx. står <RUN/STOP>, betyder det den taste, hvor der står RUN foroven og STOP forneden. De to tegn < og > viser altså, at der er tale om en bestemt taste. Den måde at skrive på bliver brugt i hele bogen.

Sørg for, at datamaten er slukket, og at der ikke er strøm på styrekassen. Forbind styrekassen med datamatens *brugerport*, som vist på side 10.

Sæt strømforsyningens stik ind i siden af styrekassen og tænd for datamaten. Du ser, at de røde lamper på styrekassen tændes. Tryk på den store, røde *afbryde-knap* til højre på styrekassen. Du ser, at alle lamper undtagen den, der sidder over knappen, bliver slukket. Tryk atter på afbryde-knappen. Så følger den med op, og de røde lamper tændes igen. Det er vigtigt, at du husker denne afbryder. Den kan du altid bruge i en nøds-situation, fx. hvis din robot løber løbsk.



Sæt disketten i diskettestationen. Hvis du bruger en 64'er, så hold én af <SHIFT> tasterne trykket ned og tryk på tasten <RUN/STOP>. Hvis du bruger en 128'er med Comal 2.02, skal du skrive kommandoen

```
load "box/128
```

og trykke på <RETURN> tasten. Så skal diskettestationen begynde at køre, og styreprogrammet bliver hentet ind i datamaten.

Først kommer der et billede på skærmen med teksten BOX. Tryk på <RETURN> tasten. Så glider billedet ovenud af skærmen, og datamaten skriver

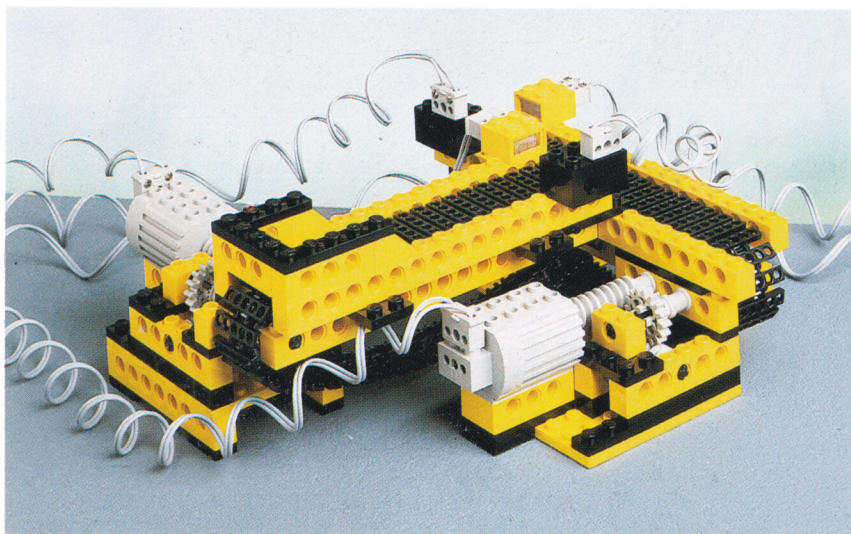
```
* system klar *
```

Samtidigt slukkes alle de røde lamper på styrekassen, undtagen den, der sidder over afbrydeknappen.

```
/**/
```

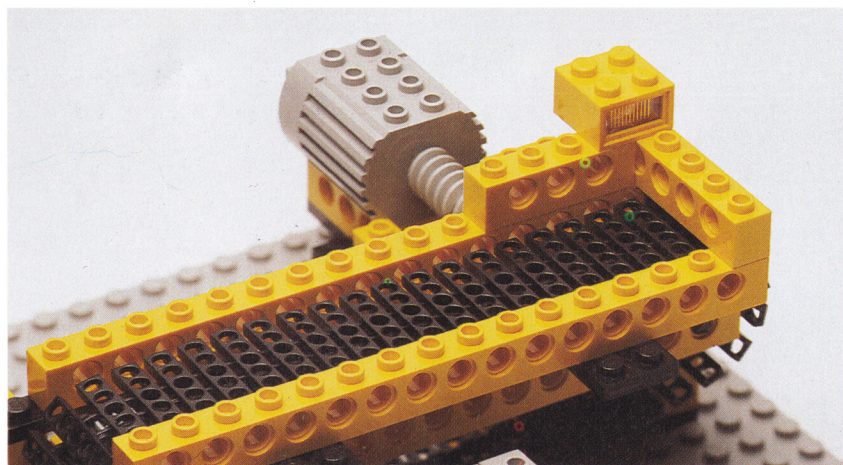
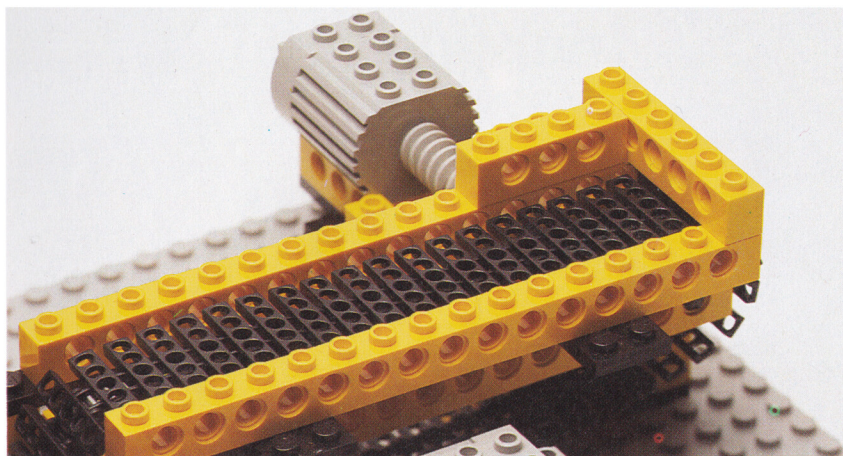
Og så håber vi, at du også er klar til at arbejde videre med at bygge og styre LEGO robotterne.

Det dobbelte transportbånd



I dette afsnit skal du bruge modellen, der er beskrevet i byggevejledningen *1090 D (det dobbelte transportbånd)*. Der skal dog foretages følgende ændringer:

Billede 5, side 7 og billede 6, side 8: Den kant, som er bygget ovenpå båndet, skal erstattes af en mindre kant og en lyssten, som vist herunder. (Andre byggeelementer kan også anvendes).



Billedet side 10: Der skal ikke bygges nogen lyssten og føler på det tværgående transportbånd. På det andet transportbånd skal lyssten og føler bygges på senere.

De på sidste side viste forbindelser mellem modellen og styrekassen skal ikke bruges. Brug i stedet anvisningerne i afsnittets øvelser.

1. Alle skal frem

Inden du begynder, skal *styreassen*, *datamaten* og *styreprogrammet* være klar til brug. Hvis det ikke er i orden, kan du læse i indledningen, hvordan det skal gøres.

Øvelse 1.1

Find billedet af modellen på sidste side af byggevejledningen. Forbind motoren, der er mærket *B*, med styrekassens *udgang 0*.

Skriv så følgende:

```
tn (Ø)
```

og tryk på <RETURN>. Så skal det transportbånd, der drives af motor *B*, begynde at køre fremad. Hvis det ikke gør det, skal stikket i motoren vendes.

Skriv derpå:

```
sl (Ø)
```

og tryk på <RETURN>. Så standser transportbåndet.

Du har sikkert allerede regnet ud, at *tn* betyder *tænd* og *sl* betyder *sluk*. Hvis du har danske bogstaver på den datamat, du bruger, kan du også skrive ordene fuldt ud, altså således:

```
tænd (Ø)
```

og

```
sluk (Ø)
```

I det følgende bliver de korte former brugt, men du må naturligvis gerne bruge de lange former, hvis du bedst kan lide det. De lange former har den fordel, at det er lettere for andre at læse, hvad du har skrevet.

Skriv derpå følgende linje (helt nøjagtigt som vist):

```
tn (Ø); vt (6); sl (Ø)
```

og tryk på <RETURN>. Så kører transportbåndet i 6 sekunder.

Du kunne også skrive linjen sådan:

```
tænd(0); vent(6); sluk(0)
```

og trykke på <RETURN>.

Prøv at flytte markøren (den lille blinkende firkant) op over den sidste linje, du har skrevet, og tryk igen på <RETURN>. Så kører båndet atter i 6 sekunder.

```
/**/
```

betyder, at øvelsen er slut

Sådanne ord som *tn(0)*, *sl(0)*, *vt(6)* eller *tænd(0)*, *sluk(0)* og *vent(6)* kaldes *kommandoer*. Når man ønsker en kommando udført, skal man altid trykke på <RETURN> tasten.

En linje, som fx. *tn(0)*; *vt(6)*; *sl(0)* kaldes en *kommando-linje*. Når linjen er skrevet, skal man trykke på <RETURN> for at få den udført.

Fra nu af skal du selv huske at trykke på <RETURN>, når du vil have en kommando eller en kommando-linje udført.

Øvelse 1.2

Find en byggesten som den her viste:



(Pst! Hvis du kan få fat på nogle ganske almindelige *2x2 byggesten*, kan du bruge dem i stedet for den viste *2x1 byggesten*. Det viser sig, at de er lettere at bruge til vort formål).

Læg byggestenen på transportbåndet i den ende, hvor motoren sidder, og få igen udført kommando-linjen:

```
tn(0); vt(6); sl(0)
```

Så skulle klodsen helst blive kørt helt frem og vipet ned på det tværgående bånd. Hvis det ikke sker, eller hvis båndet bliver ved med at køre længe efter, at klodsen er vipet af, må du ændre på tiden i kommandoen *vt(6)*.

Tryk på tasten <f7>. Så forsvinder alle tekster på skærmen og COMAL skriver igen

```
* system klar *
```

```
/**/
```

Når teksterne på skærmen slettes, forsvinder alle kommandoer og kommando-linjer. Hvis man vil have en af dem udført, må man skrive den igen, og kan ikke mere nøjes med at flytte markøren op over den og trykke på <RETURN>.

Det er ikke særlig praktisk. Vi vil prøve at gemme det hele lidt bedre.

Øvelse 1.3

Tryk på <f1>. Så svarer COMAL med at skrive

```
0010  
0020 proc
```

og markøren står og venter på, at du skal skrive noget efter ordet *proc*.

Skriv ordet "frem", så linje 0020 kommer til at se sådan ud:

```
0020 proc frem
```

og tryk på <RETURN> tasten. På næste linje skriver COMAL nu 0030 og venter igen på, at du skal gøre linjen færdig. Skriv kommando-linjen fra *øvelse 1.1*, så der kommer til at stå sådan:

```
0030 tn(0); vt(6); sl(0)
```

og tryk på <RETURN>. Derpå skifter COMAL igen linje og skriver 0040. På skærmen står der altså:

```
0010  
0020 proc frem  
0030 tn(0); vt(6); sl(0)  
0040
```


med markøren ventende i linje 0040. Tryk på tasten <f3>. Så gør COMAL selv linjen færdig ved at skrive ordet *endproc*. Som tegn på at det var afslutningen, holder COMAL op med at skrive linjenumre og overlader skærmen til dig.

Tryk på <f7>, så skærmen bliver ryddet, og der bliver skrevet

```
* system klar *
```

Det ser ud, som om teksten er borte nu. Men prøv at skrive kommandoen

```
list frem
```

(husk at trykke på <RETURN>). Så viser det sig, at COMAL har gemt det, du har skrevet, i datamatens lager. På skærmen får du denne program-liste:

```
0020 PROC frem
0030   tn(0); vt(6); sl(0)
0040 ENDPROC frem
```

Skriv derpå kommandoen

```
frem
```

og læg mærke til, hvad der sker.

```
/**/
```

En tekst som den, du fik skrevet og gemt i øvelse 1.3, kaldes en *procedure*. Derfor begynder den med ordet PROC og slutter med ordet ENDPROC. Ordet "frem" kaldes dens *navn*. Den linje, der står mellem PROC-linjen og ENDPROC-linjen, kaldes procedures *krop*. Du skal senere se, at der kan være mange linjer i en procedures krop.

Når man skriver procedures navn og trykker på <RETURN>, bliver de linjer, som står i kroppen, udført. Hver gang, du skriver *frem*, får du altså udført linjen *tn(0); vt(6); sl(0)*. Der er to fordele ved at sætte kommandoerne ind i en procedure: For det første bliver de gemt i datamatens lager, og for det andet kan du få dem udført ved blot at skrive procedures navn.

Du kan altid få en procedure at se ved at skrive ordet *list* efterfulgt af procedurens navn, altså fx. *list frem*.

Lidt ekstra: Hvis du foretrækker at skrive de lange former af kommandoerne, kan proceduren *frem* også skrives sådan:

```
0020 PROC frem
0030   tænd(0)
0040   vent(6)
0050   sluk(0)
0060 ENDPROC frem
```

Hvis du bruger denne form, må du ikke samtidigt bruge den i øvelsen foreslåede. *Man må aldrig have to procedurer med samme navn!*

Bemærkning: Nu er du færdig med første kapitel. Hvis du ønsker at holde en pause og slukke for datamaten, skal du først gemme de procedurer, du nu har skrevet, på en diskette. Til sidst i denne bog findes et afsnit, der hedder *om at gemme og hente programmer*. Der kan du læse, hvordan du skal bære dig med at gemme det, du nu har skrevet.

2. Nogen skal til højre, andre til venstre

Hvis du har afbrudt dit arbejde og slukket for datamaten eller brugt den til andre programmer, skal du huske at hente styreprogrammet ind igen, som forklaret i *indledning*. Dernæst skal du hente de procedurer, som du gemte på disketten i slutningen af kapitel 1. Læs i afsnittet *om at gemme og hente programmer*. Der står, hvordan du skal bære dig ad.

Øvelse 2.1

Forbind den motor, der er mærket *C* med styrekassens *udgange 2 og 3*, dvs. sæt stikket i på tværs ud for bogstavet *B* på kassen.

Skriv kommandoen

tn(2)

eller: **tænd(2)**

Så skal båndet køre til højre, set fra siden ind mod bånd nr. 1. Hvis det ikke er tilfældet, skal stikket i motoren *C* vendes.

Stand motoren med kommandoen

sl(2)

eller: **sluk(2)**

Læg en klods på bånd nr. 1 og skriv kommandoen

frem

og derpå kommando-linjen

tn(2); vt(5); sl(2)

eller: **tænd(2); vent(5); sluk(2)**

Så skulle klodsen gerne være kørt frem af bånd nr. 1 og ført til højre af bånd nr. 2. Hvis klodsen ikke når at blive kørt helt ud af bånd nr. 2, må tallet i *vt(5)* ændres, så motoren kører i længere tid.

Den nye kommando-linje vil vi også have sat ind i en procedure. Tryk på <f1>. COMAL svarer med at skrive

ØØ5Ø

ØØ6Ø PROC

Skriv navnet *højre* (eller *hoejre*, hvis der ikke er danske tegn på datamaten) og tryk på <RETURN>. Indtast kommando-linjen *tn(2); vt(5); sl(2)*. Tryk igen <RETURN> og slut med at trykke <f3>.

Tryk på <f7> og skriv kommandoen

list højre

eller: **list hoejre**

Så skulle du få en liste som denne på skærmen

```
0080 PROC højre
0090   tn(2); vt(5); sl(2)
0100 ENDPROC højre
```

Læg en klods på bånd nr. 1 og skriv kommando-linjen

frem; højre

eller: **frem; hoejre**

Når linjen er udført, ligger klodsen på bordet til højre for bånd nr. 2. Ikke? Ellers må du prøve igen.

//**

Lidt ekstra: Du kan også skrive proceduren *højre* således:

```
PROC højre
  tænd(2); vent(5); sluk(2)
ENDPROC højre
```

Eller sådan:

```
PROC højre
  tænd(2)
  vent(5)
  sluk(2)
ENDPROC højre
```

Opgave 1

Skriv en procedure, som får bånd nr. 2 til at lægge en klods til venstre. Giv proceduren navnet *venstre*. Den skal altså begynde således:

```
0090
0100 proc venstre
0110 ...
```

Hvis du ikke kan huske, hvordan du skal begynde og slutte, må du se efter i *øvelse 1.3*. Tænk også på, at stikket til motoren blev sat i *udgang nr. 2 og 3*, og vi skrev *tn(2); vt(5); sl(2)* for at få lagt klodsen ud til højre. Nu skal den i stedet til venstre....

Når du har afsluttet proceduren, så tryk på <f7>. Læg en klods på bånd nr. 1 og afprøv dine procedurer med kommando-linjen:

```
frem; venstre
/**/
```

Øvelse 2.2

Forbind den lampe, som blev anbragt på bånd nr. 1 over motoren, med styrekassens *udgang nr. 1*. Skriv derpå disse to linjer (husk <RETURN> efter hver):

```
25 sl(1)
35 tn(1)
```

eller: 25 sluk(1)
eller: 35 tænd(1)

Skriv kommandoen

```
list frem
```

og læg mærke til, at de to linjer er blevet sat ind i proceduren. Læg også mærke til, at de er blevet sat ind *efter nummerorden*.

Skriv kommandoerne:

```
renum
list frem
```

betyder: "pæne linjenumre"

Læg mærke til linjenumrene! Tryk <f7> og giv kommandoen

```
tn(1)
```

Nu lyser lampen som tegn på, at du må lægge en klods på bånd nr. 1. Skriv derpå kommando-linjen

```
frem; højre
```

Læg mærke til, hvornår lampen igen tændes, og læg en klods mere på, når det sker. Så snart den første klods er læsset af til højre, kan du skrive kommandoen

frem; venstre

Læg atter mærke til, hvornår lampen tændes.

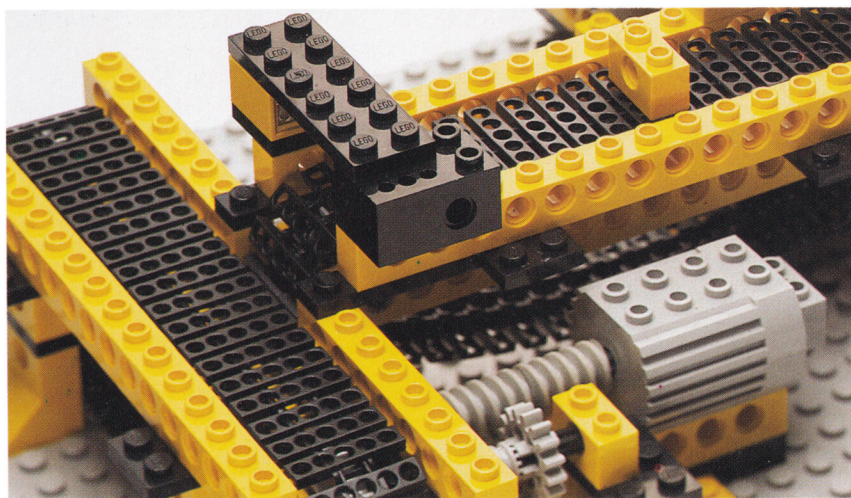
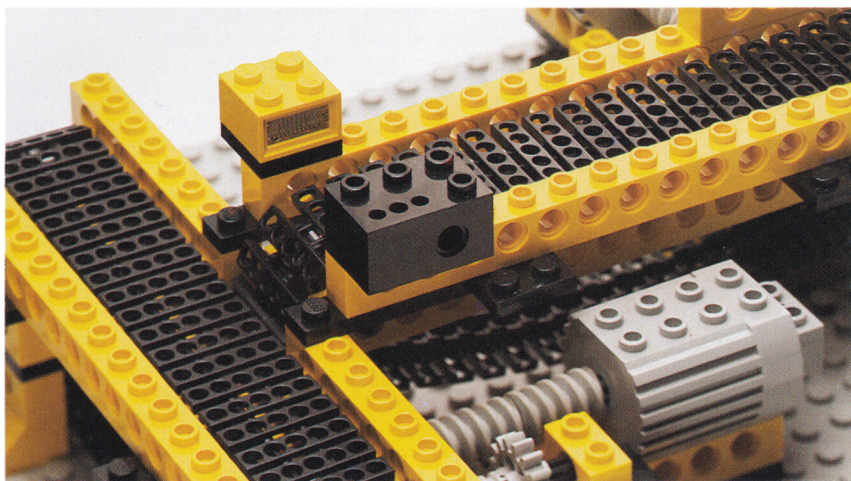
/**/

Bemærkning. Et maskine, der styres af et ur eller en datamat, kaldes også en *robot* eller en *automat*. Det dobbelte transportbånd er et eksempel på en robot. Det ord vil blive brugt i det følgende.

3. En dag kom en voyse

Når bånd nr. 1 startes, kører det altid i et bestemt antal sekunder, nemlig det, du har angivet i *vent*-kommandoen. Det er ikke altid praktisk. Måske bliver klodsen ikke anbragt helt henne ved båndets begyndelse, og så kører det i længere tid end nødvendigt. Vi vil prøve at lave det sådan, at robotten selv kan finde ud af, hvornår klodsen er nået frem til bånd nr. 2.

Først skal vi have modellen udbygget lidt. Vi skal have anbragt en *lampe* og en *føler* på bånd nr. 1 som vist herunder:



Øvelse 3.1

Forbind lampen overfor føleren med styrekassens *særlige udgang*, som sidder over den røde afbrydeknop. Du vil se, at lampen straks begynder at lyse.

Forbind derpå føleren (den store sorte klods) med *indgang nr. 6*.

Prøv at stikke en finger ind mellem lampen og føleren og læg mærke til den lille *grønne lampe*, som sidder ud for nr. 6 på styrekassen. Tag fingeren ud igen, og se atter på den grønne lampe. Prøv et par gange til.

Du ser, at hver gang noget kommer imellem lampen og føleren, lyser den grønne lampe. Sådan en opstilling vil vi kalde en *vagthund* eller en *vovse*. Hver gang, noget kommer forbi, giver vovsen sig til at "gø". Det er ganske vist kun datamaten, der kan "høre" den, du kan blot se, at den grønne lampe lyser.

Læg en klods på båndet og skriv kommando-linjen:

```
tn(Ø); vpv(6); sl(Ø)
```

Læg mærke til, hvornår båndet standser. Fjern klodsen fra hundehuset og læg den på båndet igen, men sådan, at den ligger et stykke fremme på det. Få kommando-linjen ovenfor udført en gang til, og læg atter mærke til, hvornår båndet standser.

Kommandoen *vpv(6)* betyder *vent'på'vovse(6)*, altså "vent på, at den vovse, der er sluttet til nr. 6, giver sig til at gå". Kommando-linjen ovenfor kan også skrives sådan:

```
tænd(Ø); vent'på'vovse(6); sluk(Ø)
```

Desværre er datamaten lidt for hurtig til at standse motoren. Klodsen når ikke at blive kørt ud af båndet, før hunden har stoppet det. Læg en klods på båndet og prøv med denne kommando-linje:

```
tn(Ø); vpv(6); vt(1/2); sl(Ø)
```

Nu lader vi båndet køre i et halvt sekund efter at vovsen har opdaget klodsen. På den måde kan klodsen nå at blive vippet ned på bånd nr. 2.

Linjen kan også se sådan ud:

```
tænd(Ø); vent'på'vovse(6); vent(1/2); sluk(Ø)
```

Giv kommandoen

```
list frem
```

Styr markøren op i den linje, hvor der står

```
tn(Ø); vt(6); sl(Ø)
```

og skriv den nye kommando-linje i stedet for. Det letteste er at stille markøren i begyndelsen af teksten og skrive den nye tekst oveni den gamle (pas på ikke at røre linjenummeret!). Husk også at trykke på «RETURN», når du er færdig med at rette.

Tryk på «f7», læg en klods på bånd nr. 1 og skriv kommando-linjen:

```
frem; venstre
```

Prøv et par gange til, og anbring hver gang klodsen forskellige steder på bånd nr. 1.

/**/

Lidt ekstra: Hvis du har skrevet proceduren *frem* sådan:

```
ØØ2Ø PROC frem
ØØ3Ø   tænd(Ø)
ØØ4Ø   vent(6)
ØØ5Ø   sluk(Ø)
ØØ6Ø ENDPROC frem
```

er det meget let at rette den. Du skal blot ændre linje 40 til:

```
4Ø vent'på'vovse(6)
```

og indsætte linjen

```
45 vent(1/2)
```

Kommandoen *vent(1/2)* kan også skrives således: *vent(0.5)*. Du må altså gerne bruge *decimalbrøker*, når du skriver til datamaten. Når du får skrevet en liste over en procedure, vil du se, at datamaten laver *vent(0.5)* om til *vent(.5)*. Den kaster altså nullet foran decimalpunktummet bort. Sådan må du også skrive det, hvis du synes, det er lettere. Forresten kender du sikkert skrivemåden fra din lommeregner.

Øvelse 3.2

Tryk på <f7> og skriv disse kommandoer:

```
tn(1)
for i=1 to 3 frem; venstre
```

og lad båndene ekspedere 3 klodser. Skriv derpå denne kommando-linje:

```
for i=1 to 2 frem; højre
```

og læg mærke til, hvor mange klodser, der nu bliver ekspederet.

```
/**/
```

Opgave 2.

Skriv en kommando-linje, som får robotten til at ekspedere 4 klodser *til højre*.

Skriv derpå een, som får robotten til at ekspedere 5 klodser *til venstre*.

```
/**/
```

En kommando-linje, der begynder sådan: *for i=1 to 4 ...*, får COMAL til at tælle fra 1 til 4 og hver gang udføre det, som står til sidst på linjen. Hvis du fx. skriver: *for i=1 to 4 frem; venstre*, tæller COMAL 1, 2, 3, 4 og udfører hver gang kommandoerne: *frem; venstre*. Resultatet må derfor blive, at der sendes fire klodser frem og til venstre.

I den næste øvelse skal du sætte kommando-linjer, som de lige viste, ind i nogle nye procedurer, men først skal du læse en

Meget vigtig bemærkning: I det foregående har du set, at hver gang en procedure skal skrives, kan man lettest begynde med at trykke på <f1>. Og man kan afslutte den ved at trykke på <f3>. COMAL sørger selv for at skrive linjenumrene. *Det er kun, når du skal føje nye linjer ind i den, du selv skal skrive linjenumrene.*

Derfor bliver linjenumrene ikke angivet mere, og hvis nye linjer skal ind sættes, må du selv prøve at finde ud af, hvilke numre de skal have.

Øvelse 3.3

Nu vil vi sætte kommando-linjer, som de i øvelse 2.3 viste, ind i passende procedurer. Tryk <f1> og skriv en procedure som denne:

```
proc giv'hj(x)
  for i=1 to x frem; højre
```

og tryk <f3> for at afslutte den. Tryk på <f7> og skriv kommandoen:

```
list giv'hj
```

Så skulle du få en liste som denne frem på skærmen:

```
PROC giv'hj(x)
  FOR i:=1 TO x DO frem; højre
ENDPROC giv'hj
```

Din liste har naturligvis linjenumre, men teksten skal være den samme! Læg også mærke til, at COMAL selv har føjet det engelske ord DO ind foran kommando-linjen *frem; højre*. Ordet DO betyder "udfør" og har god mening. Hele linjen, der begynder med FOR kan oversættes sådan: "Tæl fra 1 til x og UDFØR hver gang kommandolinjen *frem; højre*".

Skriv nu kommandoerne:

```
tn(1)
giv'hj(3)
```

og lad robotten ekspedere 3 klodser.

Giv derpå kommandoer, som får robotten til at ekspedere 2 klodser. Og 4 klodser (husk at få *tn(1)* med hver gang).

*/**/*

Her er lidt forklaring til proceduren *giv'hj*:

<pre>giv'hj(3)</pre>	<i>det er din kommando</i>
<pre>PROC giv'hj(x)</pre>	<i>x bliver sat til 3</i>
<pre> FOR i:=1 TO x DO frem; højre</pre>	<i>COMAL tæller fra 1 til 3, og</i>
	<i>udfører hver gang: frem; højre</i>
<pre>ENDPROC giv'hj</pre>	<i>Sådan!</i>

Hvis du i stedet skriver: *giv'hj(4)*, bliver x sat til 4, og du får kommandoerne *frem*; *højre* udført 4 gange, osv.

Opgave 3

Skriv en proceduren *giv'vn*, så den sender klodser til venstre, på samme måde som *giv'hj* sender klodser til højre. Husk at begynde med at trykke på <f1> og afslutte med at trykke på <f3>. Brug dette procedurehoved:

```
PROC giv'vn(y)
```

Når proceduren er skrevet færdig, så tryk på <f7> og skriv kommandoen

```
list giv'vn
```

så du kan se, om proceduren er i orden. Giv kommandoerne

```
tn(1)  
giv'vn(3)
```

og lad robotten lægge 3 klodser ud til venstre. Prøv også med andre talværdier i kommandoen *giv'vn*.

```
/**/
```

4. Vi gør klar til at levere

Øvelse 4.1

Tryk på <f1> og skriv denne procedure:

```
proc giv(x,y)
  giv'hj(x)
  giv'vn(y)
```

Tryk <f3> for at afslutte den. Skriv kommandoen

```
list giv
```

og kontroller, at du får denne udskrift på skærmen:

```
PROC giv(x,y)
  giv'hj(x)
  giv'vn(y)
ENDPROC giv
```

Tryk på <f7> og skriv kommandoerne:

```
tn(1)
giv(3,2)
```

Læg en klods på bånd nr. 1, hver gang lyset tænder.

```
/**/
```

Her er lidt forklaring til proceduren i *øvelse 4.1*:

<pre>giv(3,2)</pre>	<i>det er din kommando</i>
<pre>PROC giv(x,y)</pre>	<i>x bliver 3, og y bliver 2</i>
<pre> giv'hj(x)</pre>	<i>3 klodser til højre, og</i>
<pre> giv'vn(y)</pre>	<i>2 klodser til venstre</i>
<pre>ENDPROC giv</pre>	<i>Sådan!</i>

Opgave 4

Skriv en kommando, som får robotten til at levere 4 klodser til højre og 3 til venstre.

```
/**/
```

Du har sikkert lagt mærke til, at bånd nr. 1 er temmelig utålmodig. Hvis du er for længe om at lægge en klods på det, begynder det at køre af sig selv.

Hvis der er tale om et stort transportbånd, der skal flytte tunge kasser, kan det måske blive vanskeligt eller farligt at lægge en kasse på, når først båndet kører. Vi vil prøve at få bedre styr på det.

Øvelse 4.2

Skriv kommandoen

```
list frem
```

Proceduren begynder sådan:

```
PROC frem  
sl(1)  
...
```

Indsæt følgende kommando mellem *procedurehovedet* og *sl(1)*:

```
afvent
```

Husk at give kommandoen et linjenummer, der ligger *mellem nummeret på hovedet og nummeret på linjen sl(1)*.

Skriv kommandoen

```
list frem
```

og se efter, at linjen med *afvent* er kommet på plads. Tryk så på <f7> og skriv kommando-linjen

```
tn(1); frem
```

Lampen tænder, men så sker der tilsyneladende ikke mere. Prøv nu at lægge en klods på bånd nr. 1 og tryk på een af datamatens taster.

```
/**/
```

Kommandoen *afvent* får datamaten til at vente, til du trykker på en taste. Ved hjælp af denne kommando kan vi altså få robotten til at vente på os, når vi ønsker det.

Nu har vi efterhånden så godt styr på tingene, at vi kan afslutte vort første projekt.

Øvelse 4.3

I de følgende øvelser skal du skrive en procedure, som er lidt anderledes end de foregående. Den skal bruges til at fortælle robotten, hvor mange klodser, du vil have til højre, og hvor mange til venstre.

Inden du begynder at skrive proceduren, skal du lære nogle nye COMAL kommandoer at kende. Skriv først kommandoen:

```
page
```

(Husk at trykke på <RETURN>). Så bliver skærmen forhåbentligt ryddet for alt, hvad der står på den. Ordet "page" betyder "ny side", og det er, hvad du har fået. Skriv så kommandoen (omhyggeligt!):

```
print at 2,1: "mojn"
```

Det ord, der står i anførselstegn, bliver skrevet på skærmens *linje 2* og begynder helt ude i venstre side på *plads nr. 1*. Styr markøren op over kommandoen, ret tallene 2,1 til 3,2 og tryk <RETURN> igen. Læg mærke til, hvor ordet "mojn" nu bliver skrevet. Ret så kommandoen til:

```
print at 4,5: "mojn"
```

Har du gættet, hvad det første tal betyder? Og det andet? Prøv også med disse kommandoer

```
page
```

```
for i=2 to 24 print at i,i: "mojn"
```

Du kan også prøve med dit eget navn i stedet for "mojn", men pas på, du ikke bliver svimmel!

Skriv igen kommandoen

```
page
```

og skriv derpå denne kommando:

```
input at 5,2: x
```

Forhåbentligt skriver COMAL et "?" i 5'te linje på 2'den plads. Men så sker der tilsyneladende heller ikke mere. Prøv at indtaste et 7-tal og tryk på <RETURN>.

Ret kommandoen til:

```
input at 10,5: x
```

og indtast eet eller andet tal.

Skriv dernæst kommandoen

```
print at 20,5: x
```

og læg mærke til hvilket tal, du får skrevet ud på skærmen, og hvor det bliver skrevet. Prøv input-kommandoen igen, men indtast et nyt tal efter spørgsmålstegnet. Prøv at gætte, hvilken værdi x nu har, og find ud af, om det passer, ved at spørge med print-kommandoen.

```
/**/
```

Efter øvelse 4.3 kan vi opsummere:

Kommandoen PAGE rydder skærmen.

En kommando, der begynder med PRINT, skriver noget på skærmen, og med AT kan vi bestemme, hvor det skal begynde.

Kommandoer, der begynder med INPUT, kan bruges, når man vil indtaste talværdier. I en INPUT-kommando skal der være en *variabel*, fx. x eller y , til at "samle talværdien op".

Øvelse 4.4

Tryk på <f1> og skriv denne procedure:

```
proc bestil
page
print at 5,5: "Til højre"
input at 5,20: x
print at 7,5: "Til venstre"
input at 7,20: y
```

Tryk på <f3> for at afslutte proceduren og giv kommandoen:

```
list bestil
```

og se efter, at alle kommandoer er kommet med på listen. Afprøv proceduren ved at trykke på <f7> og give kommandoen

```
bestil
```

Indtast *værdien 3* til *x* og *værdien 5* til *y*. Kontroller, at *x* og *y* har fået disse værdier ved at skrive kommandoerne:

```
print x
print y
/**/
```

Flere procedurer, som hører sammen, kaldes et *program*. Du har nu næsten skrevet et fuldstændigt program til styring af det dobbelte transportbånd. Vi mangler blot at få skrevet en *startprocedure*, som kan sætte det hele igang.

Øvelse 4.5

Tryk på <f1> og skriv denne procedure

```
PROC GO
bestil
tn(1)
giv(x,y)
afbryd
```

Tryk på <f3> for at afslutte proceduren. Kontroller, at proceduren er i orden, ved at bruge list-kommandoen.

Tryk på <f7> og giv kommandoen

```
GO
```

Bestil 2 klodser til højre og 3 til venstre.

```
/**/
```

Hermed er dit første projekt slut. Hvis der er sluttet en *printer* til din datamat, kan du få skrevet en *liste* over alle procedurerne på papir ved at bruge kommandoen

```
list "lp:
```

Hvis du har lyst, kan du forbedre programmet yderligere ved først at skrive denne procedure

```

PROC signal
  WHILE KEY$="" DO
    sl(1)
    alarm
    tn(1)
    vt(.2)
  ENDWHILE
ENDPROC signal

```

Tryk på <f7> og afprøv proceduren med følgende kommando-linje:

signal

Indsæt derpå denne kommando i stedet for *afvent*, sådan at den bliver brugt, hvis bånd nr. 2 har lagt en klods fra sig, inden en ny klods er klar på bånd nr. 1.

Her er lidt forklaring til proceduren *signal*:

WHILE KEY\$="" DO	<i>så længe (WHILE), der ikke er trykket på nogen taste (KEY\$=""), udfør (DO) følgende igen og igen:</i>
sl(1)	<i>sluk udgang nr. 1</i>
alarm	<i>giv en alarmtone</i>
tn(1)	<i>tænd udgang nr. 1</i>
vt(.2)	<i>vent i 0.2 sekunder</i>
ENDWHILE	<i>Hertil!</i>

NU FØLGER EN OVERSIGT OVER DE KOMMANDOER, DU HAR BRUGT I DETTE AFSNIT:

kort form	lang form	virker sådan
tn	tænd	<i>tænder en udgang</i>
sl	sluk	<i>slukker en udgang</i>
vt	vent	<i>venter et antal sekunder</i>
vpv	vent'på'vovse	<i>venter på en vagthund</i>
list	list	<i>giver en liste over en procedure</i>
renum	renum	<i>laver nye linjenumre</i>
afvent	afvent	<i>afventer et tryk på en taste</i>
page	page	<i>ny skærmside, dvs. skærmen ryddes</i>
afbryd	afbryd	<i>slukker alle udgange</i>
alarm	alarm	<i>giver en alarmtone</i>

Eksempler:

tn (1)	<i>udgang nr. 1 tændes</i>
sl (4)	<i>udgang nr. 4 slukkes</i>
vt (5)	<i>der ventes i 5 sekunder</i>
vpv (6)	<i>vent på den vovse, der er sluttet til indgang nr. 6</i>

FASTE COMAL-ORD (skrives med stort i programlister):

PROC	<i>indleder en procedure</i>
ENDPROC	<i>afslutter en procedure</i>
PRINT	<i>skriver en tekst eller et tal på skærmen</i>
INPUT	<i>henter en tekst eller et tal ind fra tastaturet</i>
AT	<i>bestemmer, hvor på skærmen der skal skrives</i>
PAGE	<i>rydder skærmen</i>
WHILE	<i>betyder "sålænge" og indleder et afsnit, som kan gentages under visse betingelser</i>
KEY\$	<i>betyder "taste". Hvis der trykkes på en taste, opfanger KEY\$ det tegn, der tastes</i>
DO	<i>afslutter en WHILE- og en FOR-sætning</i>
ENDWHILE	<i>afslutter et afsnit, som indledes med WHILE</i>

Pariserhjulet

I dette afsnit skal du bruge modellen, der er beskrevet i byggevejledningen 1090 A (*pariserhjulet*). Der skal dog foretages følgende ændring:

Den på *billede 11, side 10*, viste *føler* på toppen af modellen skal *ikke* bruges i det følgende.



5. Her er plads til alle

Øvelse 5.1

Find billedet af modellen på sidste side af byggevejledningen. Forbind motoren, der er mærket med *A*, og føleren, der er mærket med *6*, med styrekassen som vist på billedet.

Skriv så følgende kommando

```
tn(Ø)
```

Så skal hjulet dreje sig mod uret, når det ses fra den side, hvor motor og føler sidder. Hvis ikke det er tilfældet, skal stikket til motoren vendes.

Standstille hjulet ved at skrive

```
sl(Ø)
```

Skriv kommando-linjen:

```
tn(Ø); tlt(6,12); sl(Ø)
```

og læg mærke til hjulets bevægelse. Hvis du helst vil bruge de lange kommandoer, kan du også skrive sådan

```
tænd(Ø); tæltil(6,12); sl(Ø)
```

Brug kommando-linjen igen og læg denne gang mærke til den *grønne lampe ved indgang nr. 6*.

Tryk på <f1> og skriv følgende procedure:

```
proc frem(x)
tn(Ø)
tlt(6,x)
sl(Ø)
```

Tryk på <f3> for at afslutte proceduren. Tryk derpå <f7> og skriv

```
list frem
```

for at kontrollere, at proceduren er skrevet rigtigt. Skriv kommandoen

```
frem(6)
```

og bemærk hjulets bevægelse. Skriv derpå kommandoen

frem(10)

og tæl, hvor mange gange den grønne lampe blinker.

Se efter, om den grønne lampe er tændt eller slukket og skriv kommandoen

frem(1)

Læg igen mærke til, om den grønne lampe nu er tændt eller slukket. Brug kommandoen nogle gange og læg hver gang mærke til den grønne lampe.

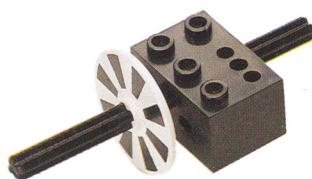
Brug kommandoen *frem(1)*, indtil *en af gondolerne er nede, og den grønne lampe er slukket*. Fra nu af skal det være det sted, hvor passagererne står af eller på.

/**/

For at forstå proceduren *frem* helt, skal vi først se lidt nærmere på *tælle-hjulet* og *føleren*.

Øvelse 5.2

Lav følgende lille opstilling:



Den side af tællehjulet med de *store sektorer* skal vende ind mod følerens *lys-hul*. Forbind føleren med *indgang nr. 7* på styrekassen. Drej hjulet langsomt rundt og læg mærke til den grønne lampe ud for nr. 7. Prøv at stille hjulet, så lampen lyser. Hvilken farve har det felt, som står ud for lys-hullet i føleren? Stil hjulet, så den grønne lampe slukker. Hvilken farve har nu det felt, som står ud for lys-hullet?

/**/

Du kan se, at der gælder følgende: *Hver gang, den grønne lampe tændes eller slukkes, dvs. hver gang, der skiftes fra hvidt til sort eller fra sort til hvidt ud for lys-hullet, sendes en tælle-impuls til datamaten.*

Nu er det let nok at forstå, hvordan proceduren *frem* virker:

<code>frem(10)</code>	<i>det er din kommando</i>
<code>PROC frem(x)</code>	<i>x får værdien 10</i>
<code>tn(0)</code>	<i>og motoren starter.</i>
<code>tl1(6,x)</code>	<i>Så skal der tælles til x,</i>
	<i>dvs. 10, på indgang nr. 6</i>
<code>sl(0)</code>	<i>og motoren kan standses.</i>
<code>ENDPROC frem</code>	<i>Sådan!</i>

Opgave 5

Skriv en procedure, som kan få hjulet til at dreje i modsat retning. Kald proceduren *bak*.

Afprøv proceduren med kommandoer, som fx.:

```
bak(6)  
  
/**/
```

Husk at gemme de procedurer, du har skrevet, hvis du holder pause nu. Kan du huske hvordan? Ellers læs i afsnittet: *Om at gemme og hente programmer.*

6. Ikke så hastigt i starten

I dette kapitel skal du bruge de procedurer, du skrev i kapitel 5.

Øvelse 6.1

Tryk <f7> og skriv kommandoerne:

```
styrke(0,4)
frem(12)
```

Læg mærke til, hvordan hjulet bevæger sig. Giv derpå kommandoerne:

```
styrke(0,8)
frem(12)
```

Læg igen mærke til hjulets bevægelse.

/**/

Ved hjælp af kommandoen *styrke* kan man stille en udgangs styrke til forskellige værdier. *Laveste styrke er 0, og højeste styrke er 8.* Kommandoen:

```
styrke(0,4)
```

bevirker, at *udgang nr. 0* bliver indstillet til *styrke 4* (ca. halv styrke).

Øvelse 6.2

Tryk på <f1> og skriv denne procedure:

```
proc betal
page
print at 5,5: "25 øre, tak!"
afvent
```

Afslut proceduren ved at trykke på <f3>. Skriv derpå denne procedure:

```
proc nye
styrke(0,4)
betal
for i=1 to 3
frem(6)
betal
endfor
endproc
```

(Fra nu af må du selv huske at trykke på <f1> i begyndelsen og <f3> til slut).

Tryk på <f7> og få skrevet en liste over de to procedurer. Sammenlign listerne med disse

```
PROC betal
  PAGE
  PRINT AT 5,5: "25 øre, tak!"
  afvent
ENDPROC betal

PROC nye
  styrke(0,4)
  betal
  FOR i:=1 TO 3 DO
    frem(6)
    betal
  ENDFOR i
ENDPROC nye
```

Brug kommandoen *frem* for at få stillet en af gondolerne, så den er klar til at modtage en gæst (se slutningen af *øvelse 5.1*).

Skriv så kommandoen

```
nye
```

Tryk på en taste som tegn på, at gæsten har betalt. Se efter, at du har mulighed for at få en gæst i hver gondol.

```
/**/
```

Når alle fire gondoler er besat, kan den egentlige tur starte. Det kan naturligvis ikke nytte at starte for brat. Du skal prøve, om du kan få hjulet til at gå op i hastighed ved at bruge *styrke 4, 5, 6, 7 og 8*. Mellem hver styrke-regulering skal du lade systemet tælle til 6.

Øvelse 6.3

Skriv denne procedure

```
PROC start
  PAGE
  PRINT AT 5,8: "så kører vi!"
  tænd(Ø)
  FOR s:=4 TO 8 DO
    styrke(Ø,s)
    tlt(6,6)
  ENDFOR s
ENDPROC start
```

Tryk på <f7> og skriv denne række af kommandoer:

```
nye; start; tlt(6,48); sl(Ø)
```

Sæt kommandoen *tlt(6,48)* - indgang nr. 6 tager tælling til 48 - ind i en procedure, som du kalder *turen*.

Tryk på <f7>, når du har skrevet den og giv disse kommandoer:

```
nye; start; turen; sl(Ø)
```

```
/**/
```

Nu gælder det blot om at få standset hjulet på en pæn måde og ikke bare slå bremserne i, som det skete i *øvelse 6.3*. Du skulle jo nødtigt have dine kunder til at falde ud af gondolerne.

Øvelse 6.4

Skriv denne procedure:

```
PROC stands
  FOR s:=8 TO 4 STEP -1 DO
    styrke(Ø,s)
    tlt(6,6)
  ENDFOR s
  sluk(Ø)
ENDPROC stands
```

Tryk på <f7> og skriv kommando-linjen:

```
nye; start; turen; stands
```

Så skulle det hele helst gå ordentligt til.

```
/**/
```

En linje som denne

```
FOR s:=8 TO 4 STEP -1 DO
```

får COMAL til at tælle *baglæns fra 8 til 4*, altså sådan: 8, 7, 6, 5, 4. Linjen lige efter

```
styrke(Ø,s)
```

bevirker altså, at styrken først stilles til 8 - fuld styrke - derpå til 7, så 6, så 5 og til slut 4. Hjulet kører stille og roligt ned i fart.

Opgave 6

Skriv proceduren *go* så dens krop indeholder de fire kommandoer fra slutningen af *øvelse 6.3*. Skriv de fire kommandoer på hver sin linje.

Gør systemet klar og afprøv hele programmet ved at give kommandoen

```
go
```

```
/**/
```

Nu har du fået styr på pariserhjulet. Hvis du har lyst at byde dine kunder lidt mere fornøjelse, kan du prøve at lade turen bestå af både en forlæns tur, som nu, og derpå en baglæns tur. Du har jo allerede skrevet proceduren *bak*, som kan bruges her. Men pas på! Der skal også startes og standses pænt, når hjulet kører baglæns. Måske var det en ide at skrive nogle procedurer, som kunne hedde *start'bak* og *stands'bak*. Husk, at *udgang 0* tændes, når hjulet drejer fremad, men *udgang 1* skal tændes, hvis hjulet skal gå baglæns.

I dette afsnit har du brugt to nye styrekommandoer:

kort form	lang form	virker sådan
tlt styrke	tælt til styrke	<i>Afventer et antal tællinger Stiller en udgangs styrke til et tal mellem 0 og 8.</i>

Eksempel

tlt(6,12)

*Venter på, at indgang nr. 6
tager tælling til 12.*

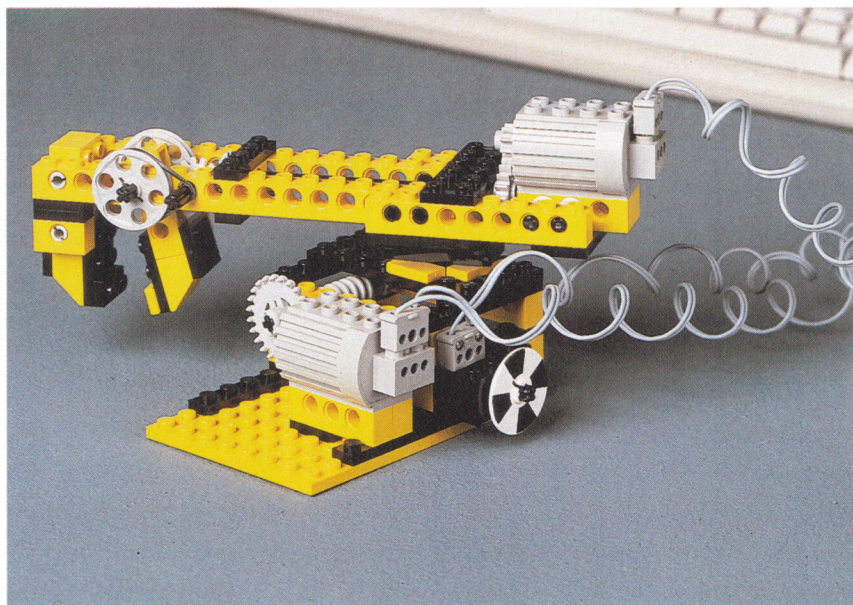
styrke(2,4)

*Indstiller udgang nr. 2 til
en styrke på 4.*

Robotarm og transportbånd

I dette afsnit skal du bruge modellen, som er beskrevet i byggevejledning nr. 1090 E (*robotarmen*). Der skal dog foretages følgende ændring:

På byggevejledningens *sidste side* ses det, at tællehjulet vender siden med 16 sektorer bort fra føleren. Det skal være lige omvendt. Siden med de 16 sektorer skal altså vende ind mod føleren. Motorer og føler forbindes med styrekassen som vist i vejledningen.



7. Man må have greb om tingene - og kunne sno sig

Øvelse 7.1

Skriv denne kommando-linje:

```
tn(Ø); vt(1/3); sl(Ø)
```

Den skal få kloen til at lukke sig. Hvis den ikke gør det, skal stikket til motoren ovenpå armen vendes.

Få kloen til at åbne sig igen ved at skrive

```
tn(1); vt(1/3); sl(1)
```

Skriv derpå denne procedure

```
PROC grib  
  tn(Ø)  
  vt(1/5)  
  sl(Ø)  
ENDPROC grib
```

Tryk på <f7> og giv kommandoen

```
grib
```

for at få kloen til at lukke sig.

```
/**/
```

Opgave 7

Skriv en procedure, som kan få kloen til at åbne sig. Kald proceduren *slip*.

Når den er i orden, så prøv med denne kommando-linje

```
for i=1 to 10 grib; slip
```

Så skulle du gerne få det store bifald fra robotten.

```
/**/
```

Nu kan robotten gribe og slippe. Så skal du have den til at dreje.

Øvelse 7.2

Skriv kommando-linjen

```
tn(2); tlt(7,20); sl(2)
```

Nu skal armen dreje sig med uret, når man ser den ovenfra. Hvis den ikke gør det, skal stikket i drejeladets motor vendes.

Skriv denne procedure

```
PROC frem(x)
  tn(2)
  tlt(7,x)
  sl(2)
ENDPROC frem
```

Gør systemet klar (dvs. tryk på <f7>) og giv kommandoen

```
frem(40)
```

Prøv evt. med et par andre talværdier i *frem*, men pas på, at ledningerne til motoren ovenpå armen ikke bliver snoet for meget rundt.

/**/

Opgave 8

Skriv en procedure af samme slags som *frem*, men sådan at den får armen til at dreje mod uret. Kald proceduren *bak*. Begynd sådan

```
proc bak(x)
```

Gør systemet klar og giv kommandoen

```
bak(50)
```

Prøv også med fx. denne kommando-linje

```
frem(50); vt(1); bak(50)
```

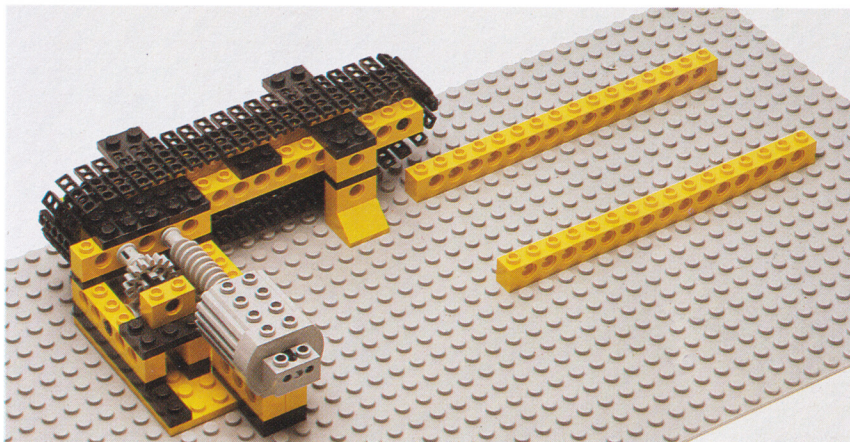
Prøv selv at finde ud af, hvor mange tællinger, der skal til, for at få armen til at dreje en halv omgang. En kvart omgang. En hel omgang (men pas på ledningerne!).

/**/

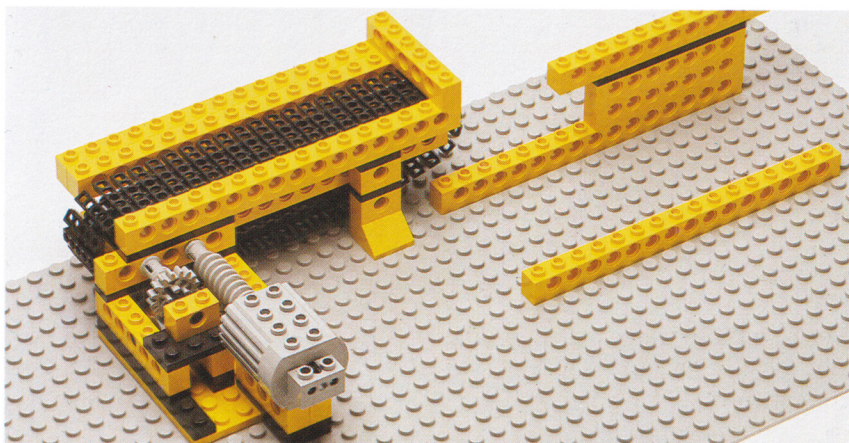
8. Madsens maleservice

Nu skal du igang med det afsluttende projekt. Til det skal du bruge endnu en kasse 1090. Derfor skal du muligvis slå dig sammen med en anden for at klare det. Endvidere skal du anvende *en stor LEGO byggeplade (art. nr. 1085)*.

Først skal du bygge det ene af de bånd, du startede med i kapitel 1. Brug byggevejledning nr. 1090 D. Følg vejledningens *billede 1-6*, men uden at sætte de på *billede 4* viste klodser øverst på båndet. Sæt det næsten færdige bånd på byggepladen, som vist her:

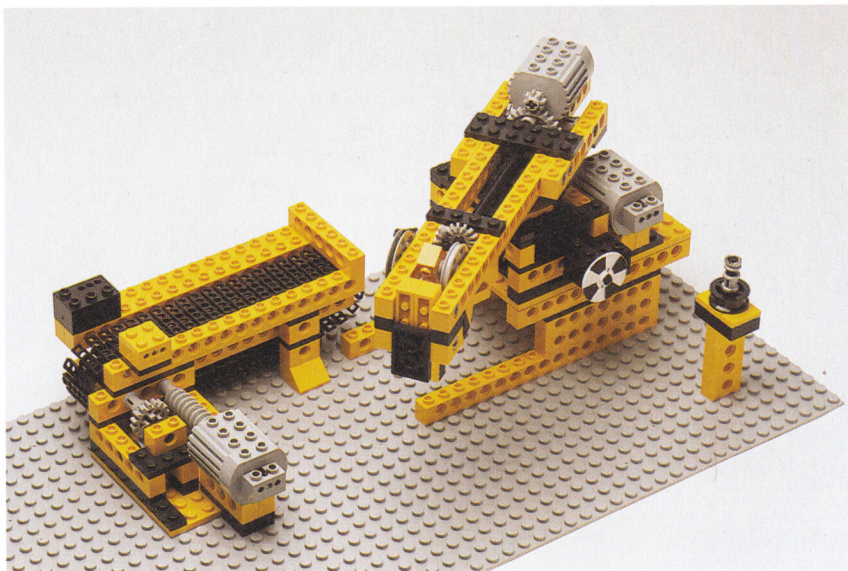


Sæt også to *1x16 stænger* på byggepladen som vist på billedet. Sæt 3 *stk. 1x16 klodser* på siderne af båndet og 1 *stk. 1x6* på tværs, som vist her

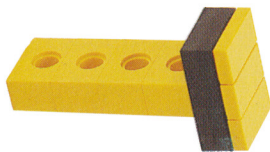


Byg videre ovenpå de to lange stænger ved siden af båndet.

Sæt til slut en *føler* og en *lampe* på båndet og sæt robotarmen op på det færdige fundament, så du får den færdige model:



Den lille *konsol* yderst til højre på billedet skal bygges sådan:



Til slut skal du bygge nogle "*dimser*" på denne måde:



En "*dim*s" skal anbringes på konsollen, som vist på fig. 12.

Slut båndets motor til *udgang nr. 4* og lampen til *udgangen over afbryderen* på styrekassen. Slut føleren til *indgang nr. 6*.

Øvelse 8.1

Giv kommandoen

```
tn (4)
```

og kontroller, at båndet kører fremad. Hvis ikke det er tilfældet, skal stikket i motoren vendes. Sluk for udgangen med kommandoen

```
sl (4)
```

Sæt en finger ind mellem lampen og føleren og se efter, om vovsen giver sig til at "gø", som den skal.

/**/

Afsluttende projekt

Skriv en procedure, som udfører følgende:

```
starter båndet
venter på vovsen
standser båndet
venter i 5 sekunder
starter båndet
venter i 1 sekund
standser båndet
```

Kald proceduren for *maling*.

Afprøv den ved at sætte en "dims" på båndet og starte det med kommandoen

```
maling
```

Når dimsen kører hen over båndet, bliver den malet. Når den står og venter i 5 sekunder, bliver den tørret. Til slut bliver den kastet ud af båndet.

Skulle du være i tvivl om fremgangsmåden, kan du kigge i kapitel 1. Der lavede du noget lignende.

Sæt en "dims" på konsollen ud for robotarmen. Sørg for at robotens klo står åben. Få armen til at dreje, så "dimsen" står midt i robotens klo. Hvis tingene ikke rigtigt kan passe sammen, har du rimeligvis anbragt konsollen forkert.

Få robotten til at gribe "dimsen" og derpå dreje den hen over båndet. Du skal sandsynligvis bruge kommandoen

frem(170)

hvis du har fulgt byggevejledningen nøjagtigt.

Lad robotten slippe "dimsen" og derpå dreje et stykke tilbage. Brug fx. *bak(20)*.

Til slut skal "dimsen" males og robotarmen dreje helt tilbage for at hente en ny "dime".

Når du har fået det hele til at virke, så skriv en procedure, der kan udføre alle de nødvendige kommandoer. Kald den fx. *go*.

*/**/*

Meget mere styr på robotterne

I dette afsnit skal du bruge *det dobbelte transportbånd og den en-armede robot med transportbånd*, som du kender fra de tidligere afsnit.

De to modeller bliver imidlertid styret på en noget anden måde, som mere ligner den, der bruges ved programmering af industrirobotter.

Det kræver til gengæld mere COMAL-programmering, end du hidtil har brugt. Det er bedst, hvis du kender en del til COMAL i forvejen. Hvis ikke du gør det, må du være forberedt på, at det kan tage lidt tid at finde ud af de procedurer, der bliver brugt. Du kan læse om COMAL i fx. "*COMAL-80 på 64'eren*" (SYSTIME 1984).

9. Om fordelene ved at have et øje på hver finger

Opgave 9

Slut det dobbelte transportbånd til styrekassen på samme måde, som beskrevet i kapitel 1-4. Hent styreprogrammet ind og prøv at lade robotten køre efter det program, du skrev i kapitel 1-4.

/**/

Læg mærke til, at bånd nr. 1 ikke begynder at køre, før bånd nr. 2 har kastet sin last ud, selvom du har lagt en ny klods til rette og trykket på en taste. Det er jo i grunden spild af tid! Bånd nr. 1 kunne lige så godt begynde af køre fremad med den nye klods, mens bånd nr. 2 gør sig færdig.

Det er let at se, hvorfor det ikke kan lade sig gøre, når vi programmerer, som vi hidtil har gjort. Lad os fx. se nærmere på proceduren *venstre*:

```
PROC venstre
    tænd(3); vent(5); sluk(3)
ENDPROC
```

Den bliver udført på følgende måde: først *startes motoren* med *tænd(3)*; derpå *venter* datamaten i 5 *sekunder* med *vent(5)*, før den *standser motoren* med *sluk(3)*. Når *vent(5)* virker, kan datamaten ikke lave andet end at vente. På data-engelsk siger man, at "the computer is busy waiting", dvs. "datamaten har travlt med at vente".

Det betyder fx., at den *ikke samtidigt* kan høre efter *vovsen*, der skal holde øje med, om en ny klods er kommet frem med bånd nr. 1. Det er altså sådan, at hvis vi kører med bånd nr. 1 samtidigt med bånd nr. 2, kunne vi risikere, at en klods slipper gennem hundehuset og falder ned på bånd nr. 2, uden at det bliver opdaget, fordi datamaten har travlt med at vente og derfor ikke hører, at hunden giver hals.

Hvis vi skal løse det problem, må vi programmere sådan, at datamaten både kan holde øje med uret, for at se om de 5 sekunder er gået, og høre efter hunden. Det ville nemlig være meget uheldigt, hvis en klods falder ned på bånd nr. 2, mens det fx. kører til højre, fordi den næste klods måske skal til venstre.

Øvelse 9.1

Sæt en lyssten i forbindelse med *udgang nr. 0* på styrekassen. Skriv derpå denne procedure

```
PROC blink
  tænd(Ø)
  vent(.7)
  sluk(Ø)
  vent(.5)
ENDPROC
```

Afprøv proceduren med fx. denne kommando

```
for i=1 to 5 do blink
```

Slet proceduren med kommandoen

```
del blink
```

og skriv i stedet denne

```
PROC blink
  CASE p'bl OF
    WHEN 1
      IF udløbet(1) THEN
        tænd(Ø)
        stilur(1,.7)
        p'bl:=2
      ENDIF
    WHEN 2
      IF udløbet(1) THEN
        sluk(Ø)
        stilur(1,.5)
        p'bl:=1
      ENDIF
  ENDCASE
ENDPROC blink
```


Afprøv den med disse kommandoer

```
p'bl=1
while true do blink
```

Tryk på <RUN/STOP>, når du vil standse blinklyset. Hvis du har standset processen, mens lyset er tændt, kan du slukke det med kommandoen

```
sluk(Ø)
```

```
/**/
```

Her er lidt forklaring til den store blink-procedure

```
PROC blink
CASE p'bl OF
WHEN 1
    IF udløbet(1) THEN
        tænd(Ø)
        stilur(1,.7)
        p'bl:=2
    ENDIF
WHEN 2
    IF udløbet(1) THEN
        sluk(Ø)
        stilur(1,.5)
        p'bl:=1
    ENDIF
ENDCASE
ENDPROC blink
```

når p'bl har værdien 1:
hvis ur nr. 1 er udløbet:
tænd lampen
stil ur nr. 1 til 7/10 sek.
gør klar til at slukke

når p'bl har værdien 2:
hvis ur nr. 1 er udløbet:
sluk lampen
stil ur nr. 1 til 5/10 sek.
gør klar til at tænde

ikke flere jobs i denne proc.

Når de to kommandoer

```
p'bl=1
while true do blink
```

udføres, bliver proceduren *blink* kaldt igen og igen med få brøkdeler sekunders mellemrum. Første gang, proceduren udføres, er *p'bl* sat til 1, så den første af WHEN-blokkene udføres. Fra starten er *alle ure sat til 0*, så betingelsen *udløbet(1)* er opfyldt. Derfor tændes lampen, uret indstilles til at gå i 7/10 sekund og *pegepinden p'bl* sættes til 2. Dermed er proceduren udført for første gang. Når proceduren et øjeblik senere kaldes igen, udføres den anden WHEN-blok, netop fordi vi har sat *p'bl* til 2.

Uret var indstillet til at gå i $7/10$ sekund, men så lang tid er der slet ikke gået. Derfor er betingelsen *udløbet(1)* ikke opfyldt, så de tre sætninger efter IF-sætningen udføres ikke. Altså har *p'bl* stadig værdien 2, og lampen lyser stadigvæk.

Næste gang, proceduren kaldes, møder vi altså igen den anden WHEN-blok, og det bliver vi ved med, indtil der er gået $7/10$ sekund. Når det er sket, bliver der lukket op til de tre linjer efter IF-sætningen med det resultat, at lampen slukkes, uret stilles til $5/10$ sekund og *p'bl* sættes til 1.

Nu er vi tilbage i den første WHEN-blok, men vi kommer ikke ind bag IF-sætningen, før der er gået et halvt sekund. Lampen bliver altså ikke tændt igen, før det halve sekund er gået.

Og sådan bliver det ved, indtil du trykker på <RUN/STOP> tasten.

Lidt ekstra. Hvis du gerne vil vide, hvor mange gange COMAL kan nå at kigge på uret, mens lampen er tændt, altså mens $7/10$ sekund går, kan du indsætte et par testlinjer på følgende steder i proceduren *blink*:

```
...  
WHEN 1  
  if t>0 then stop  
...  
WHEN 2  
  t:=+1  
...  
  
```

indsættes

indsættes

Skriv derpå denne procedure

```
PROC test  
  p'bl:=1; t:=0  
  LOOP  
    blink  
  ENDLOOP  
ENDPROC test
```

Tryk som sædvanlig på <f7> og skriv kommandoerne

```
test  
print t
```

Øvelse 9.2

Skriv denne procedure

```
PROC go
  p'bl:=1
  LOOP
    blink
  ENDLOOP
ENDPROC
```

Afprøv den.

/**/

De tre linjer

```
LOOP
  blink
ENDLOOP
```

kan forstås sådan: "bliv ved med at kalde proceduren *blink* i det uendelige", hvilket i praksis vil sige, indtil du trykker på <RUN/STOP> tasten.

Ideen med at kalde procedurer med meget korte tidsmellemlum bliver naturligvis først interessant, hvis der er tale om flere procedurer. Så vil det være som om, man udfører dem *næsten samtidigt*.

Øvelse 9.3

Slut en *føler* til *indgang nr. 7*. Tag føleren i hånden, så der ikke falder lys direkte ind i den. Hold en finger for lyshullet, og iagttag den *grønne kontrollampe* ved *indgang nr. 7*. Løft fingeren og læg atter mærke til kontrol-lampen. Skriv denne kommando

```
nulstil(7)
```

Hold fingeren for lyshullet et kort øjeblik og skriv derpå

```
print tæller(7)
```

Den første kommando *nulstiller den tæller*, der er knyttet til *indgang nr. 7*. Funktionen *tæller(7)* giver det antal tællinger, der er registreret, siden sidste nulstilling. Det lille eksperiment ovenfor skulle gerne give resultatet 2, idet der tælles hver gang, lampen *tændes eller slukkes*. Hvis du holder

føleren rigtigt, slukkes kontrollampen, når du holder fingeren for lyshullet, og tænder igen, når fingeren fjernes. Holder du føleren op mod en kraftig lyskilde, reagerer den lige omvendt (prøv).

Skriv følgende procedure

```
PROC skift
  CASE p'sk OF
    WHEN 1
      IF tæller(7)>1 THEN
        blinker:=TRUE
        nulstil(7)
        p'sk:=2
      ENDIF
    WHEN 2
      IF tæller(7)>1 THEN
        blinker:=FALSE
        sluk(Ø)
        nulstil(7)
        p'sk:=1
      ENDIF
  ENDCASE
ENDPROC skift
```

og udvid proceduren go til følgende

```
PROC go
  p'b1:=1; p'sk:=1
  blinker:=TRUE
  LOOP
    skift
    IF blinker THEN blink
  ENDLOOP
ENDPROC go
```

Start hele programmet med kommandoen

```
go
```

Hold en finger hen foran følerens lyshul, og læg mærke til, hvad der sker. Prøv igen. Og igen...

```
/**/
```

Her er først lidt forklaring til proceduren *skift*

```
PROC skift
CASE p'sk OF
WHEN 1
    IF tæller(7)>1 THEN      hvis der er talt til mindst 2:
        blinker:=TRUE        der skal blinkes
        nulstil(7)           nulstil tæller 7
        p'sk:=2              skift til 2. tilfælde
    ENDIF
WHEN 2
    IF tæller(7)>1 THEN      hvis der er talt til mindst 2:
        blinker:=FALSE       der skal ikke blinkes
        sluk(Ø)              sluk lampen
        nulstil(7)           nulstil tæller 7
        p'sk:=1              skift til 1. tilfælde
    ENDIF
ENDCASE
ENDPROC skift
```

I proceduren *go* har du bl.a. skrevet disse fire linjer

```
LOOP
    skift
    IF blinker THEN blink
ENDLOOP
```

Som du ser, kan programmet nu holde øje med både *blink* og *skift*. Ikke helt samtidigt, men så hurtigt efter hinanden, at hverken ure eller tællere når at "løbe ud", uden at det bliver opdaget i tide.

Størrelsen *blinker* kaldes et *flag*. Det kan enten være *hejst* (TRUE) eller *taget ned* (FALSE). Sætningen *IF blinker THEN blink* sørger for, at *blink* kun bliver udført, *hvis flaget er hejst* (TRUE). I dit program er det *skift*, der passer flaget, og den foretager sig ikke meget andet. For en sikkerheds skyld er det dog lavet sådan, at lampen bliver slukket, når der ikke skal blinkes. Det sørger kommandoen *sluk(0)* for.

Gem programmet på disketten og ryd arbejdsområdet med kommandoen

```
del-
```

Øvelse 9.4

I denne øvelse skal du bruge det dobbelte transportbånd, som blev bygget og anvendt i kapitel 1-4. Motorer, lamper og føler skal forbindes med styrekassen som vist tidligere.

Skriv følgende procedure og funktion

```
PROC frem
CASE p'fr OF
WHEN 1
    IF objekt'klar THEN
        tænd(0); sluk(1)
        p'fr:=2
    ENDIF
WHEN 2
    IF vovse(6) THEN
        sluk(0); tænd(1)
        p'fr:=3
    ENDIF
WHEN 3
    IF NOT bånd'2'kører THEN
        tænd(0); sluk(1)
        vent(1)
        sluk(0); tænd(1)
        objekt'fri:=TRUE
        p'fr:=1
    ENDIF
ENDCASE
ENDPROC frem
FUNC objekt'klar
    RETURN KEY$<>""
ENDFUNC objekt'klar
```

Afprøv programmet med følgende kommandoer

```
bånd'2'kører=false
p'fr=1
while true do frem
```

Sæt en klods på bånd nr. 1 og tryk på mellemrumstasten.

Tryk på <RUN/STOP> og brug kommandoerne ovenfor igen, idet dog *false* i første linje ændres til *true*. Sæt igen en klods på bånd nr. 1 og tryk på mellemrumstasten.

/**/

Opgave 10

Forklar virkningen af de to sæt kommandoer ved at gå proceduren *frem* igennem linje for linje. Lad evt. COMAL skrive en liste over *frem* på printeren ved at bruge kommandoen

```
list frem "lp:
```

Skriv på denne liste dine egne kommentarer ud for hver enkelt linje. Brug fx. samme stil som er anvendt ovenfor i forklaringen til proceduren *blink*.

/**/

Øvelse 9.5

Skriv denne procedure

```
PROC højre
CASE p'hj OF
WHEN 1
    IF objekt'fri THEN
        tænd(2)
        stilur(2,4)
        bånd'2'kører:=TRUE
        objekt'fri:=FALSE
        p'hj:=2
    ENDIF
WHEN 2
    IF udløbet(2) THEN
        vent(1); sluk(2)
        tæl'hj:=+1
        bånd'2'kører:=FALSE
        p'hj:=1
    ENDIF
ENDCASE
ENDPROC højre
```

Skriv også denne test-procedure

```
PROC test
  p'fr:=1; p'hj:=1
  objekt'fri:=FALSE
  bånd'2'kører:=FALSE
  LOOP
    frem
    højre
  ENDLOOP
ENDPROC test
```

Afprøv programmet ved at skrive kommandoen *test*, lægge en klods på bånd nr. 1 og trykke på mellemrumstasten.

Læg derpå tre klodser på bånd nr. 1 med ca. een centimeters mellemrum og tryk tre gange på mellemrumstasten.

Tryk på <RUN/STOP>, når du er færdig med afprøvningen.

/**/

Opgave 11

Forklar forløbet af afprøvningen i *øvelse 9.5*.

Skriv derpå en procedure, som får bånd nr. 2 til at køre til venstre med klodsen. Kald proceduren *venstre*. Lav om på *test*, så den kan bruges til afprøvning af *frem* sammen med *venstre*.

Gennemfør en afprøvning af *frem* og *venstre* svarende til den, der blev lavet med *frem* og *højre* i *øvelse 9.5*.

/**/

Øvelse 9.6

Slet proceduren *test* med

```
del test
```

og skriv denne procedure

```
PROC giv'hj(x)
  tæl'hj:=0
  p'hj:=1
  WHILE tæl'hj<x DO
    frem
    højre
  ENDWHILE
ENDPROC giv'hj
```

Skriv en ny *test*-procedure med følgende indhold

```
PROC test
  p'fr:=1
  bånd'2'kører:=FALSE
  objekt'fri:=FALSE
  giv'hj(2)
ENDPROC test
```

Afprøv programmet ved at give kommandoen *test*; læg to klodser på bånd nr. 1, således at den ene ligger ved begyndelsen af båndet og den anden næsten helt henne ved hundehuset, og tryk to gange på mellemrumstasten. Når testen er færdig, så prøv at få værdien af *tæl'hj* at vide ved at bruge kommandoen

```
print tæl'hj
```

```
/**/
```

Opgave 12

Skriv en procedure som *giv'hj*, men sådan at den lægger klodser til venstre i stedet for til højre. Kald proceduren *giv'vn*. Lav om på *test*, så den kan bruges til afprøvning af *giv'vn*.

Foretag afprøvning.

/**/

Nu kan vi afslutte vort projekt med

Øvelse 9.7

Skriv disse procedurer

```
PROC bestil
  PAGE
  INPUT AT 5,5: "til højre? ": x
  INPUT AT 7,5: "til venst? ": y
  PAGE
  PRINT AT 5,5: "klar til at levere"
  PRINT AT 0,5: x;"til højre, og"
  PRINT AT 0,5: y;"til venstre."
  PRINT AT 0,5: "ialt: ";x+y
ENDPROC bestil
```

```
PROC go
  bestil
  p'fr:=1
  objekt'fri:=FALSE
  bånd'2'kører:=FALSE
  giv'hj(x)
  giv'vn(y)
  afbryd
ENDPROC go
```

Afprøv ved at skrive *go* og bestille et antal klodser til højre og venstre.

/**/

Ekstraopgave

Hvis du kan låne *to ekstra lamper* og *en føler* fra et andet sæt 1090, kan du prøve at udbygge det dobbelte transportbånd ved at bygge et hundehus på hver ende af bånd nr. 2 og forbinde *begge følere* med *indgang nr. 7* (det ene LEGO stik skal sættes ind i siden af det andet).

Derpå skal du ændre i procedurerne *højre* og *venstre*, så det ikke længere er et *ur*, der bestemmer, hvornår båndet skal standse, men derimod *vovse (7)* - som forresten er et par tvillinger, der sidder og gør ad hinanden.

Husk også, at lade bånd nr. 2 køre i ca. *1 sekund* efter at hundene har givet hals, så klodsen kan komme helt ud af båndet.

/**/

10. To robotter laver en aftale

Øvelse 10.1

I denne øvelse skal du bruge robotarmen og transportbåndet, som blev beskrevet i *kapitel 8-9*. De to robotter skal forbindes med styrekassen, som beskrevet tidligere.

Skriv dette program

```
PROC robotarm
CASE p'ro OF
WHEN 1
  IF objekt'klar THEN
    grib
    p'ro:=2
  ENDIF
WHEN 2
  nulstil(7)
  tænd(2)
  p'ro:=3
WHEN 3
  IF tæller(7)=170 THEN
    sluk(2)
    p'ro:=4
  ENDIF
WHEN 4
  IF bånd'klar THEN
    slip
    nulstil(7)
    tænd(3)
    p'ro:=5
  ENDIF
WHEN 5
  IF tæller(7)=10 THEN
    objekt'fri:=TRUE
    nulstil(7)
    p'ro:=6
  ENDIF
```

```

    WHEN 6
        IF tæller(7)=160 THEN
            sluk(3)
            p'ro:=1
        ENDIF
    ENDCASE
ENDPROC robotarm

PROC grib
    tænd(0)
    vent(.25)
    sluk(0)
    objekt'i'klo:=TRUE
ENDPROC grib

```

Sørg for, at robotarmen står over konsollen, og at der står en dims på den. Afprøv programmet med kommandoerne

```

bånd'klar=true
p'ro=1
while true do robotarm

```

Tryk på mellemrumstasten. Hvis programmet er skrevet rigtigt, skal robotarmen gribe dimsen, aflevere den til båndet og dreje tilbage over konsollen igen.

Brug kommandoerne igen, men ret *true* i første linje til *false*. Nu skal robotarmen gribe dimsen, køre den hen over båndet og standse uden at slippe dimsen. Grunden hertil er naturligvis, at *bånd'klar* er sat til *false*. Robotarmen kan altså ikke vide andet, end at båndet *ikke er klar* til at modtage dimsen.

Få robotarmen til at vende tilbage ved at bruge kommandoerne

```

slip
tænd(3); tæltil(7,170); sluk(3)

```

/**/

Opgave 13

Gå proceduren robotarm igennem linje for linje og forklar, hvordan den virker. Forklar betydningen af de forskellige flag.

/**/

Ved hjælp af flaget *objekt'fri* melder robotarmen, at en dims nu er klar til at blive malet. Denne melding skal båndet bruge, for at vide, hvornår det skal starte. Omvendt må robotarmen have at vide, om båndet er klar til at modtage et ny dims. Vi vil forlange, at *båndet skal stå stille, for at armen må slippe dimsen løs*. Flaget *bånd'klar* skal altså være *taget ned*, når båndet *kører*, og *hejst*, når båndet *er standset*.

Øvelse 10.2

Afslut programmet med følgende procedurer

```
PROC maling
  CASE p'ma OF
    WHEN 1
      IF objekt'fri THEN
        tænd(4)
        bånd'klar:=FALSE
        objekt'fri:=FALSE
        p'ma:=2
      ENDIF
    WHEN 2
      IF vovse(6) THEN
        sluk(4)
        stilur(1,5)
        bånd'klar:=TRUE
        p'ma:=3
      ENDIF
    WHEN 3
      IF udløbet(1) THEN
        tænd(4)
        stilur(1,1)
        bånd'klar:=FALSE
        p'ma:=4
      ENDIF
```

```

WHEN 4
  IF udløbet(1) THEN
    sluk(4)
    bånd'klar:=TRUE
    p'ma:=1
  ENDIF
ENDCASE
ENDPROC maling

```

Afprøv det.

/**/

Opgave 14

Sæt en lampe ovenpå den motor, der driver robotarmens klo, og slut den til udgang nr. 5.

Udvid programmet, så denne lampe blinker, hver gang robotarmen kører.

Vink: Brug fx. proceduren *blink*, som blev skrevet i *kapitel 9*. Indfør et flag med navnet *arm'kører*, og lad det bestemme, om der skal blinkes eller ej. Pas også på, hvilke ure, der anvendes. To processer, der kører næsten samtidigt, må normalt ikke styres af det samme ur. Du har *to ure* til din rådighed, nr. 1 og nr. 2. Og robotarmen bruger nr. 1!!

/**/

NU FØLGER EN OVERSIGT OVER DE NYE KOMMANDOER, DER HAR VÆRET BRUGT I DETTE AFSNIT:

kort form	lang form	virker sådan
stu	stilur	<i>stiller et ur til et antal sekunder</i>
udl	udløbet	<i>viser, om et ur er udløbet</i>
nst	nulstil	<i>nulstiller en tæller</i>
tlr	tæller	<i>giver en tællers stand</i>

Eksempler:

stilur(1,10)	<i>ur nr. 1 stilles til at gå i 10 sekunder</i>
udløbet(2)	<i>antager værdien TRUE, når ur nr. 2 er udløbet</i>
nulstil(7)	<i>nulstiller tælleren på indgang nr. 7</i>
tæller(7)	<i>giver antal tællinger siden sidste nulstilling</i>

Appendix

Om at gemme og hente programmer

Du har allerede brugt de tre funktionstaster <f1>, <f3> og <f7>. Når du skal *gemme* et program på disketten eller *hente* et fra disketten, får du brug for funktionerne *f2*, *f4* og *f6*. Du kan se disse betegnelser på forsiden af *funktionstasterne* (se efter!).

Når du skal *gemme* et program, skal du først trykke på <f6>, dvs. <SHIFT> + <f5>. Du skal altså holde <SHIFT> tasten trykket ned og samtidigt trykke på <f5>. Hvis du gør det, skriver datamaten denne tekst på skærmen

```
Programnavn: max 12. tegn.  
Indtastes et allerede eksisterende navn,  
SLETTES programmet på disketten.
```

Programnavn:

Markøren står og venter på, at et *programnavn* bliver tastet ind. Som du ser, må dette navn *ikke være på over 12 tegn*. Her er nogle eksempler på lovlige navne:

```
birger/Ø3  
erik bach/4Ø  
robot1/Ø1  
proc frem  
jp og hn
```

Når programnavnet er skrevet, skal du trykke på <RETURN> tasten. Så slettes teksten ovenfor, undtagen det, du har skrevet, og datamaten skriver

Tryk nu på f6 igen!

Du skal altså trykke på <f6> tasten (husk nu at holde <SHIFT> trykket ned mens du trykker på funktionstasten). Så snart, du har gjort det, begynder diskettedrevet at køre, og alt, hvad du har liggende i arbejdsområdet, bliver gemt.

Læg mærke til advarslen: *Indtastes et eksisterende navn, slettes programmet på disketten*. Det vil fx. sige, at hvis du tidligere har gemt et program med navnet *jens/05* og så bruger dette navn igen, bliver det tidligere gemte program slettet og *erstattet* af det, du nu har i datamaten.

Du kan udnytte denne egenskab ved systemet, hvis du har forbedret et program, og ønsker at gemme dette I STEDET FOR det tidligere. Det svarer til det, der sker, når du indspiller et nyt musikstykke på et bånd, du tidligere har brugt. Når du indspiller den nye musik, slettes det, der hidtil har ligget på båndet. Man siger også, at det gamle musikstykke overspilles.

Når et program på en diskette bliver erstattet af et nyt, siger man, at det gamle program bliver *overskrevet*.

Men i almindelighed *skal du tænke dig godt om, før du genbruger et navn ved gemning på disketten*.

Hvis du vil kende navnene på de programmer, der indtil nu er gemt på disketten, skal du trykke på <f2> tasten (altså <SHIFT>+<f1>). Så får du en udskrift af diskettens *katalog*.

Hvis du ønsker at *hente* et program fra disketten ind i datamatens arbejdsområde, skal du først få skrevet et *katalog* ved at trykke på <f2> (altså <SHIFT>+<f1>). Når kataloget er skrevet, skal du styre markøren opad, så den står ud for navnet på det program, du vil hente. Derpå skal du trykke på <f4> (altså <SHIFT>+<f3>). Markøren forsvinder, mens programmet hentes, og lidt efter skriver datamaten

*** system klar ***

Ved at bruge *list*-kommandoen kan du kontrollere, at programmet er hentet ind.

Vigtigt at bemærke: Hvis du i forvejen har et program liggende i datamatens arbejdsområde, bliver det sat sammen med det, der hentes ind fra disketten. Hvis du ikke ønsker dette, må du først slette det, der nu ligger i arbejdsområdet. Det kan du gøre ved at bruge kommandoen

del-

Men vær varsom med denne kommando. Den sletter alt, hvad der står i arbejdsområdet, undtagen selve styreprogrammet. Den skal altså kun bruges, når der rigtigt skal ryddes op!

I nogle tilfælde kan det være, du ønsker at sætte flere programmer på disketten sammen til eet program i arbejdsområdet. Så skal du blot bruge fremgangsmåden, der er beskrevet ovenfor, på hver enkelt af de programmer, du ønsker at sætte sammen. Man siger også, at programmerne bliver flettet sammen. Det engelske ord "merge", som kommer frem på skærmen, når du trykker på <f4>, betyder netop "flette".

Læg mærke til, at alle programmer, der bliver gemt på den måde, som er beskrevet i begyndelsen, får sat teksten "leg." foran det navn, du taster ind. Det er for at man kan se, at disse programmer er beregnet til at styre LEGO robotter med. Hvis der ligger andre programmer på disketten, kan du få det fuldstændige katalog at se ved at skrive kommandoen

cat

Hvis du vil vide mere om, hvordan man gemmer og henter programmer med COMAL, må du læse i den håndbog, som følger med COMAL-kapslen, eller i en lærebog om COMAL, fx. "COMAL-80 på 64'eren" (SYSTIME 1984).

VIGTIG BEMÆRKNING. Hvis en ydre enhed, fx. en diskettestation eller en printer, bruges samtidig med, at en model kører, går urene for langsomt, og man kan risikere, at en tæller mister et eller flere skift. I almindelighed bør man altså *ikke* bruge andre ydre enheder samtidigt med styrekassen.

Stikordsregister

- afbrydeknapp 9
- afvent 29
- alarm 33
- at 31
- automat 21
- byggeplade 47
- decimalbrøker 25
- dobbelt transportbånd 12
- flag 58
- flag hejst 58
- flag taget ned 58
- flette programmer 71
- for-kommandoen 25
- funktionstasterne 69
- føler 8
- gemme program 69
- hente program 70
- input 31
- interfacebox 9
- katalog 70
- kommando 14
- kommando-linje 14
- list 17
- liste over program 32
- lys-hul i føler 8
- lysfølsomt element 8
- nst 68
- nulstil 58
- overskrevet program 70
- page 30
- pariserhjulet 35
- print 31
- printer 32
- procedure 16
- procedurehoved 27
- procedures krop 16
- procedures navn 16
- program 32
- programnavn 69
- renum 20
- robot 21
- robotarmen 44
- samtidige processer 56
- sl 14
- sluk 14
- stilur 54
- strømforsyning 9
- stu 68
- styrekasse 9
- styreprogram 9
- styrke 39
- technic 1090 8
- tlr 68
- tlt 36
- tn 14
- tælle-impuls 38
- tællehjul 8
- tæller 58
- tæltil 36
- tænd 14
- udl 68
- udløbet 54
- vagthund 23
- variabel 31
- vent 14
- vent 'på' vovse 23
- vovse 23
- vpv 23
- vt 14

Hold styr på robotterne – med Comal 80 handler om, hvordan man kan styre og kontrollere robotter – små model-robotter bygget af LEGO klodser. Model-robotter kan programmeres helt efter de samme principper, som bruges ved programmering af industri-robotter.

Styreprogrammerne er skrevet med COMAL-80 som grundlag, men man behøver ikke have nogen forkundskaber i dette sprog. Det, man har brug for at vide, er forklaret i teksten, hvorfor bogen kan anvendes både til undervisning og til selvstændigt arbejde med robot-teknik.

For at kunne anvende systemet kræves:

- LEGO Technic Control I, art. nr. 1090
- LEGO Interface A, art. nr. 9750
- LEGO Byggeplade, art. nr. 1085
- LEGO kabel til C64, art. nr. 9765

ISBN 87-7351-544-2