RcComal 80
Brugervejledning

PN : 99000927

Nøgleord: RcComa180, PICCOLINE, Partner

Resume Denne vejledning beskriver RcComa180 på PICCOLINE og Partner mikrodatamaterne.

Udgave Januar 1987

Udgivet af A/S Regnecentralen af 1979, København

RcComal80

Brugervejledning

Indholdsfortegnelse

1. Introduktion

COMAL (COMmon Algorithmic Language) er et programmeringssprog, der siden 1975 har været det mest anvendte sprog i den danske undervisningssektor. Sproget blev først implementeret af Regnecentralen a/s på datamatserierne RC3600, RC7000 og RC8000.

COMAL blev udarbejdet for at tilgodese de brugere, der ønskede bedre sprogstruktur og flere faciliteter end dem, man møder i BASIC. Dog ønskede man at fastholde det nære interaktive miljø, der er kendetegnet for en række BASIC-fortolkere, og som har gjort dette sprog velegnet til undervisningsbrug.

Det oprindelige forslag til COMAL blev udarbejdet af studielektor Børge Christensen, Tønder Statsseminarium i 1974.

I 1979 nedsatte man en arbejdsgruppe bestående af Børge Christensen, repræsentanter for undervisningssektoren samt repræsentanter for en række maskinleverandører.

Målet for arbejdsgruppen var, på grundlag af de erfaringer man bl.a. havde høstet fra de tidlige COMAL implementeringer, at udarbejde et forslag til forbedringer af COMAL.

Resultatet af dette arbejde blev et forslag til standardisering, den såkaldte COMAL80-kerne.

Denne manual indeholder en komplet beskrivelse af sproget RcComal80, samt en beskrivelse af, hvordan sproget er implementeret på Rc PICCOLINE og Rc Partner. RcComal80 indeholder en række faciliteter udover selve COMAL80-kernen.

Manualen skulle kunne læses af såvel den uerfarne som den trænede programmør. Den er delt op i tre hoveddele:

Første del er afsnittene 2-12, der indeholder en beskrivelse af, hvordan man indtaster programmer, hvilke variabeltyper der eksisterer, samt en koncentreret gennemgang af en række RcComal80 faciliteter.

Anden del er afsnit 13, der er et referenceafsnit, hvori alle RcComal80-nøgleord er beskrevet præcist.

Tredje del er appendix A-H, der indeholder referencer, mulige fejlmeddelelser, en detaljeret gennemgang af blandt andet skærm og tastatur, ASCII-tabel og en række RcComal80-programeksempler, der kan indtastes og udføres på enten PICCOLINE eller Partner.

Den uerfarne programmør bør starte med at læse afsnit 2-11, hvorimod programmøren med COMAL-erfaring kan gå direkte til afsnit 13.

Manualen indeholder en række programmer som eksempler. Disse er alle indtastet og afprøvet på PICCOLINE og Partner. Programmerne skal illustrere sætningstyper, sprogstrukturer osv., og må ikke altid opfattes som værende det bedste program til løsning af det givne problem. Der er primært lagt vægt på at eksemplerne er letforståelige og lettilgængelige. Ud fra denne målsætning skal det derfor bemærkes, at programmerne forudsætter, at man indtaster tekst med små bogstaver. Dette er gjort ud fra ønsket om at gøre programmerne letlæselige og for ikke at gøre problemet omkring store/små bogstaver til det centrale.

2. Opstart og indtastning af programmer

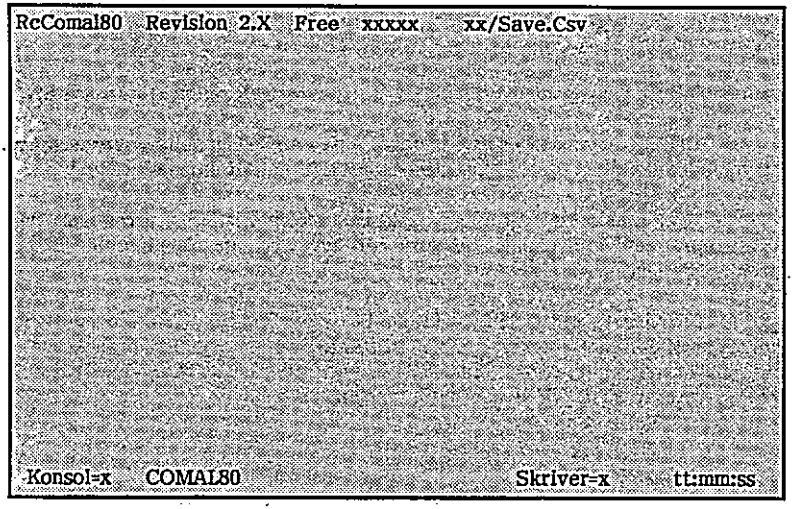
Før man kan komme igang med at bruge RcComal80 skal man installere RcComal80-systemet på en systemdisk. Dette gøres som beskrevet i den medfølgende pakkebeskrivelse. Vi går i det følgende ud fra, at man har foretaget installeringen.

Hvis man ønsker at benytte grafik-kommandoer på Partner eller PICCOLINE, skal man indlæse GSX-grafikmodulet før kaldet af RcComal80. Dette gøres enten

- fra menu systemet eller
- med TMP kommandoen
  GRAPHICS<retur>

Herefter kan RcComal80 startes op. Dette gøres med kommanden:
COMAL80<retur>

RcComal80 startes nu op, og følgende skærmbillede fremkommer :



Cursoren (den blinkende firkant eller understregning) fremkommer på skærmen, som tegn på, at systemet er klar til at modtage input.

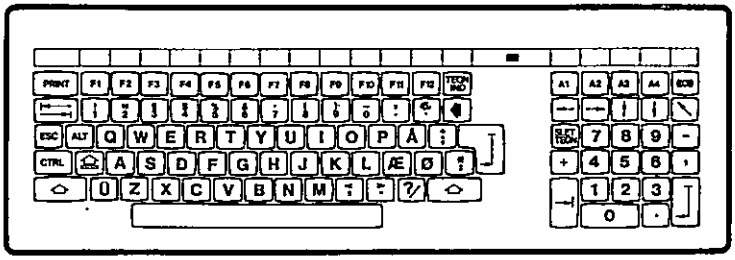
Der kan herefter indtastes program-sætninger eller kommandoer.

Når man vil forlade RcComal80, tastes kommandoen:

BYE<retur>

2.1 Tastaturet

På Partner og PICCOLINE ser tastaturet (RC730/RC739) således ud:



Tasterne omfatter en gruppe lyse og en gruppe farvede eller mørke taster.

De lyse taster anvendes som tasterne på en almindelig skrivemaskine. Nedtrykning af en tast giver normalt en udskrift af det indgraverede tegn.

De farvede eller mørke taster er funktionstasterne. Disse taster har ingen grafisk repræsentation, f.eks. bevirker et tryk på PRINT tasten ikke, at teksten PRINT udskrives.

Nogle af funktionstasterne har altid en bestemt funktion; virkningen af den pågældende funktion kan dog være forskellig afhængig af programmellet. Andre skal først tildeles en funktion, de er programmerbare. På RC730 og RC739 tastaturet findes følgende funktionstaster:

SHIFT ⇧ Fungerer som skiftetasten på en skrivemaskine.

LOCK ⇧ Fungerer som skiftelåsen på en skrivemaskine; dog virker denne kun på bogstaverne. Hvis både SHIFT og LOCK aktiveres, skrives små bogstaver.

CONTROL (CRTL) — Denne tast anvendes altid sammen med en af de lyse taster. Ved at holde CRTL-tasten nedtrykket og derefter trykke på en af de lyse taster, dannes en funktionsværdi i stedet for et tegn. De mest anvendte funktionsværdier har deres egen tast, men kan også dannes vha. CRTL-tasten. Eksempelvis svarer CRTL-M til et tryk på RETUR-tasten.

ALT — ALT-tasten anvendes på lignende måde som CRTL-tasten, blot dannes der et tegn fra det udvidede tegnsæt i stedet for en funktionsværdi.

ESCAPE (ESC) — Programafhængig. Afbryder normalt igangværende funktion (program, automatisk linienummerering, udskrivning ...).

TAB HØJRE (⟶|) — Bevirker tabulering til hver 8. kolonne på skærmen.

TAB VENSTRE (|⟵) — Sletter skærmen.

PRINT — PRINT-tasten benyttes normalt, når man ønsker en direkte udskrift af skærmbilledet på en tilsluttet skriver. Benyttes altid sammen med CRTL-tasten (CRTL+PRINT).

RETUR(↵) — Som hovedregel anvendes tasten for at markere afslutningen på en linie.

SLET (⟵) — Anvendes slet alene, flyttes cursoren en position mod venstre. Trykkes CRTL+SLET, slettes det sidst indtastede tegn, og cursoren flyttes en position mod venstre.

F1 - F12 — Programmerbare funktionstaster. Disse tasters funktion kan programmeres. De er standard udstyret med følgende værdier:

F1: List	F7: Print
F2: Auto	F8: Rename "
F3: Con	F9: Renumber
F4: Del	F10: Save
F5: Delete "	F11: Load
F6: Enter "	F12: Run

TEGN IND og SLET TEGN — Programmerbare funktioner. Anvendes standard ved indsætning og sletning af tegn midt i en linie.

↖ — Programmerbar funktionstast. Anvendes standard til at flytte cursoren til øverste, venstre hjørne af skærmen.

A1 - A4 Programmerbare funktionstaster. Anvendes på samme måde som F1 - F12 (se ovenfor). De er standard udstyret med følgende funktioner:
A1: Slet skærm
A2: Slet resten af skærmen
A3: Slet resten af linien
A4: Slet tegn

SHIFT+A3
SHIFT+A4 Programmerbare funktionstaster. Anvendes ved at holde SHIFT-tasten nede, mens der trykkes på funktionstasten A3 eller A4. De er standard programmeret med følgende funktioner:

SHIFT+A3 bevirker at linien med linienummeret før den aktuelle linie udskrives ovenover den aktuelle linie. Om nødvendigt ruller billedet nedad.

SHIFT+A4 bevirker at linien med linienummeret efter den aktuelle linie udskrives nedenunder den aktuelle linie . Om nødvendigt ruller billedet opad.

((O)) Anvendes ved til- og frakobling af tastaturets lydgiver. Når denne er tilkoblet, høres et klik, hver gang en tast påvirkes.

Funktionstasterne vil virke som beskrevet, med mindre tasternes funktion bliver ændret af brugeren (se C.2.1).

RcComal80 på Partner og PICCOLINE er skærmorienteret, dvs. at man kan benytte de fire cursorpile til at "hoppe" rundt i skærmbilledet, hvorved man kan "genbruge" tekst, der står på skærmen.

Alle taster pånær SHIFT, LOCK, CRTL og ALT repeterer, hvis de holdes nedtrykket i mere end 0,7 sek. Repetitionshastigheden vil langsomt stige, og den maksimale repetitionshastighed opnås efter ca. 20 repetitioner.

RcComal80 tekstskinnen viser, hvad funktionstasterne F1-F12 og A1-A4 er programmeret til.

2.2 Indtastning af programmer

Et program består af en række programlinier. Hver programlinie indledes med et linienummer, der bestemmer liniens placering i programmet.

Et linienummer er et heltal mellem 1 og 9999. Programlinierne kan indtastes i vilkårlig orden, idet RcComal80-systemet selv ordner dem efter voksende linienumre.

Efter linienummeret følger den egentlige sætning. Linien kan eventuelt afsluttes med en kommentar. En kommentar består af 2 skråstreger (//) efterfulgt af kommentarteksten.

Eksempel

```
100          PRINT CHR$(12)  //SLETTER SKÆRMEN
|            |               |
linienr      sætning         kommentar
```

Linien skal afsluttes med <Retur>.

Når linien er afsluttet, vil RcComal80-systemet undersøge, om det er en korrekt RcComal80-sætning (man siger, at systemet syntaksanalyserer linien). Er linien korrekt, gemmes den i program-lageret. Programlageret er det sted internt i RcComal80-systemet, hvor systemet husker de linier, der er indtastet som korrekte RcComal80-sætninger. Tilsammen udgør disse linier et RcComal80-program.

Hvis linien ikke er korrekt, udskrives en fejlmeddelelse i skærmens øverste højre hjørne, og cursoren placeres der, hvor fejlen blev fundet. Det er herefter muligt at rette i linien ved hjælp af de taster, der er beskrevet i afsnit 2.1, og når rettelserne er foretaget, skal linien atter afsluttes med Retur.

Indtastes en sætning med samme linienummer, som en allerede eksisterende, vil den nye sætning erstatte den gamle.

Eksempler på lovlige sætninger

```
10 saldo:=100;slutsaldo:=200;rente:=16
20 år:=1983
30 print år;saldo
40 while saldo<slutsaldo do
50 år:=år+1;saldo:=saldo*(1+rente/100)
60 print år;saldo
70 endwhile
```

Hvad de enkelte linier betyder, er ikke væsentlig i denne sammenhæng. Det er nok at fortælle, at programmet udskriver hvorledes 100 kr forrentes, hvis de i 1983 sættes ind til 16 % i rente. Programmet stopper, når saldoen er nået over 200 kr.

Man kan få en udskrift af de indtastede linier ved at skrive

LIST

efterfulgt af tryk på Retur (eller der kan bruges funktionstast F1).

Eksempel
Tastes linierne fra ovenstående eksempel ind og skrives der LIST, fås følgende udskrift:

```
0010 saldo:=100; slutsaldo:=200; rente:=16
0020 år:=1983
0030 PRINT år;saldo
0040 WHILE saldo<slutsaldo DO
0050    år:=år+1; saldo:=saldo*(1+rente/100)
0060    PRINT år;saldo
0070 ENDWHILE
```

Bemærk, at nogle ord er skrevet med store bogstaver (PRINT, WHILE, DO, ENDWHILE) hvorimod andre er skrevet med små bogstaver. Ordene, der er skrevet med store bogstaver, er de såkaldte nøgleord, som har en bestemt betydning overfor RcComal80 systemet.
Når man er færdig med indtastningen, kan man udføre programmet ved at skrive

RUN

efterfulgt af tryk på Retur (eller funktionstast F12).

Eksempel
Skrives RUN til programmet fra før, fås følgende udskrift:

```
1983 100
1984 116
1985 134.56
1986 156.0896
1987 181.063936
1988 210.03416576
END
AT 0070
```

Hvis man ønsker at afprøve programmet ved 14 % i rente, gøres følgende:

- ved hjælp af cursorpilene på tastaturet rykkes cursoren op i linie 10, hvor der står

  0010 saldo:=100; slutsaldo:=200; rente:=16

- flyt cursoren hen, så den står på 6-tallets plads i 16
- tast 4
- tryk Retur, dvs. at der er indtastet en "ny" linie 10, hvor der står 14 i stedet for 16
- tryk funktionstast A1 (skærmen slettes)
- skriv RUN (husk Retur)

Hermed udskrives:
1984 114
1985 129.96
1986 148.1544
1987 168.896016
1988 192.54145824
1989 219.4972623936
END
AT 0070

Vi har dermed set, at de 100 kr er fordoblet i 1988 ved 16 % i rente, hvorimod de først er fordoblet i 1989 ved 14 % i rente.

Sålænge en linie står på skærmen, kan man altså ved hjælp af cursorpilene ændre i linien eller genbruge tekst fra linien.
Man kan få en vilkårlig af programlinierne frem på skærmen vha. kommandoen LIST.

Ved hjælp af SHIFT+A3 og SHIFT+A4 er det muligt at se et andet udsnit af programmet på skærmen end det nuværende. Hvis man benytter SHIFT+A3 vises linien før den aktuelle på skærmen. Hvis man benytter SHIFT+A4 viseslinien efter den aktuelle linie på skærmen.

2.2.1 Kommandoer til programindtastning

Der findes en række kommandoer, der er nyttige at kende i forbindelse med programindtastning.

2.2.1.1 NEW

Før man starter indtastningen af et nyt program, skal man slette indholdet af programlageret og datalageret. Dette gøres med kommandoen : NEW.

2.2.1.2 AUTO

Hvis flere linier skal indtastes, er det praktisk at lade systemet sætte linienumrene under indtastningen. Dette gøres med kommandoen :

AUTO

hvilket bevirker, at linienummeret 0010 skrives på skærmen. Når linien er indtastet, tastes Retur. Hvis linien er korrekt udskrives 0020, derefter 0030, 0040 osv.

Med AUTO-kommandoen kan man både bestemme startlinienummeret og springet mellem linienumrene. Indtastes f.eks.:

AUTO 100,5

udskrives først 0100. Derefter 0105, 0110, ...

Man standser den automatiske linienummerering igen ved at trykke på ESC-tasten.

Funktionstast F2 betyder AUTO efterfulgt af Retur.

2.2.1.3 RENUMBER

En gang imellem er det nødvendigt at ændre linienumrene i et program f.eks. for at kunne indsætte yderligere programlinier. Dette gøres med kommandoen:

RENUMBER

der bevirker, at linienumrene i programlageret bliver lavet om, således at første linie får linienummeret 0010, anden linie 0020, derefter 0030, 0040 osv.

I kommandoen kan man tillige angive startlinienummeret og springet mellem linienumrene efter omnummereringen. Hvis man angiver:

RENUMBER 100,20

vil linienumrene i programlageret blive lavet om til linienumrene 0100, 0120, 0140 osv.

Funktionstast F9 betyder RENUMBER efterfulgt af Retur.

2.2.1.4 DEL

Hvis man vil fjerne enkelte linienumre, kan det gøres med kommandoen DEL.

Ønsker man f.eks. at fjerne linie 0070 i programlageret, kan det gøres med kommandoen

DEL 70

Ønsker man at fjerne linierne fra linie 0110 til linie 0270, kan det gøres med kommandoen

DEL 110,270

Ønsker man at fjerne linie 1000 og resten af programmet, kan det gøres med kommandoen

DEL 1000,

Hvis man endelig ønsker at fjerne alle linierne fra begyndelsen og til og med linie 0250, kan det gøres med kommandoen

DEL ,250

Funktionstast F4 betyder DEL.

2.2.2 Kommandoer til programudførelse (RUN og CON)

Når et program skal udføres, gøres det med kommandoen

RUN

Hvis man ønsker, at udskriften fra programmet, der normalt kommer på skærmen, skal udskrives på skriveren, gøres det med følgende to kommandoer:

```
SELECT OUTPUT "printer"
RUN
```

Ledetekst i INPUT-sætninger og fejlmeddelelser udskrives dog stadig på skærmen.

Man kan stoppe programudførelsen ved at trykke på ESC-tasten.

Herefter kan man ændre variables værdier, udskrive værdier. Man kan også genstarte udførelsen af programmet med kommandoen

CON

Man må ikke ændre programlinier før en CON-kommando, men på samme måde som før, kan man ændre udskriftsenheden med følgende to kommandoer:

```
SELECT OUTPUT "printer"
CON
```

Funktionstast F12 betyder RUN efterfulgt af Retur, mens funktionstast F3 betyder CON efterfulgt af Retur.

3. Symboler og reserverede ord

3.1 Identifikatorer

Identifikatorer benyttes til at navngive forskellige størrelser i et RcComal80-program. En identifikator består af et bogstav efterfulgt af op til 15 bogstaver, cifre eller tegnet understregning (_), altså et navn på maximalt 16 tegn.

Bemærk der skelnes ikke mellem store og små bogstaver i identifikatornavne. RcComal80-systemet udskriver altid navnene med små bogstaver.

Identifikatorer kan betegne følgende størrelser i et program:

- simple numeriske variable
- taltabeller (vektorer og matricer)
- simple strengvariable
- indicerede strengvariable (teksttabeller)
- brugerdefinerede funktioner
- procedurer
- formelle parametre
- etiketter
- pakker

Hvad disse begreber dækker over, vil blive gennemgået i det følgende.

Samme identifikator må aldrig betegne forskellige størrelser i det samme program, og identifikatorerne må ikke være de samme som systemets reserverede ord, de såkaldte nøgleord.

3.2 Nøgleord

Følgende ord er nøgleord i RcComal80. De har en bestemt betydning overfor systemet, og må ikke benyttes i nogen anden betydning.

ABS
AND
APPEND
AT
ATN
AUTO

BYE

CASE
CHAIN
CHR
CIRCLE
CLEAR
CLOSE
CLOSED
CON
CONTINUE
COPY
COS
CREATE

DATA
DATE
DEL
DELETE
DIM
DIR
DISABLE
DISCARD
DIV
DO
DOCU
DRAW
DRAWTO
DUMP

EDIT
ELSE
ENABLE
END
ENDCASE
ENDFUNC
ENDIF
ENDPACKAGE
ENDPROC
ENDWHILE
ENTER
EOD
EOF
ERR
ERRTEXT
EXEC
EXITPROC
EXP
EXTERNAL

FALSE
FILE
FOR
FROM
FUNC

GET
GLOBAL
GOTO
GPARM
GRAPHICS
GSX

HANDLER

IF
IMPORT
IN
INPUT
INT

KEY

LEN
LIST
LISTPACK
LOAD
LOADPACK
LOCATE
LOG

MARGIN
MOD
MOUNT
MOVE
MOVETO

NEW
NEXT
NOT

OF
ON
OPEN
OR
ORD
OTHERWISE
OUT
OUTPUT

PACKAGE
PALETTE
PASSWORD
PENCOLOR
PI
PREFIX
PRINT
PRINTER
PROC
PUBLIC

RANDOM
RANDOMIZE
READ
REF
RENAME
RENUMBER
REPEAT
RESTORE
RETRY
RETURN
RND
RUN

SAVE
SAVEPACK
SCREEN
SELECT
SGN
SHOWPACK
SHOWPROC
SIN
SIZE
SQR
STEP
STOP
STR
SYS

TAB
TAN
TEXT
TIME
THEN
TO
TRUE

UNTIL
USE
USER
USING

VAL

WHEN
WHILE
WINDOW
WRITE

ZONE

4. Tal og tekst

I dette kapitel vil vi se på, hvordan man foretager udregninger, samt hvordan man behandler tekst i RcComal80. Tekster kaldes også strenge.

4.1 Numeriske variable

På en lommeregner har man ofte et eller flere registre (memories), hvori man kan gemme mellemresultater. Normalt skelner man registrene fra hinanden ved hjælp af et nummer. I RcComal80 skelner man registrene fra hinanden ved hjælp af et navn. Et tal-register kaldes i RcComal80 for en numerisk variabel.

En numerisk variabel, forkortet med symbolet nvar, er et eksempel på en identifikator, og har derfor et navn bestående af et bogstav efterfulgt af indtil 15 bogstaver, tal eller tegnet understregning (_).

Eksempel

Lovlige navne	Ulovlige navne
i42	%k
længde	7t
tid	efterspørgselsindex
gennemsnitsalder	
max_len_10	

4.1.1 Tildeling

En numerisk variabel kan tildeles værdier med sætninger som f.eks.

```
10 pris := 10.85
```

Bemærk, at I RcComal80 benyttes decimalpunktum og ikke komma.

Hvis man skal skrive meget store eller meget små tal benyttes den såkaldte eksponentielle notation, dvs. man skriver tallet som nogle cifre gange en tierpotens. I RcComal80 angives en tierpotens ved bogstavet E efterfulgt af eksponenten.

Eksempel 1

Tal	RcComal80 notation
$1.4*10^{100}$	1.4E+100
-0.0000000001234	-1.234E-10
$200*10^{15}$	2E17

Bemærk, at udelades fortegn foran eksponenten vil det blive opfattet som fortegnet +.
Højre side af lighedstegnet i en tildelingssætning kan enten være tal eller hele udtryk, som f.eks:

```
0010 pris:=13.85
0020 antal:=5
0030 total:=pris*antal
0040 PRINT total
```

Bemærkninger
linie 0030 * betyder "gange med"

Vi kommer nærmere ind på udtryk i de næste afsnit.

:= kan oversættes med "sættes lig med". Betydningen af sætningen er:

udregn højre side og tildel værdien til variablen på venstre side.

Det er altså ikke et matematisk lighedstegn, da f.eks. følgende sætning er lovlig:

```
dagnr := dagnr + 1
```

Her udregnes værdien af dagnr + 1 og værdien tildeles dagnr, variablen dagnr tælles med andre ord op med 1.

4.2 Beregningsudtryk

Man kan f.eks. foretage simple udregninger i RcComal80 ved at benytte PRINT-sætningen.

Eksempel
Indtast følgende lille programeksempel

```
10 PRINT 2+3
```

(Husk at skrive NEW før indtastningen af hvert nyt program). Udfør programmet (skriv RUN). RcComal80 svarer med følgende:

```
5
END
AT 0010
```

Vi har fået udregnet 2 + 3. Man kan selvfølgelig skrive mere indviklede beregningsudtryk, og man må benytte følgende regneoperatorer:

+	addition
-	subtraktion
*	multiplikation
/	division
^	potensopløftning
DIV	heltalsdivision, dvs. 11 DIV 4 = 2, -11 DIV 4 = -3
MOD	rest ved heltalsdivision, dvs. 11 MOD 4 = 3,

Desuden er det tilladt at sætte parenteser.

Bemærk at * aldrig kan underforstås, 2(x+y) skal se således ud i RcComal80:

```
2*(x+y)
```

Endelig kan man benytte funktioner i regneudtryk. RcComal80 har blandt andet følgende matematiske funktioner indbygget:

ABS(x)	Den numeriske værdi af x
ATN(x)	Arcus Tangens til x, resultatet i radianer
COS(x)	Cosinus til x, x i radianer
EXP(x)	Exponentialfunktionen af x
INT(x)	Heltalsdelen af x
LOG(x)	Den naturlige logaritme til x
SGN(x)	Fortegnet af x (-1:negativ, 0:nul, 1:positiv)
SIN(x)	Sinus til x, x i radianer
SQR(x)	Kvadratroden af x
TAN(x)	Tangens til x, x i radianer

I stedet for x kan der enten stå en konstant, en variabel eller et udtryk.

Eksempel

```
0010 // Eksemplet udregner hypotenusen i en retvinklet
0020 // trekant, hvor man kender de to kateter
0030 side1:=3; side2:=4
0040 hypotenuse:=SQR(side1*side1+side2*side2)
0050 PRINT hypotenuse
RUN
5
END
AT 0050
```

4.2.1 Regler for de almindelige regneoperationer

Udregningen af beregningsudtryk følger de almindelige matematiske regneregler, dvs.

1. Udtryk i parenteser udregnes først. Hvis der optræder flere niveauer af parenteser, udregnes den inderste først.
2. Dernæst udregnes funktioner.
3. De almindelige regneoperationer har følgende udregningsprioritet:
   a. Positivt og negativt fortegn
   b. Potensopløftning
   c. Multiplikation, division, modulus og heltalsdivision
   d. Addition og subtraktion
4. Hvis to operatorer har samme prioritet, udregnes udtrykket fra venstre mod højre.

Bemærk at f.eks. 2/3/4 er lig (2/3)/4, dvs. 2/(3*4)

4.2.2 Beregningspræcision

Når man foretager beregninger i RcComal80, regnes der med 13 betydende cifre. Tiereksponenten kan gå fra -128 til 126, dvs. RcComal80 kan regne i følgende positive talområde:

1.000000000000E-128 $\leq$ n $\leq$ 9.999999999999E126

et tilsvarende negativt og endelig tallet 0.

Hvis en beregning giver et positivt resultat, der er mindre end 1E-128 sættes resultatet til nul, hvis resultatet er større end 9.999999999999E126 fås en fejlmelding.

Eksempel

```
0010 megetlille:=1E-128
0020 endnumindre:=megetlille/100
0030 PRINT endnumindre
0040 megetstor:=9E126
0050 forstor:=megetstor*megetstor
0060 PRINT forstor
RUN
0
AT 0050
ERROR: 0106
```

En oversigt over fejlkoder findes i appendix B i denne manual.

4.3 Strengvariable

Indtil nu har vi set på, hvorledes man foretager udregninger i RcComal80. Man kan dog også behandle tekster i sproget. Tekster kaldes normalt for strenge.

RcComal80 tillader brugen af både strengvariable og strengkonstanter. En strengvariabel er et eksempel på en identifikator, dvs. den har et navn som består af et bogstav efterfulgt af indtil 15 bogstaver, tal eller tegnet understregning (_), efterfulgt af et dollar-tegn ($).

Eksempel

Lovlige strengnavne	Ulovlige strengnavne
tekst$	streng ($ mangler)
efternavn1$	7$ (skal starte med bogstav)
postnrogområde$	key$ (reserveret ord)

4.3.1 Erklæring af strengvariable

Strengvariable skal altid erklæres i en DIM-sætning før de benyttes (se afsnit 13.23). I denne sætning angives strengvariablens navn og det maximale antal tegn, der skal kunne gemmes i den.

Eksempel
```
DIM linie$ of 50
```

Når ovenstående sætning udføres, vil der blive reserveret plads i lageret til en streng på maximalt 50 tegn.

4.4 Strengbehandling

4.4.1 Strengkonstanter

Når man skal angive en strengkonstant i et program, skal strengen altid angives i anførselstegn. Dette gælder både tildelingssætninger og PRINT-sætninger.

Eksempel
```
0010 DIM linie$ OF 50
0020 linie$:="ABCDEFGHIJKLMNOPQRSTUVXYZÆØÅ"
0030 PRINT "Dette er en strengkonstant"
0040 PRINT linie$
RUN
```

```
Dette er en strengkonstant
ABCDEFGHIJKLMNOPQRSTUVXYZLÆØÅ
END
AT 0040
```

Bemærk forskellen på PRINT-sætningerne i følgende program:

```
0010 tal:=5
0020 PRINT "tal"
0030 PRINT tal
RUN
tal
5
END
AT 0030
```

I linie 20 udskrives teksten tal, i linie 30 udskrives værdien af variablen tal (=5).

Bemærk Et enkelt anførselstegn kan angives i en strengkonstant ved at skrive to anførselstegn ("")

Eksempel

```
0010 PRINT "Han sagde: ""STOP"" og gik"
RUN
Han sagde: "STOP" og gik
END
AT 0010
```

4.4.2 LEN-funktionen

Man har ofte brug for at vide, hvad den aktuelle længde af en streng er. Til det formål skal LEN-funktionen benyttes.

Eksempel

```
0010 DIM tekst$ OF 80
0020 tekst$:="en lang tekst med mellemrum"
0030 PRINT LEN(tekst$)
0040 tekst$:="0123456789"
0050 PRINT LEN(tekst$)
RUN
27
10
END
AT 0050
```

Bemærk at også mellemrum tæller med i længden af en streng.

LEN-funktionen er en numerisk funktion, og den kan indgå i beregningsudtryk på samme måde som de matematiske funktioner i afsnit 4.2.

4.4.3 Delstrenge

Man kan nøjes med at benytte en del af en strengvariabel i udtryk, udskrifter og lignende.

For en strengvariabel er formatet f.eks.:

linie$(10:20)

Dette betyder, at man udtager delstrengen, der starter i det 10. tegn i strengen linie$ og slutter i det 20. tegn i linie$.

Det er tilladt at udelade det andet argument i delstrengen, f.eks.:

linie$(10:)

Det svarer præcis til delstrengen
linie$(10:10)

Man får med andre ord udtaget det 10. tegn i strengen linie$.

Eksempel
tekst$:= "RcComal80 brugermanual"

Delstreng	**Indhold**
tekst$(1:7)	RcComal
tekst$(2*(4+8/2)+1:22)	manual
tekst$(8:9)	80
tekst$(11:LEN(tekst$))	brugermanual
tekst$(9:)	0

4.4.4 Sammensætning af strenge

Sammensætning af strenge kan ske ved angivelse af + mellem de enkelte elementer.

Eksempel 1

```
0010 DIM tekst$ OF 15, nr$ of 7
0020 nr$:="759"
0030 tekst$:="rc"+nr$
0040 PRINT tekst$
RUN
rc759
END
AT 0040
```

Eksempel 2

```
0010 DIM slogan$ OF 50
0020 slogan$:="Dette er smart"
0030 PRINT slogan$
0040 slogan$:=slogan$(1:9)+"meget "+slogan$(10:14)
0050 PRINT slogan$
RUN
Dette er smart
Dette er meget smart
END
AT 0050
```

Bemærkning
Med dette program er skitseret en metode, hvormed man er i stand til at indsætte en streng ("meget ") i en i forvejen eksisterende streng.

4.5 Logiske udtryk

Som vi skal se på senere, har man ofte brug for at formulere betingelser i RcComal80.

Tegnet > betyder større end.

Eksempel

```
0010 PRINT 7>3
0020 PRINT 3>7
RUN
1
0
END
AT 0020
```

Værdien 1 benyttes til at markere, at en betingelse er sand (7 er jo større end 3). 0 betyder, at betingelsen er falsk.

> kaldes en sammenligningsoperator.

Følgende sammenligningsoperatorer findes i RcComal80:

<	mindre end
>	større end
<=	mindre end eller lig med
>=	større end eller lig med
=	lig med
<>	forskellig fra

Sammenligningsoperatorerne kan ikke blot benyttes til at sammenligne tal, de kan også benyttes til at sammenligne strenge.

Sammenligning af strenge er en alfabetisk sammenligning, og den følger de almindelige regler for alfabetisk ordning, dvs.

" " < "0" < "1" < ... < "9" < "A" < ... < "Å" < "a" < ... "å"

Denne ordning af tegn følger ASCII tabellen i appendix D.

Mere formelt kan man sige, at en sammenligning af to strenge foregår tegn for tegn fra venstre mod højre i strengene, indtil der enten konstateres en forskel, eller at man når til slutningen af den ene eller begge strenge.

Reglerne er herefter følgende:

- Hvis en forskel er konstateret på samme position i de to strenge, er den streng størst, hvis tegns ASCII-værdi er størst.
- Hvis strengene er ens position for position, er den streng størst, som er længst.

Eksempler

```
"AAA"    <    "AAB"
"RC"     <    "RCPartner
"COMAL"  =    "COMAL"
```

Man kan også sammensætte logiske udtryk.

Eksempel
```
0010 tal := 5
0020 PRINT (0<tal) AND (tal<10)
RUN
1
END
AT 0020
```

Bemærkninger
linie 0010 : Den numeriske variabel tal sættes lig 5
linie 0020 : Her udskrives om 0 er mindre end værdien af tal og værdien af tal er mindre end 10. I dette tilfælde er det sandt (tal var jo lig 5), og udtrykket får værdien 1.

PAS PÅ ! Man må ikke skrive betingelsen i linie 20 som

```
PRINT 0<tal<10
```

Dette udtryk vil blive udregnet fra venstre mod højre. Hvis tal f.eks. har værdien 20, vil resultatet alligevel blive 1 (sand).

```
0<tal<10 = 1(sand)<10 = 1(sand)
```

Man kalder ordet AND for en logisk operator.

Følgende logiske operatorer findes i RcComal80 :
AND logisk "og"
OR logisk "eller"
NOT logisk "negering"

4.5.1 IN-operatoren

I forbindelse med strengbehandling findes en nyttig operator, IN-operatoren. Den angiver om en streng optræder som en delstreng i en anden streng, og i bekræftende fald angiver den positionen.

Eksempel

Udtryk	Værdi
"b" IN "abc"	2
"70" IN "RC700"	3
"opera" IN "operator"	1
"bg" IN "bog"	0
"" IN "tekst"	6

Læg mærke til de to sidste værdier.

4.6 Numeriske udtryk

Man kan sammensætte beregningsudtryk og logiske udtryk i samme sætning. Beregningsudtryk, logiske udtryk og sammensætningen af disse kaldes under et for numeriske udtryk.

Eksempel
```
sign:=(x>0)-(x<0)
```

Bemærkninger
Hvis x er større end nul er sign=1(sand)-0(falsk)=1
Hvis x er lig nul er sign=0(falsk)-0(falsk)=0
Hvis x er mindre end nul er sign=0(falsk)-1(sand)=-1

Dette svarer til definitionen af SGN-funktionen (se 4.2).

4.6.1 Aritmetiske-, sammenlignings- og logiske operatorer

For numeriske udtryk gælder følgende prioritetsregler:

Prioritet	Symbol	Funktion
1	+	Positivt fortegn
1	-	Negativt fortegn
2	↑	Potensopløftning
3	*	Multiplikation
3	/	Division
3	DIV	Heltalsdivision
3	MOD	Rest ved heltalsdivision
4	+	Addition
4	-	Subtraktion
5	=	Lig med
5	<>	Forskellig fra
5	<	Mindre end
5	>	Større end
5	<=	Mindre end eller lig med
5	>=	Større end eller lig med
5	IN	Delstrengsposition
6	NOT	Logisk negering
7	AND	Logisk OG
8	OR	Logisk ELLER

5. Kontrolstrukturer

En kontrolstruktur er enten en betinget sætning eller en løkkestruktur.

5.1 Betingede sætninger

En betinget sætning betyder, at en sætning kun udføres såfremt en betingelse er opfyldt.

5.1.1 IF-sætninger

Der er to typer betingede sætninger, nemlig IF-sætninger og CASE-sætninger. Den enkleste form for betingede sætninger er den simple IF-sætning:

Eksempel

```
0010 INPUT "Indtast et beløb :": beløb
0020 IF beløb>100 THEN PRINT "Beløbet er større end 100"
0030 END
```

Bemærkninger

linie 0010 : Her indtastes værdien af variablen beløb

linie 0020 : Den simple IF-sætning. Hvis værdien af beløb er større end 100 udskrives teksten:

Beløbet er større end 100

på skærmen.

Hvis man ønsker at udføre mere end en sætning, når det logiske udtryk (betingelsen) er sandt, kan man bruge den udvidede IF-sætning:

Eksempel

```
0010 INPUT "Indtast et beløb :": beløb
0020 IF beløb>100 THEN
0030    PRINT "Beløbet er større end 100"
0040    PRINT "Der gives 10 % rabat"
0050    beløb:=beløb*0.9
0060 ENDIF
0070 PRINT "Beløbet er ";beløb
```

Bemærkninger

linie 0010 : Her indtastes værdien af beløb

linie 0020-0060 : Den udvidede IF-sætning. Hvis det logiske udtryk er sand (beløb>100) udskrives:

Beløbet er større end 100

Der gives 10 % rabat

Hvorefter beløbet sættes lig 90 % af det gamle beløb.

Bemærk at man benytter ordet ENDIF til at markere, at her slutter rækken af sætninger, der skal udføres, hvis det logiske udtryk er sandt.

Ønsker man at udføre en række sætninger, hvis et logisk udtryk er sandt, og en anden række, hvis udtrykket er falskt, kan man bruge IF-ELSE-ENDIF konstruktionen. I det efterfølgende kaldes en sammenhængende række sætninger for en sætningsliste.

Eksempel

```
0010 INPUT "Indtast et beløb >": beløb
0020 IF beløb>100 THEN
0030    PRINT "Beløbet er større end 100"
0040    PRINT "Der gives 10 % rabat"
0050    beløb:= beløb*0.9
0060 ELSE
0070    PRINT "Beløbet er mindre end 100"
0080    PRINT "Ekspeditionsgebyret er 10 kr."
0090    beløb:= beløb+10
0100 ENDIF
0110 PRINT "Beløbet er herefter ";beløb
```

Bemærkninger

linie 0010 : Her indtastes værdien af beløb.
linie 0030-0050 : Disse linier udføres, hvis det logiske udtryk er sandt, dvs. værdien af beløb er større end 100
linie 0070-0090 : Disse linier udføres, hvis betingelsen ikke er opfyldt.

5.1.2 Multiforgreninger (CASE)

Ved IF-ELSE-ENDIF kan man vælge mellem to alternative sætningslister, men ofte kommer man ud for at skulle kunne vælge mellem flere alternativer. Til dette formål benyttes en multiforgrening (CASE). Den består af et nøgleudtryk og en række sætningslister, hvoraf kun en af sætningslisterne vil blive udført, afhængig af værdien af nøgleudtrykket.

Eksempel

```
0010 INPUT "Indtast et tal : ": tal
0020 CASE SGN(tal) OF
0030 WHEN -1
0040    PRINT tal;"er negativ"
0050 WHEN 0
0060    PRINT "0 er nul"
0070 WHEN 1
0080    PRINT tal;"er positiv"
0090 ENDCASE
0100 END
```

Bemærkninger
linie 0010 : Her tildeles tal en værdi fra tastaturet.
linie 0020 : SGN(tal) er her nøgleudtrykket, det er en funktion, der har værdien +1 hvis tal er positiv, 0 hvis tal er nul og -1 hvis tal er negativ.

Nøgleudtrykket kan enten være et numerisk udtryk eller et strengudtryk.

Eksempel
```
0010 DIM fkt$ OF 1
0020 INPUT "Indtast funktion: I(ndsæt,U(dskriv,S(lut ":fkt$
0030 CASE fkt$ OF
0040 WHEN "I","i"
0050   EXEC indsæt
0060 WHEN "U","u"
0070   EXEC udskriv
0080 WHEN "S","s"
0090   EXEC slut
0100 ENDCASE
0110 END
```

Bemærkning
Ovenstående program kan ikke umiddelbart udføres, da procedurerne indsæt, udskriv og slut, ikke er erklæret.

CASE-konstruktionen udføres således, at den starter med den første WHEN-sætning og udregner, om et af udtrykkene efter WHEN er lig nøgleudtrykket. Hvis dette er tilfældet, udføres sætningerne efter WHEN, ellers undersøges udtrykkene efter det andet WHEN osv.

Hvis ingen WHEN-sætning indeholder et udtryk, der er lig nøgleudtrykket, udskrives en fejlmelding.

Dette kan dog undgås ved angivelse af en OTHERWISE-sætning:

Eksempel
```
0010 DIM fkt$ of 1
0020 INPUT "Indtast funktion: I(ndsæt,U(dskriv,S(lut ":fkt$
0030 CASE fkt$ OF
0040 WHEN "I","i"
0050   EXEC indsæt
0060 WHEN "U","u"
0070   EXEC udskriv
0080 WHEN "S","s"
0090   EXEC slut
```

```
0095 OTHERWISE
0096   PRINT "*** Funktionen eksisterer ikke"
0100 ENDCASE
0110 END
```

Bemærkning
Tastes der ikke I,i,U,u,S eller s, vil programmet nu udskrive:
*** Funktionen eksisterer ikke

Bemærk, at kun een sætningsliste udføres. Følgende program giver således ingen mening:

```
0010 CASE 2 OF
0020 WHEN 2
0030   PRINT "linie 0030"
0040 WHEN 2
0050   PRINT "linie 0050"
0060 ENDCASE
0070 END
```

WHEN-sætningen i linie 0020 vil "opfange" nøgleværdien fra linie 0010 og linie 0050 vil aldrig blive udført.

Dette kan bruges til store betingede strukturer som f.eks.:

```
0010 INPUT "Indtast brevets vægt > ": brevvægt
0020 IF brevvægt<=20 THEN
0030   porto:=280
0040 ELSE
0050   IF brevvægt<=100 THEN
0060     porto:=380
0070   ELSE
0080     IF brevvægt<=250 THEN
0090       porto:=650
0100     ELSE
0110       IF brevvægt<=500 THEN
0120         porto:=1000
0130       ELSE
0140         porto:=1400
0150       ENDIF
0160     ENDIF
0170   ENDIF
0180 ENDIF
0190 PRINT "Portoen er ";porto;"øre."
```

Virkningen af denne konstruktion er den samme som virkningen af følgende CASE-konstruktion:

```
0010 INPUT "Indtast brevets vægt >": brevvægt
0020 CASE TRUE OF
0030 WHEN brevvægt<=20
0040   porto:=280
0050 WHEN brevvægt<=100
0060   porto:=380
0070 WHEN brevvægt<=250
0080   porto:=650
0090 WHEN brevvægt<=500
0100   porto:=1000
0110 OTHERWISE
0120   porto:=1400
0130 ENDCASE
0140 PRINT "Portoen er ";porto;"øre."
```

Bemærkning
I linie 0020 sættes nøgleudtrykket til TRUE, dvs. konstanten SAND. Det bevirker, at WHEN-sætningerne gennemløbes, indtil der findes et udtryk, der er sandt. Er intet udtryk sandt, udføres sætningen efter OTHERWISE. Under alle omstændigheder vil kun en af sætningslisterne efter WHEN (eller OTHERWISE) blive udført.

5.2 Løkkestrukturer

I RcComal80 eksisterer 3 forskellige løkkestrukturer, de er alle beskrevet i det følgende.

5.2.1 Tællesløjfer (FOR-NEXT)

Begrundelsen for at have tællestrukturer ses bedst af et eksempel. Lad os forestille os, at vi har fået til opgave at udskrive en tabel over de 100 første positive tal, deres kvadrattal og deres kubiktal. Dette kunne gøres med følgende program :

```
0010 ZONE 20 // Sæt tabuleringen til 20 tegn
0020 PRINT "X","X*X","X*X*X"
0030 PRINT 1,1*1,1*1*1
0040 PRINT 2,2*2,2*2*2
...
1010 PRINT 99,99*99,99*99*99
1020 PRINT 100,100*100,100*100*100
1030 END
```

I stedet for dette lange program, kan man klare sig med følgende:

```
0010 ZONE 20 // Sæt tabuleringen til 20 tegn
0020 PRINT "X","X*X","X*X*X"
0030 FOR x:=1 TO 100 DO PRINT x,x*x,x*x*x
```

```
0040 END
```

Udskriften fra programmet bliver den samme.

Som man kan se, er virkningen af linie 0030 i det andet eksempel den samme som virkningen af linierne 0030,...,1020 i det første eksempel. Linie 0030 virker således:

1.Først sættes x lig 1
2.Dernæst testes, om x er blevet større end slutværdien (x>100). Hvis den er det, forsættes med linie 0040
3.Ellers udføres PRINT-sætningen, den udskriver x og x*x og x*x*x.
4.Derefter forøges x med 1 og man hopper til pkt 2)

Læg mærke til, at x forøges med 1 (trinværdien er 1). Ønskes en anden trinværdi (f.eks. 10) får sætning 30 følgende udseende :

```
0030 FOR x:=1 TO 100 STEP 10 DO PRINT x,x*x,x*x*x
```

Hermed får x værdierne 1,11,21,31,...,91. 91 er den sidste værdi, for hvis man lægger 10 til, er værdien større end 100 og slutbetingelsen opnået.

Trinværdien kan være negativ. Hvis den er det, vil løkken fortsætte, indtil x (også kaldet tællevariablen) er mindre end slutværdien.

Startværdien, slutværdien og trinværdien behøver ikke at være talkonstanter. Det er også tilladt at angive numeriske udtryk:

```
0030 FOR x:=1/10+1 TO SIN(y*2)*p DO ...
```

Man skal blot være opmærksom på, at startværdien og slutværdien kun udregnes når man løber ind i løkken. Derefter huskes værdierne som konstanter.

Ønsker man at udføre mere end en sætning, kan man benytte den udvidede FOR-NEXT sætning. Den har følgende struktur :

```
0030 FOR x:=1 TO 100 DO
0031    PRINT x,x*x,x*x*x
0032 NEXT x
```

Alle sætninger mellem FOR og NEXT udføres for hvert gennemløb af løkken. Det er desuden tilladt at have FOR-NEXT løkker indeni hinanden. Hvis vi fortsætter med ovenstående program, kunne det have følgende udformning:

```
0030 FOR x:=1 TO 100 DO
0031    FOR potens := 1 to 3 do PRINT x'potens,
0032    PRINT
0033 NEXT x
```

5.2.2 REPEAT-konstruktionen

Hvor FOR-NEXT strukturen gentager en række sætninger på grundlag af en tællevariabel, kan repeat også gentage en række sætninger, indtil et logisk udtryk (en betingelse) er sandt.

Eksempel

```
0010 RANDOMIZE
0020 ZONE 20
0030 PRINT "Terning 1","Terning 2"
0040 REPEAT
0050    terning1:=RND(1,6)
0060    terning2:=RND(1,6)
0070    PRINT terning1,terning2
0080 UNTIL terning1=terning2
0090 END
```

Bemærkninger

Programmet simulerer kast med to terninger. Programmet fortsætter med at køre, indtil de to terninger viser lige mange øjne.

linie 0010 : RANDOMIZE bevirker, at de "tilfældige tal" bliver forskellige fra gang til gang.

linie 0020 : Udskriftszonen sættes til 20 tegn.

linie 0030 : Overskrift udskrives.

linie 0040-0080 : REPEAT-strukturen. Sætningerne i linie 0050-0070 bliver gentaget, indtil værdien af terning1 er lig værdien af terning2.

linie 0050-0060 : "Kast" af de to terninger. RND(1,6) er en funktion der returnerer med et tilfældigt heltal i intervallet 1,2,...,6.

linie 0070 : Udskrift af antallet af terningernes øjne.

Man bør bemærke, at sætningerne mellem REPEAT og UNTIL altid bliver udført mindst 1 gang . Dette gør, at konstruktionen ofte benyttes i forbindelse med INPUT og kontrol af det indtastede.

Eksempel

```
0010 DIM svar$ OF 3
0020 REPEAT
0030    INPUT "Ønsker du at fortsætte ? ": svar$
0040    IF svar$<>"ja" AND svar$<>"nej" THEN
0050       PRINT "Svar venligst ja eller nej"
0060    ENDIF
0070 UNTIL svar$="ja" OR svar$="nej"
```

Bemærkninger
linie 0010 : Alle strengvariable skal erklæres.
linie 0030 : Her indlæses værdien af svar$ fra tastaturet.
linie 0040-0060 : Hvis svar$ ikke er lig teksten "ja" eller "nej" udskrives: Svar venligst ja eller nej.
linie 0020-0070 : Linierne gentages, indtil svar$ er enten "ja" eller "nej".

5.2.3 WHILE-konstruktionen

I stil med REPEAT er WHILE en konstruktion, der gentager udførelsen af en eller flere sætninger på grundlag af et logisk udtryk (en betingelse). Forskellen er, at man ved WHILE konstruktionen fortsætter sålænge en betingelse er opfyldt

Eksempel
```
0010 INPUT "Indtast det indsatte beløb : ": startbeløb
0020 INPUT "Indtast det ønskede beløb  : ": slutbeløb
0030 INPUT "Indtast rentesatsen        : ": rente
0040 PRINT "Termin              Saldo"
0050 saldo:=startbeløb; termin:= 0
0060 WHILE saldo<slutbeløb DO
0070   termin:= termin+1
0080   saldo:= saldo*(1+rente/100)
0090   PRINT USING "$$$$$      $$$$$$$$$$.$$":termin,saldo
0100 ENDWHILE
```

Bemærkninger
Programmet udregner den tid, der går før et indsat beløb i en bank får vokset sig op til et ønsket slutbeløb, forudsat at renten er konstant.
linie 0010-0030 : Her indtastes de ønskede værdier for startbeløbet, slutbeløbet og renten.
linie 0040 : Kolonneoverskrift.
linie 0050 : Saldoen sættes lig det indsatte beløb, og terminnummeret sættes til nul.
linie 0060-0100 : Linierne repeteres, sålænge saldo er mindre end slutbeløb.
linie 0070 : Terminen forøges med 1 periode.
linie 0080 : Der lægges renter til saldoen.
linie 0090 : Formateret udskrift, der bevirker, at tallene kommer til at stå under hinanden (enere under enere, tiere under tiere osv.).

6. Indlæsning og udskrivning

Dette kapitel indeholder en kort fremstilling af de muligheder, der er for udskrift og indlæsning på skærm, skriver, tastatur og diskettefiler.

En fuldstændig gennemgang af sætningerne findes i kapitel 13 i følgende afsnit:

6.1 INPUT-sætningen

Når man skal indlæse tal eller tekst direkte fra tastaturet, kan det gøres med sætninger som f.eks.:

```
0010 INPUT alder,højde
```

Når denne sætning udføres, fremkommer cursoren på skærmen som tegn på, at systemet forventer indtastning fra tastaturet.

Brugeren kan derefter indtaste værdien for alder, efterfulgt af et komma (,) og værdien for højde, efterfulgt af Retur, f.eks.

```
14,160
```

Da det kan være svært at huske rækkefølgen for inddata kan man få udskrevet en ledetekst med følgende sætning:

```
0010 INPUT "Indtast alder og højde > ":alder,højde
```

Når denne sætning udføres, udskrives

```
Indtast alder og højde >
```

Cursoren fremkommer tillige på skærmen, som et tegn på, at systemet forventer indtastning fra tastaturet. Herefter kan brugeren indtaste de ønskede værdier.

I anførselstegnene må der angives en vilkårlig tekst, som dog ikke selv må indeholde anførselstegn.

6.2 PRINT-sætningen

Når man skal udskrive resultater på skærmen benyttes PRINT-sætningen. Sætningen kan udskrive

- tekst
- værdien af et udtryk
- værdien af variable og konstanter
- tomme linier

Efter PRINT kan man anføre en række elementer, adskilt med komma eller semikolon. Kommaet deler hver udskriftslinie i en række kolonner (zoner), hvis bredde kan angives med ZONE-sætningen.

Eksempel 1

```
0010 ZONE 10
0020 PRINT "x","tekst"
```

Når ovenstående program udføres, fås følgende udskrift :

```
x          tekst
```

x udskrives i kolonne 1, tekst i kolonnerne 11 til 15. Udskriftskolonnerne er således 10 tegn i bredden. Hvis ZONE ikke angives, anvendes ZONE 0.

Angives semikolon mellem elementerne sættes 1 mellemrum, hvis foregående element er et tal, ellers sættes intet mellemrum.

Eksempel 2
0010 PRINT "RC";35*20+59;"PICCOLINE"

Når ovenstående sætning udføres, fås følgende udskrift:

RC759 PICCOLINE

6.3 Datastrømme

6.3.1 Filer og ydre enheder

På disken eller kassettebåndet kan man både lagre programmer og data. For at kunne gøre dette, skal programmet eller dataene have et navn. Et område, hvor program eller data gemmes, kaldes for en fil.

Et navn på en fil har følgende format :

unit/navn.type

unit er diskens nummer (altså 1,2 osv.). Dette svarer til CCP/M's unit begreb, A i CCP/M svarer til 1, B til 2, C til 3, osv. 0 svarer til kasettebåndoptager.

navn er et navn bestående af maximalt 8 tegn. Navnet må bestå af alle tegn bortset fra følgende tre: "?/ . Det er lovligt at angive små bogstaver i navnet, men det vil internt blive lavet om til store bogstaver. Dette gælder ikke æ, ø og å, idet det ikke er tilladt at bruge Æ, Ø og Å. Bogstaverne Æ, Ø og Å ændres derfor til æ, ø og å.

type er et navn bestående af maximalt 3 tegn. Navnet må bestå af alle tegn bortset fra følgende tre: "?/ . Det er lovligt at angive små bogstaver i navnet, men det vil internt blive lavet om til store bogstaver. Benytter man SAVE og LOAD-kommandoerne tilføjer RcComal80 automatisk typen CSV.

Lovlige filnavne	Ulovlige filnavne
1/HANOI	9/Miltprogram
2/demoprog.csv	2/demoprog?

Ofte arbejder man kun på en bestemt diskenhed ad gangen. I stedet for hele tiden at angive <unit>/ i alle navne kan man selv erklære et såkaldt præfix, som er en tekststreng, der vil blive sat foran filnavnene i sætningerne.

Eksempel
PREFIX "1/NBA"
LOAD "PROG"

svarer til

PREFIX ""
LOAD "1/NBAPROG"

Efter opstart af RcComal80 udfører systemet en prefix-kommando svarende til den unit man starter RcComal80 op fra. Hvis man starter RcComal80 op fra unit A i CCP/M, vil systemet udføre kommandoen

PREFIX "1/"

Hvis man alligevel ikke ønsker at benytte et eventuelt præfix, skal det første tegn i filnavnet være /.

Eksempel	Filnavn
PREFIX "1/"	
LOAD "/2/DEMO"	2/DEMO
SAVE "PROGRAM"	1/PROGRAM

De ydre enheder har tillige navne, så de kan bruges som filer. Filer og ydre enheder udgør tilsammen datastrømme.

Navnene på de ydre enheder er:

Navn	Enhed
1/CONSOLE	Tastatur og skærm. Indlæses der fra tastaturet, vises tegnene på skærmen.
1/KEYBOARD	Tastatur. Når der indlæses, vises tegnene ikke på skærmen.
1/PRINTER	Skriver.
1/V24	Seriel udgang på maskinen. Inddata skal afsluttes med et linieskift.
portnr/PORT	En af systemets HW-porte.
1/PRINTER0	Systemets skriver 0
1/PRINTER1	Systemets skriver 1
1/PRINTER2	Systemets skriver 2
1/COM	COM-porten på systemet. Inddata skal afsluttes med et linieskift (kun på Partner).
1/SOUND	Systemets lydkreds

Omkring brugen af disse enheder henvises til afsnit 6.3.3.3.

6.3.2 Skrivning og læsning af programfiler

Hvis man har indtastet et program og ønsker at gemme det på disken, kan dette gøres med kommandoen

SAVE filnavn

eller med kommandoen

LIST filnavn

hvor filnavn er navnet på en datastrøm angivet i anførselstegn (se afsnit 6.3.1).

SAVE-kommandoen bevirker, at indholdet af programlageret bliver udskrevet direkte på disken eller kassettebåndet (binært), medens LIST-kommandoen bevirker, at programmet bliver udskrevet i tekstform på disken eller kassettebåndet (ASCII). LIST-kommandoen kan tillige bruges til at udskrive programmet på skiveren.

Eksempel
LIST "PRINTER"

Forudsat at PREFIX er sat til "1/" vil ovenstående kommando bevirke, at programmet i programlageret vil blive udskrevet på skriveren.

Programmet kan hentes igen fra disken eller kassettebåndet med følgende kommandoer

LOAD filnavn

eller

ENTER filnavn

LOAD-kommandoen anvendes, hvis programmet er gemt med SAVE, og ENTER-kommandoen anvendes, hvis programmet er gemt med LIST.

LOAD-kommandoen sletter programlageret før den indlæser programmet, medens ENTER-kommandoen indlæser programmet oveni det program, der eventuelt ligger i programlageret.

Udelades type i filnavnet ved SAVE og LOAD på Partner og PICCOLINE tilføjes automatisk typen CSV (Comal SaVe).

Bemærk, at ved LOAD af et program, udskrives navnet på den LOADede fil i statuslinien øverst på skærmen. Hvis man herefter foretager nogle rettelser i programmet og ønsker at gemme det igen under samme navn, skrives blot

SAVE

hvorefter det rettede program vil erstatte det forrige.

SAVE/LOAD bør anvendes frem for LIST/ENTER, da

- Udskrivning og indlæsning går hurtigere
- Filerne normalt fylder mindre

Funktionstast F10 betyder SAVE, og funktionstast F11 LOAD. Funktionstast F6 er ENTER.

6.3.3 Skrivning og læsning af datafiler

Der eksisterer to typer datafiler:

1. Sekventielle filer
2. Filer med direkte tilgang (RANDOM-filer)

En sekventiel fil er en datafil, hvor data kun kan læses eller skrives forfra. Man kan sammenligne en sekventiel fil med et bånd på en båndoptager, hvor man kun kan spole helt tilbage, og så afspille forfra, hvis man har brug for noget inde på båndet.

En fil med direkte tilgang er en datafil, hvor man kan læse en vilkårlig del af filen, uden først at have læst det foregående. Man kan sammenligne en fil med direkte tilgang med en grammofonplade på en grammofon, hvor man kan flytte pickup-armen direkte ind til det, man har "brug" for.

Bemærk, at man ikke kan benytte filer med direkte tilgang på kassettebånd.

Den generelle håndtering er den samme for begge filtyper, nemlig:

- Før brug skal filen oprettes med en CREATE-sætning (dette gøres dog automatisk ved sekventielle filer)
- I et program skal følgende trin udføres, hvis en fil bruges:
  1. "Åbning" med en OPEN-sætning
  2. Læsning/skrivning i filen
  3. "Lukning" med en CLOSE-sætning

6.3.3.1 Sekventielle filer

Når man skal bruge en sekventiel fil i programmet skal den åbnes (OPEN), dvs. at filen skal knyttes til et såkaldt strømnummer i programmet. Formatet for en OPEN-sætning er:

OPEN FILE strømnr , filnavn , mode

hvor

strømnr er et numerisk udtryk lig enten 1,2,3,4 eller 5
filnavn er navnet på filen
mode er enten WRITE, APPEND eller READ

Betydningen af mode er følgende:

- WRITE: Filen oprettes, og der skrives forfra i den
- APPEND: Der skrives i forlængelse af det, der tidligere er skrevet ud i filen. APPEND-mode kan ikke benyttes til filer på kassettebånd.
- READ: Der læses forfra i filen.

Når filen er åbnet, kan der enten læses eller skrives i den. Enhver reference til filen foregår via strømnr.

Man kan skrive i filen på forskellige måder:

- WRITE FILE bevirker binært udskrift af data
- PRINT FILE bevirker tekstudskrift af data, dvs. i samme format som ved en simpel PRINT-sætning (ASCII-format).

WRITE FILE anvendes normalt i forbindelse med udskrift i diskette- eller kassettebåndsfiler, hvorimod PRINT FILE normalt benyttes i forbindelse med ydre enheder som f.eks. skærm og skriver.

Eksempel

```
0010 OPEN FILE 1,"DATAFIL",WRITE
0020 REPEAT
0030   INPUT "Indtast et tal (0=slut) ": tal
0040   WRITE FILE 1:tal
0050 UNTIL tal=0
0060 CLOSE FILE 1
```

Bemærkninger
Programmet modtager tal fra tastaturet, og skriver dem i filen DATAFIL.
Linie 0010 : Filen oprettes og åbnes til skrivning.
Linie 0020-0050 : Tal indtastes, gemmes i filen og hvis den indtastede værdi er nul hoppes ud af løkken. Nul er den sidste værdi, der gemmes i filen.
Linie 0060 : Filen lukkes efter brug.

Herefter kan dataene indlæses igen. Det kan gøres med følgende program :

Eksempel
```
0010 OPEN FILE 1,"DATAFIL",READ
0020 WHILE NOT EOF(1) DO
0030   READ FILE 1:tal
0040   IF NOT EOF(1) THEN PRINT tal
0050 ENDWHILE
0060 CLOSE FILE 1
```

Bemærkninger
Linie 0010 : Filen åbnes til læsning
Linie 0020-0050 : Her benyttes funktionen EOF. Det er en logisk funktion, der bliver sand, når der forsøges at læse ud over det sidste element. Linie 0030 og 0040 bliver således udført, sålænge der er data i filen.
Linie 0030 : READ FILE er læsesætningen, der læser data skrevet med WRITE FILE.
Linie 0060 : Filen lukkes efter brug.

READ FILE indlæser de data, der er udskrevet ved hjælp af WRITE FILE, og man må gerne angive flere argumenter i både READ FILE og WRITE FILE sætningerne. De enkelte argumenter skal da blot være adskilt med et komma (,).

Filer, der er skrevet med PRINT FILE kan indlæses med INPUT FILE. Her skal man blot erindre, at formatet i filen nøjagtig svarer til det format, man får med simple PRINT sætninger f.eks. på skærmen. Kommaer i parameterlisten vil derfor kun bevirke tabulering til næste PRINT-kolonne, hvilket kan give fejl i forbindelse med strengbehandling.

Hvis man ønsker at benytte PRINT FILE og INPUT FILE er det sikreste således kun at angive eet PRINT- eller INPUT-argument pr. linie, da man derved er sikker på, at de enkelte argumenter bliver adskilt med Retur i filen.

Når man er færdig med at læse eller skrive i filen, skal filen lukkes. Det gøres med sætningen

CLOSE FILE strømnr

der lukker en bestemt strøm, eller

CLOSE

der lukker alle åbne strømme.

6.3.3.2 Filer med direkte tilgang

En fil med direkte tilgang består af en række nummererede poster, der kan hentes hver for sig. På forhånd skal brugeren fastlægge, hvad en post skal indeholde og dermed udregne dens størrelse.

Bemærk, at man ikke kan benytte filer med direkte tilgang på kassettebånd.

Ved udregning af poststørrelsen skal man benytte, at

- en numerisk variabel fylder 8 tegn
- en streng fylder "længden af strengen + 2" tegn

Eksempel
Hvis vi tænker os følgende dimensionering

DIM navn$ OF 30,klasse$ OF 5

og lader nr og hægte betegne numeriske variable, da gælder:

Post	Poststørrelse
nr,navn$,klasse$	8+(30+2)+(5+2)=47
nr,hægte	8+8=16

Desuden skal brugeren vurdere det maximale antal poster i filen. Når dette er gjort, kan filen oprettes. Det gøres med sætningen

CREATE filnavn,længde

filnavn er en strengvariabel eller en strengkonstant der indeholder navnet på filen,
længde er et numerisk udtryk, der angiver filens længde i blokke a 1024 tegn.

Eksempel
CREATE "MEDLEMSDATA",20

Et udtryk for filens længde får man lettest ved at benytte formlen:

fillængde:=(poststørrelse*antalposter) DIV 1024 + 1

Når man skal bruge filen i programmet skal den åbnes (OPEN), dvs. at filen skal knyttes til et såkaldt strømnummer i programmet. I en OPEN-sætning skal man ikke angive, om man ønsker at læse eller at skrive i filen, da begge dele er tilladt, når filen er åbnet.

Formatet for en OPEN-sætning er:

OPEN FILE strømnr,filnavn,RANDOM recl

hvor

strømnr er et numerisk udtryk, der enten er lig 1,2,3,4 eller 5.
filnavn er en strengvariabel eller en strengkonstant, der indeholder navnet på filen,
recl er et numerisk udtryk, der angiver poststørrelsen i tegn (bytes).

Når man har åbnet filen, kan man både læse og skrive i den.

Formatet for skrivning er:

WRITE FILE strømnr,postnr:postbeskrivelse

og formatet for læsning er:

READ FILE strømnr,postnr:postbeskrivelse

hvor
strømnr er det nummer, der anvendtes ved OPEN.
postnr er den post, man ønsker at læse/skrive. Første post har nummeret 1.
postbeskrivelse er de argumenter, der skal læses/skrives.

Eksempel

```
0010 READ FILE 1,17:nr,navn$,klasse$
0020 WRITE FILE 2,92:nr,hægte
```

Efter brug skal filen lukkes igen ved hjælp af en CLOSE-sætning. Formatet er enten

CLOSE FILE strømnr

hvis man ønsker at lukke en bestemt strøm, eller

CLOSE

hvis man ønsker at lukke alle åbne strømme.

6.3.3.3 Ydre enheder

De ydre enheder skal behandles som normale sekventielle tekstfiler, dvs. de åbnes med enten

OPEN strømnr , navn , READ

eller

OPEN strømnr , navn , WRITE

Man kan normalt kun skrive med PRINT FILE-sætningen samt læse med enten INPUT FILE-sætningen eller GET$-funktionen.

GET$-funktionen læser et angivet antal tegn fra en strøm.

Eksempel

```
0010 DIM t$ OF 1
0020 OPEN 1,"KEYBOARD",READ
0030 REPEAT
0040    t$:=GET$(1,1)
0050    IF t$>="0" AND t$<="9" THEN PRINT t$;
0060 UNTIL t$<"0" OR t$>"9"
0070 CLOSE
```

Bemærkninger

linie 0010 : Her dimensioneres en streng til at kunne indeholde eet tegn.
linie 0020 : Den ydre enhed "keyboard" åbnes. Når der indlæses tegn fra tastaturet, vises de ikke på skærmen.
linie 0040 : Der indlæses et tegn fra enheden "keyboard".
linie 0050 : Hvis tegnet er et tal, udskrives det på skærmen.
linie 0060 : Linierne 0030-0060 gentages indtil det indtastede tegn ikke er et tal.

På HW-portene er det muligt både at læse og skrive. For at spare på antallet af strømme er det lovligt både at læse og skrive på den samme strøm, selvom den er åbnet i enten READ- eller WRITE-mode.

For ikke at få automatiske linieskift i forbindelse med HW-port udskrivning, skal man huske at sætte sidebredden til 0 med sætningen

MARGIN 0

6.3.4 Fjernelse af filer

En fil kan fjernes fra disketten med sætningen

DELETE filnavn

Eksempel
0010 PREFIX "1/"
0020 DELETE "/2/DEMO.SAV"
0030 DELETE "HANOI.LST"

Udføres sætningen DELETE filnavn, når der ikke findes en fil der hedder filnavn, gives der ikke nogen fejlmeddelelse.
Udføres kommandoen DELETE filnavn, når der ikke findes en fil der hedder filnavn, gives der en fejlmeddelelse.

DELETE-kommandoen har ingen virkning på kassettebåndsfiler.

Funktionstast F5 betyder DELETE.

6.3.5 Omdøbning af filer

Ønsker man at give filen et andet navn, kan den omdøbes med sætningen

RENAME gl,nyt

hvor
gl er filens nuværende navn
nyt er det ønskede filnavn uden angivelse af unit i navnet

Eksempel 2
RENAME "/2/DEMO.CSV","LOGON.CSV"
0010 RENAME "FIL.DAT","FIL.OLD"

Ulovlige omdøbninger:	**Bemærkninger**
RENAME "/2/DEMO","/2/PROG"	Man må ikke angive unitnr i nyt
RENAME "F","FIL0123456789"	Det nye navn skal være lovligt

6.4 Oversigt over I/O sætninger

Skemaet i dette afsnit giver et overblik over den række indlæsnings- og udskrivnings- (I/O-) sætninger og kommandoer, der eksisterer i RcComal80.

Medium/Datatype	Indlæsning	Udskrivning	Format
Tastatur og skærm	INPUT	PRINT	ASCII
	KEY$	SCREEN$	Direkte
Datastrømme	INPUT FILE	PRINT FILE	ASCII
	READ FILE	WRITE FILE	Binært
	GET$		Transparent
Programmer	ENTER	LIST	ASCII
	LOAD	SAVE	Binært

7. Tal- og tekst-tabeller

Indtil nu har vi kun beskæftiget os med de såkaldt simple variable, dvs. at en talvariabel kun kan indeholde et tal, og at en tekstvariabel kun kan indeholde en tekst.

Ofte kommer man ud for, at man gerne vil gemme flere tal eller flere tekster under det samme navn. Til dette formål har man tabelbegrebet.

7.1 Taltabeller

En taltabel (eller array) består af en række tal. Værdierne eller elementerne i tabellen kaldes tabellens komponenter, og de kan hver for sig opfattes som numeriske variable. En taltabel kan have en eller flere dimensioner. Har den dimensionen en, kaldes den en vektor, - har den dimensionen to, kaldes den en matrix.

7.1.1 Taltabelkomponenter

Hvert element i en taltabel er kendetegnet med navnet på taltabellen, efterfulgt af et index i parentes, f.eks.

pris(1), pris(2) ,... pris(9), pris(10)

For en todimensional taltabel angiver det første index rækkenummeret, og det andet index søjlenummeret. Tabellen kan f.eks. have følgende udseende :

antal(1,3), antal(1,4), antal(1,5), antal(1,6)
antal(2,3), antal(2,4), antal(2,5), antal(2,6)
antal(3,3), antal(3,4), antal(3,5), antal(3,6)

7.1.2 Erklæring af taltabeller

Før man kan benytte en taltabel, skal den være erklæret i en DIM sætning (se afsnit 13.23). I denne sætning angives tabellens navn, øvre indexgrænse og eventuelt nedre indexgrænse. Angives den nedre indexgrænse ikke, sættes den til 1.

Eksempel 1
0010 DIM pris(10),antal(3,3:6)

Med denne sætning erklæres de to tabeller i afsnit 7.1.1.

Eksempel 2

```
0010 INPUT "Antal elever ? ":maxelev
0020 DIM højde(maxelev)
0030 FOR i:=1 TO maxelev DO INPUT "Højde ? ":højde(i)
0040 // Så er højden indtastet og lagret
0050 sum:=0
0060 FOR i:=1 TO maxelev DO sum:=sum+højde(i)
0070 gennemsnit:=sum/maxelev
0080 maxhøjde:=højde(1)
0090 minhøjde:=højde(1)
0100 FOR i:=2 TO maxelev DO
0110   IF højde(i)>maxhøjde THEN maxhøjde:=højde(i)
0120   IF højde(i)<minhøjde THEN minhøjde:=højde(i)
0130 NEXT i
0140 PRINT "Gennemsnitshøjde :";gennemsnit
0150 PRINT "Max højde        :";maxhøjde
0160 PRINT "Min højde        :";minhøjde
```

Bemærkning
Man kan i ovenstående eksempel se, at index for en taltabel kan indlæses, og det behøver således ikke være fastlagt på det tidspunkt, man siger RUN. Det skal blot være fastlagt før erklæringen af tabellen i programmet.

7.2 Teksttabeller

En teksttabel består af en række strenge (tekster).

Hvert enkelt element i tabellen kan opfattes som en strengvariabel. En teksttabel kan have en eller flere dimensioner.

7.2.1 Teksttabelkomponenter

Hvert element i en teksttabel er kendetegnet med navnet på teksttabellen (incl $) efterfulgt af et index i parentes, f.eks.

navn$(1), navn$(2), ... navn$(10)

Ønsker man at udtage en delstreng af en teksttabel er formatet f.eks.

navn$(1)(3:5)

Dette betyder, at man udtager delstrengen, der starter i det 3. tegn i elementet navn$(1) og slutter med det 5. tegn i elementet.

7.2.2 Erklæring af teksttabeller

Før man kan benytte en teksttabel, skal den erklæres i en DIM sætning (se afsnit 13.23). I denne sætning angives tabellens navn, hvert elements maximale længde, øvre indexsgrænse og eventuelt nedre indexgrænse. Angives den nedre indexgrænse ikke, sættes den til 1.

Eksempel
```
0010 DIM navn$(10) OF 80
```

Hermed defineres en teksttabel med 10 elementer, indiceret fra 1 til 10. Hvert element må maximalt indeholde 80 tegn.

8. Procedurer og funktioner

Når man skal løse større problemer, er det oftest hensigtsmæssigt at splitte det store problem op i delproblemer og løse disse hver for sig.

Til dette formål er procedure- og funktionsbegrebet udviklet.

For at illustrere anvendelsen af procedurer, betragter vi følgende programmeringsopgave:

På en skole skal oprettes et register over eleverne og disses adresser. Til dette formål skal vi udarbejde et program, der kan:

- indsætte nye elevers data
- rette i oplysningerne for en elev
- slette elevers data

Programmet har da følgende hovedstruktur:

```
opstart
GENTAG følgende:
  INDLÆS kommando
  HVIS kommando=ind SÅ
    få elevoplysninger
    find ledigt elevnr
    gem elevoplysninger
  HVIS kommando=ret SÅ
    få elevnr
    hent elevoplysninger på disk
    ret i oplysninger
    gem elevoplysninger
  HVIS kommando=slet SÅ
    få elevnr
    hent elevoplysninger på disk
    slet elevoplysninger
INDTIL kommando=færdig
afslut
```

Programmet kunne se således ud i RcComal80:

```
0010 EXEC opstart
0020 REPEAT
0030    PRINT
0040    INPUT "Kommando: I(nd, R(et, S(let, F(ærdig ": k$
0050    PRINT
0060    CASE k$ OF
```

```
0070    WHEN "I","i"
0080      EXEC fåelevoplysn
0090      EXEC findledigtnr(elevnr)
0100      EXEC gemelev(elevnr)
0110    WHEN "R","r"
0120      EXEC fåelevnr
0130      EXEC hentelev(elevnr)
0140      EXEC retelev
0150      EXEC gemelev(elevnr)
0160    WHEN "S","s"
0170      EXEC fåelevnr
0180      EXEC hentelev(elevnr)
0190      EXEC sletelev(elevnr)
0200    WHEN "F","f"
0210      // Slut på program
0220    OTHERWISE
0230      PRINT "*** Ulovlig kommando"
0240    ENDCASE
0250  UNTIL k$="F" OR k$="f"
0260  EXEC afslut
```

Bemærkninger
EXEC betyder "udfør" og navnet bagefter refererer til et procedurenavn, der beskrives i det følgende. Ordet EXEC kan udelades. Læg mærke til, at flere procedurer (fåelevnr, hentelev, gemelev) kaldes flere gange. Procedurerne befinder sig andre steder i programmet, eller de er gemt på disken som externe procedurer.

8.1 Simple procedurer

En af de procedurer, som anvendes i programmet, er proceduren fåelevoplysn. Den kunne f.eks. se således ud:

Eksempel

```
0400  PROC fåelevoplysn
0410    REPEAT
0420      INPUT "Klasse     :": klasse$
0430      INPUT "Navn       :": navn$
0440      INPUT "Adresse    :": adresse$
0450      INPUT "Postnr by :": postnrby$
0460      INPUT "Er oplysningerne korrekte ? (J/N) ": svar$
0470    UNTIL svar$="J" OR svar$="j"
0480  ENDPROC fåelevoplysn
```

Bemærkninger

linie 0400-0480 : Her defineres proceduren fåelevoplysn. Når der i hovedprogrammet står EXEC fåelevoplysn, vil sætningerne mellem PROC og ENDPROC blive udført. Proceduren indlæser de enkelte oplysninger for en elev, og giver brugeren mulighed for at ændre i det indtastede.

En procedure må gerne kalde sig selv. I dette tilfælde kaldes proceduren rekursiv.

Til vores program har vi desuden brug for en procedure, der kan indlæse et nummer på en elev, og checke, om eleven eksisterer i registeret.

Eksempel

```
0290 PROC fåelevnr
0300   REPEAT
0310     INPUT "Elevnr : ":elevnr
0320     ok:=FALSE
0330     IF elevnr>=1 AND elevnr<=max THEN
0340       ok:=(status$(elevnr:elevnr)="O") // optaget
0350     ENDIF
0360     IF NOT ok THEN PRINT "*** Eleven eksisterer ikke"
0370   UNTIL ok
0380 ENDPROC fåelevnr
```

Bemærkninger

linie 0290-0380 : Her defineres proceduren fåelevnr. Når der i hovedprogrammet står EXEC fåelevnr, vil sætningerne mellem PROC og ENDPROC blive udført.

linie 0340 : status$ er en streng med længden max, der har følgende betydning.
status$(n:n)="O" : elevnr n optaget
status$(n:n)="S" : elevnr n slettet
status$(n:n)="L" : elevnr n ledig.

linie 0300-0370 : Man løber i strukturen indtil brugeren angiver et elevnr, der er optaget (dvs. i brug).

8.2 Parameteroverførsel

Ofte er det nødvendigt at overføre data fra hovedprogrammet til proceduren. Dette kan f.eks. ske som i proceduren hentelev:

Eksempel

```
0560 PROC hentelev(nr)
0570   READ FILE 1,nr: klasse$,navn$,adresse$,postnrby$
0580 ENDPROC hentelev
```

Bemærkninger
Denne procedure har en forskel i forhold til de tidligere viste procedurer, idet der er angivet en variabel i en parentes efter procedurens navn. Denne variabel kaldes den formelle parameter til proceduren. Hvis man f.eks. kalder proceduren med sætningen EXEC hentelev(8) vil nr i linie 0560 blive erstattet med tallet 8. Kaldes proceduren med sætningen EXEC hentelev(x) vil nr blive erstattet med værdien af x.

Som almindelige parametre, kan man overføre numeriske variable og strengvariable.

8.3 REF angivelse

Hvis proceduren producerer en værdi, som skal tilbage til hovedprogrammet, kan dette gøres ved at angive REF foran parameteren.

Eksempel
```
0500 PROC findledigtnr(REF nr)
0510   nr:=1
0520   WHILE nr<=max AND status$(nr:nr)="O" DO nr:=nr+1
0530   PRINT "Eleven har fået nummer ";nr
0540 ENDPROC findledigtnr
```

Bemærkninger
linie 0500 : Før parameteren nr står ordet REF. Det angiver, at ikke blot værdien af parameteren i en tilsvarende EXEC sætning skal overføres, men at værdien af nr skal returneres til den kaldende parameter. Dette betyder, at hvis proceduren kaldes med sætningen EXEC findledigtnr(elevnr), vil variablen elevnr have samme værdi som nr, når der returneres fra proceduren.
linie 0520 : Her fremfindes et nr der ikke er optaget.

Da man returnerer værdier gennem REF-parametre, må man ikke kalde proceduren med andet end variable som parametre. Følgende kald er derfor ulovligt :

```
EXEC findledigtnr(8)
```

Et forslag til programmet, der er blevet skitseret i begyndelsen af dette kapitel, er vist i appendix E.

Hvis en parameter i en procedure eller funktion er en vektor, en matrix eller en teksttabel, skal ordet REF altid angives foran parameteren, uanset om proceduren eller funktionen ikke ændrer på værdierne.

Eksempel

```
0010 PROC skrivtabel(REF a(),dima)
0020   FOR i:=1 TO dima DO PRINT a(i),
0030   PRINT
0040 ENDPROC skrivtabel
0050 INPUT "Antal elever ":n
0060 DIM alder(n),højde(n)
0070 MARGIN 80
0080 ZONE 10
0090 FOR nr:=1 TO n DO
0100   INPUT "Alder ? ":alder(nr)
0110   INPUT " Højde ? ":højde(nr)
0120 NEXT nr
0130 PRINT "Alder:"
0140 EXEC skrivtabel(alder,n)
0150 PRINT "Højde:"
0160 EXEC skrivtabel(højde,n)
```

Bemærkninger

linie 0010-0040 : Her erklæres en procedure med navnet skrivtabel. Proceduren har to parametre. Den første er en endimensional taltabel (en vektor), den anden er et tal. Parentesen efter a-et markerer, at parameteren er en vektor. Proceduren udskriver elementerne i a, fra element nr 1 til element nr dima.

linie 0050 : Indlæsning af antallet af elever

linie 0060 : Dimensionering af to vektorer.

linie 0070 : Sidebredden sættes til 80 tegn.

linie 0080 : Kolonnebredden sættes til 10 tegn.

linie 0090-0120 : Indlæsning af alder og højde.

linie 0130 : Udskrift af teksten Alder:

linie 0140 : Kald af proceduren skrivtabel. Første parameter er vektoren alder, anden parameter er antallet af elementer i alder.

linie 0150 : Udskrift af teksten Højde:

linie 0160 : Kald af proceduren skrivtabel. Første parameter er nu vektoren højde, anden parameter er antallet af elementer i højde.

Hvis parameteren til en procedure skal være en matrix skrives f.eks. REF a(,) i procedurehovedet. (I dette tilfælde er matricen en 2-dimensionel taltabel).

Parametertype	Eksempel på procedurehoved	Eksempel på procedurekald
Tal	PROC p(i)	EXEC p(7) eller EXEC p(j)
	PROC p(REF i)	EXEC p(j)
Vektor (Array)	PROC p(REF v())	EXEC p(vektor)
	PROC p(REF m(,,,))	EXEC p(matrix)
Streng	PROC p(s$)	EXEC p("tekst") eller EXEC p(str$)
	PROC p(REF s$)	EXEC p(str$)
Teksttabel	PROC p(REF t$())	EXEC p(strtabt$)

8.4 Lukkede procedurer

Ofte bliver procedurer små selvstændige programmer med egne variabelnavne. For at undgå konflikt med hovedprogrammets variable, kan procedurerne gøres lukkede.

Vi forestiller os, at vi skal lave et program, der skriver 10 linier med 10 stjerner på hver linie. Man kunne da lave følgende program:

Eksempel

```
0010 PROC stjerner
0020    FOR x:=1 TO 10 DO PRINT "*";
0030    PRINT
0040 ENDPROC stjerner
0050 FOR x:=1 TO 10 DO
0060    EXEC stjerner
0070 NEXT x
0080 END
```

Udføres ovenstående program, fås følgende udskrift:

```
**********
```

Altså kun en linie med 10 stjerner. Det som går galt i programmet er, at variablen med navnet x forsøges anvendt som tællevariabel i 2 strukturer indeni hinanden. Når stjerner er blevet udført en gang, vil x have værdien

10, og slutbetingelsen for strukturen i linierne 0050-0070 er opnået. Problemet kan løses, ved man angiver, at variablene i proceduren stjerner alle skal være lokale og ikke have noget med hovedprogrammets variable at gøre. Dette gøres ved angivelse af ordet CLOSED efter PROC stjerner. Proceduren er dermed en lukket procedure.

Eksempel

```
0010 PROC stjerner CLOSED
0020    FOR x:=1 TO 10 DO PRINT "*";
0030    PRINT
0040 ENDPROC stjerner
0050 FOR x:=1 TO 10 DO
0060    EXEC stjerner
0070 NEXT x
0080 END
```

Udføres ovenstående program, fås følgende udskrift:

```
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
```

Og dermed er problemet løst.

Man kan komme ud for, at man alligevel gerne vil have adgang til visse af hovedprogrammets variable eller procedurer fra en lukket procedure. Dette gøres med GLOBAL sætningen indeni den lukkede procedure.

Eksempel

```
0010 PROC p CLOSED
0020    GLOBAL j
0030    i:=10
0040    j:=10
0050 ENDPROC p
0060 i:=1; j:=1
0070 EXEC p
0080 PRINT "i=";i;" j=";j
```

Bemærkninger
linie 0010-0050 : Her erklæres en lukket procedure med navnet p.
linie 0020 : Sætningen angiver, at den variabel j, der benyttes i proceduren er den samme som hovedprogrammets j.
linie 0030 : i sættes lig 10. Da i ikke optræder i GLOBAL sætningen, er det en lokal variabel, der er uafhængig af hovedprogrammets variabel i.
linie 0040 : j sættes lig 10. Da j er angivet i GLOBAL sætningen, er det hovedprogrammets j, der sættes lig 10.
linie 0060 : i og j i hovedprogrammet sættes lig 1
linie 0070 : proceduren p kaldes
linie 0080 : værdien af i og j udskrives.
Hovedprogrammets i er ikke ændret i proceduren p, det er j derimod.

Udføres ovenstående program fås således følgende udskrift:

```
i=1  j=10
```

GLOBAL "henter" variable ude fra hovedprogrammet. Hvis man ønsker at "hente" variable fra det niveau, hvor EXEC-sætningen blev udført, anvendes IMPORT-sætningen.

Forskellen mellem GLOBAL og IMPORT fremkommer kun, når lukkede procedurer optræder indeni hinanden.

Eksempel
```
0010 PROC p CLOSED
0020   PROC q CLOSED
0030     GLOBAL i
0040     PRINT i
0050   ENDPROC q
0060   i:=10
0070   EXEC q
0080 ENDPROC p
0090 i:=0
0100 EXEC p
```

Udføres ovenstående program fås følgende udskrift:

```
0
END
AT 0100
```

Variablen i inde i proceduren q er med andre ord den samme som hovedprogrammets i.

Erstattes linie 0030 med linien

```
0030 IMPORT i
```

fås følgende udskrift:

```
10
END
AT 0100
```

Variablen i inde i proceduren q er nu variablen i fra proceduren p (der hvor EXEC-sætningen står).

8.5 Externe procedurer

I et større program vil der ofte være procedurer, der ikke benyttes særlig ofte. For at spare programlagerplads kan disse defineres som externe procedurer og placeres på en disk. Proceduren optager da kun plads i programlageret, når den kaldes, og pladsen kan senere overtages af andre externe procedurer.

Eksempel
Under navnet PROC1 gemmes følgende procedure:

```
0010 PROC a CLOSED
0020    PRINT "a"
0030 ENDPROC a
```

Under navnet STJERNE gemmes følgende procedure:

```
0010 PROC stjerne(n) CLOSED
0020    FOR i:=1 TO n DO PRINT "*";
0030    PRINT
0040 ENDPROC stjerne
```

programlageret:

```
0010 PROC a EXTERNAL "PROC1"
0020 PROC stjerne(n) EXTERNAL "STJERNE"
0030 EXEC a
0040 EXEC stjerne(10)
```

```
RUN
a
**********

END
AT 0040
```

Bemærkninger
PROC1 og STJERNE er begge navne på diskettefiler, der indeholder de tilhørende procedurer. Procedurene blevet gemt på disken med kommandoen SAVE.

fil PROC1:
linie 0010-0030 : En extern procedure skal altid erklæres CLOSED.
linie 0020 : Proceduren udskriver et a før den returnerer.

fil STJERNE:
linie 0010 : Proceduren har een parameter, n.
linie 0020 : Udskrift af n stjerner.

programlageret:
linie 0010 : Erklæring af proceduren a og i hvilken fil proceduren befinder sig.
linie 0020 : Erklæring af proceduren stjerne og i hvilken fil proceduren befinder sig, samt en angivelse af, hvilke parametre proceduren har.
linie 0030 : Kald af proceduren a
linie 0040 : Kald af proceduren stjerne

Forskellige externe procedurer skal gemmes i hver sin programfil på disken.

Externe procedurer er meget nyttige, idet brugeren hermed har mulighed for at lave sit eget procedurebibliotek, hvor de enkelte procedurer i biblioteket kan benyttes af en række forskellige programmer.

8.6 Fejlbehandlingsprocedure

En HANDLER er en RcComal80 fejlbehandlingsstruktur, der har form som en procedure. Forskellen er blot, at man ikke kan "kalde" en fejlbehandlingsprocedure med EXEC, men at RcComal80 systemet "kalder" strukturen, når der opstår en fejl.

Eksempel
```
0010 PROC fejl HANDLER
0020   PRINT AT(60,1);CHR$(27);CHR$(144);" *** TAL forventet ";
0030   PRINT CHR$(27);CHR$(128)
0040   RETRY
0050 ENDPROC fejl
0060 PRINT CHR$(12)
0070 ENABLE fejl
0080 INPUT AT(10,10),"Indtast et tal :": tal
0090 PRINT AT(60,1);CHR$(31) // Slet eventuel fejlmeddelelse.
0100 DISABLE
0110 ...
```

Bemærkninger

linie 0010-0050 : Her defineres en fejlbehandlingsprocedure. Den udskriver teksten
*** TAL forventet
i øverste højre hjørne på skærmen.

linie 0040 : RETRY betyder, at programudførelsen fortsætter med selve sætningen, hvori fejlen opstod.

linie 0060 : Skærmen slettes.

linie 0070 : HANDLERen fejl gøres aktiv.

linie 0080 : INPUT-sætning hvori der kan opstå fejl, hvis der ikke indtastes et tal, men f.eks. en tekst.

linie 0090 : En eventuel fejlmeddelelse slettes.

linie 0100 : HANDLERen deaktiveres, og systemet overtager fejlmeddelelserne igen.

Man må gerne erklære flere HANDLERe i samme program, men der kan selvfølgelig kun være en aktiveret ad gangen. ENABLEr man en HANDLER, medens en anden HANDLER er ENABLEd, DISABLEs den første HANDLER automatisk.

Fra en fejlbehandlingsprocedure er det muligt at returnere til hovedprogrammet på tre måder:

- RETRY bevirker, at man gentager udførelsen af den fejlbehæftede linie. Dette benyttes oftest i forbindelse med indlæsning/udskrivning.
- CONTINUE bevirker, at programudførelsen forsætter med linien efter den fejlbehæftede linie.
- RETURN har to forskellige virkninger, afhængig af, om HANDLERen blev kaldt i en procedure (el. funktion) eller i hovedprogrammet. Hvis den blev kaldt i en procedure (el. funktion) vil sætningen blive udført, "som om" den stod i proceduren. Hvis HANDLERen blev kaldt i hovedprogrammet, standser programudførslen med en normal fejlmeddelelse.

Eksempel

```
0010 PROC fejl HANDLER
0020    IF ERR=213 THEN CONTINUE
0030 ENDPROC fejl
0040 ENABLE fejl
0050 CREATE "RANDOMFIL",10
0060 DISABLE
0070 ...
```

Bemærkninger

linie 0020 : ERR er en systemfunktion, som returnerer nummeret på den fejl, der forårsagede et kald af HANDLERen. Hvis fejl 0213 (FIL EKSISTERER ALLEREDE) opstår, skal programudførelsen fortsætte med linien efter den linie, hvori fejlen opstod.

linie 0040 : HANDLERen fejl gøres aktiv.

linie 0050 : Her oprettes filen "RANDOMFIL". Hvis der allerede findes en fil med dette navn, kaldes handleren automatisk.

linie 0060 : HANDLERen deaktiveres.

Hvis man ikke ønsker at fortsætte programudførslen i forbindelse med visse fejl, kan man enten have en STOP-sætning i HANDLERen eller man kan lade programudførelsen nå frem til ENDPROC sætningen. Nås ENDPROC-sætningen, udskriver systemet normal fejlmeddelelse. Følgende HANDLER vil således skrive en tekst og derefter give normal fejlmeddelelse, som hvis ingen HANDLER havde været aktiveret:

Eksempel

```
0010 PROC fejlbegået HANDLER
0020   PRINT "Du har begået en fejl"
0030 ENDPROC fejlbegået // ENDPROC nås altid
0040 ENABLE fejlbegået
0050 i:=1/0
RUN
Du har begået en fejl
AT 0050
ERROR: 0104
```

8.7 Funktioner

En funktion er en procedure, der returnerer en værdi.

Udover de indbyggede funktioner i RcComal80 kan brugeren definere sine egne funktioner.

Eksempel

```
0010 FUNC max(a,b)
0020   IF a>b THEN
0030     RETURN a
0040   ELSE
0050     RETURN b
0060   ENDIF
0070 ENDFUNC max
0080
```

```
0090 PRINT max(7,9)
0100 j:=32
0110 størst:=max(max(1,j),11)
0120 PRINT størst
0130 END

RUN
9
32
END
AT 0130
```

Bemærkninger

linie 0010-0070 : Her defineres en funktion med navnet max. Funktionen har to parametre, der skal være to tal, og funktionen returnerer med værdien af det største af tallene.

linie 0030,0050 : RETURN benyttes til at tildele funktionen sin værdi, samt til at returnere fra funktionen.

linie 0090 : Funktionens værdi kan udskrives som enhver anden numerisk variabel eller funktion.

linie 0110 : Et kald af en funktion kan optræde alle steder, hvor der forventes et numerisk udtryk.

Desuden kan brugeren definere strengfunktioner.

Eksempel

```
0010 FUNC uppercase$(enstreng$) CLOSED
0020   FOR i:=1 TO LEN(enstreng$) DO
0030     IF enstreng$(i:)>=CHR$(96) THEN
0040       enstreng$(i:):=CHR$(ORD(enstreng$(i:)-32)
0050     ENDIF
0060   NEXT i
0070   RETURN enstreng$
0080 ENDFUNC uppercase$
0090 PRINT uppercase$("RcComal80")
0100 PRINT uppercase$("RcComal80")(3:7)
```

Bemærk at man tillige kan udtage delstrenge af strengfunktioner (linie 0100).

De regler, der er gennemgået i det foregående for procedurer, kan direkte anvendes på funktioner, dvs. man kan have almindelige parametre, REF parametre, lukkede funktioner samt externe funktioner.

9. Kommandoer

En række af de sætninger og kommandoer, der benyttes til at skrive programmer i RcComal80, gennemgås i dette kapitel. Det skal dog bemærkes, at de oftest benyttede kommandoer (NEW, AUTO, RENUMBER, DEL, RUN og CON) er gennemgået i afsnit 2.2.

Samtlige kommandoer er:

-	AUTO	(13.5)	Automatisk linienummerering
-	BYE	(13.6)	Returner til styresystemet
-	CON	(13.14)	Fortsæt programudførelse
-	COPY	(13.16)	Kopier fil
-	DEL	(13.21)	Slet programlinie(r)
-	DELETE	(13.22)	Slet fil
-	DIR	(13.24)	Udskriv fortegnelse over filer
-	DISCARD	(13.26)	Fjerner pakke(r)
-	DOCU	(13,28)	Udskriv dokumentation
-	EDIT	(13.31)	Ret programlinie(r)
-	ENTER	(13.34)	Indlæs program
-	LIST	(13.63)	Udskriv program
-	LISTPACK	(13.64)	Udskriv procedurer i pakke
-	LOAD	(13.65)	Hent program
-	LOADPACK	(13.66)	Henter RcComal80-pakke
-	NEW	(13.73)	Slet program- og data-lager
-	PRINTER	(13.90)	Skifter skrivernummer
-	RENAME	(13.98)	Omdøb fil
-	RENUMBER	(13.99)	Omnummererer linienumre i program
-	RUN	(13.105	Udfør program
-	SAVE	(13.106)	Gem program
-	SAVEPACK	(13.107)	Gemmer RcComal80-pakke
-	SHOWPACK	(13.111)	Udskriver pakkenavne
-	SHOWPROC	(13.112)	Udskriver procedurenavne
-	SIZE	(13.114)	Udskriv forbrugt lager
-	USER	(13.126)	Skifter brugernummer

En systematisk gennemgang af kommandoerne fås i kapitel 13. Dette kapitel skal udelukkende opfattes som en kort introduktion.

9.1 Kommandoer til programindtastning eller programoversigt

Programindtastningskommandoerne er for næsten alles vedkommende beskrevet i afsnit 2.2 (og underafsnit).

De kommandoer der ikke er beskrevet i afsnit 2.2, bruges til følgende:
- at udskrive en procedure- eller funktionserklæring
- rettelse af et program
- at scrolle (dvs. flytte det viste programudsnit forlæns eller baglæns)
- at udskrive alle procedure- og funktionserklæringer
- at udskrive en eventuel dokumentation for en procedure eller funktion
- at udskrive hvor meget lager, der er brugt.

Hvis man skal rette mange linier i et program, er det ikke altid det nemmeste at LISTe programmet på skærmen, og derefter foretage rettelserne ved hjælp af cursor-pilene og Retur-tasten. Det er derfor muligt at afbryde en LISTning midlertidigt med et tryk på mellemrumstasten og fortsætte LISTningen ved endnu et tryk på mellemrumstasten. Desuden kan en LISTning afbrydes fuldstændigt ved et tryk på ESC-tasten.

I stedet for at angive hvilke linienumre, der skal listes ved kommandoen LIST, kan man angive et procedure- eller funktionsnavn, hvorefter de linier, der er en del af proceduren eller funktionen listes.

Eksempel
Fra eksemplet i kapitel 8 (hvis fuldstændige LISTning er i afsnit E.9 i appendix E) er det muligt at liste en af procedurerne:

```
LIST hentelev
0560 PROC hentelev(nr)
0570    READ FILE 1,nr: klasse$,navn$,adresse$,postnrby$
0580 ENDPROC hentelev
```

Når man vil rette i mange linier, kan man skrive :

```
EDIT
```

hvilket bevirker, at den første linie vil blive udskrevet på skærmen, samt at cursoren vil være placeret umiddelbart efter linienummeret. Man er derefter i stand til at rette i linien ved hjælp af cursorpilene, indsæt-tegn og slet-tegn tasterne. Når en linie er ændret på skærmen tastes Retur for at ændre den tilsvarende linie i programlageret.udskrives.

Hvis man kun skal rette en del af programlinierne, kan man angive startlinienummeret og slutlinienummeret for de linier, der skal rettes. Hvis man skriver:

```
EDIT 50,170
```

vil alle linierne fra 50 til 170 blive udskrevet enkeltvis, som ovenfor beskrevet.

Hvis man ikke ønsker at rette yderligere linier indenfor det angivne interval, trykkes på ESC-tasten.

Hvis man allerede har en programlinie stående på skærmen, men desuden ønsker at se linien før denne programlinie, kan det gøres ved at flytte cursoren op i programlinien og derefter trykke på SHIFT+A3. Derved kommer linien med linienummeret før den aktuelle programlinie op på skærmen. Hvis man ønsker linien efter bruges SHIFT+A4. Man behøver ikke have en programlinie, men kan bruge enhver linie, der starter med et tal.

Ved kommandoen SHOWPROC kan man få en oversigt over de erklærede procedurer og funktioner.

Eksempel
I eksemplet i afsnit E.9 i appendix E er der følgende erklæringer:

```
SHOWPROC
0290 PROC fåelevnr
0400 PROC fåelevoplysn
0500 PROC findledigtnr(REF nr)
0560 PROC hentelev(nr)
0600 PROC gemelev(nr)
0650 PROC sletelev(nr)
0710 PROC skrivelev
0770 PROC retelev
0960 PROC opstart
1120 PROC afslut
```

Ved kommandoen DOCU er det muligt at udskrive erklæringen af en procedure eller funktion sammen med de kommentarer, der er til erklæringen.

Eksempel

```
0010 PROC sletskærm
0020    // sletter skærmen ved at udskrive tegn nr 12
0030    PRINT CHR$(12)
0040 ENDPROC sletskærm
0050 ...

DOCU sletskærm
0010 PROC sletskærm
0020    // sletter skærmen ved at udskrive tegn nr 12
```

DOCU kan også anvendes ved pakker, se kapitel 12.

Der findes tillige en kommando, der udskriver, hvor meget lager der er benyttet i maskinen. Hvis man skriver:

SIZE

kan systemet f.eks. udskrive:

```
program  data    free
00310    00000   56131
```

hvilket betyder, at det program, man har indtastet fylder 310 tegn i lageret, og at der på nuværende tidspunkt er 56131 ledige tegn i lageret. Man skal her blot huske, at alle programmer skal bruge yderligere lager til data, når de udføres.

9.2 Sætninger udført som kommandoer

En række RcComal80 sætninger kan udføres samtidig med, at de indtastes. Dette gøres ved udeladelse af linienummeret i programlinien.

Denne faciliitet er speciel nyttig i forbindelse med

- brug af datamaten som bordkalkulator
- fejlfinding i programmer
- datastrøms indlæsning/udskrivning

Der er dog en række RcComal80 sætninger, der ikke har mening som kommandoer. Disse sætninger er:

CASE-WHEN-ENDCASE	IF(-ELSE)-ENDIF
DATA	INPUT
DISABLE	PROC-ENDPROC
ENABLE	PROC-EXTERNAL
END	REPEAT-UNTIL
FOR(-NEXT)	RETRY
FUNC-ENDFUNC	RETURN
FUNC-EXTERNAL	STOP
GLOBAL	WHILE(-ENDWHILE)
GOTO	

9.2.1 Datamaten som bordkalkulator

Udregninger kan foretages ved hjælp af tildelings- og PRINT-kommandoerne.

Eksempel	Kommantar
NEW	Så er program- og data-lager tomt
antal:=20	variablen antal oprettes og tildeles en værdi
stkpris:=47.85	variablen stkpris oprettes og tildeles en værdi
PRINT antal*stkpris	Resultatet udskrives på skærmen.

9.2.2 Fejlfinding i programmer

Det, at adskillige RcComal80-sætninger kan udføres som kommandoer, gør fejlfinding i programmer lettere.

Hvis man ikke retter i et program efter en RUN, er det således muligt at kalde procedurer enkeltvis med kommandoen

EXEC navn

Efter hvert enkelt kald af procedurerne er det muligt at udskrive samt ændre variables værdier, hvorved fejlfindingsproceduren er gjort simpel.

9.2.3 Datastrømsindlæsning/udskrivning

Alle I/O-sætninger kan tillige bruges som kommandoer, dvs. man kan oprette, åbne, skrive, læse og lukke filer direkte fra tastaturet.

Hvis man f.eks. får fejl 0217: IKKE ÅBEN/ALLEREDE ÅBEN, fordi en fil allerede er åben, kan man ofte klare sig med kommandoen

CLOSE

samt udføre programmet på ny.

Desuden er det f.eks. muligt at få udskrevet navnene på filerne diskdrev nr. 1 med kommandoerne:

```
SELECT OUTPUT "printer"
DIR 1
```

9.3 Diskdrev kommandoer (USER og DIR)

Hvis man ønsker at skifte det aktuelle brugernummer, skal man markere dette overfor systemet. Dette gøres med kommandoen:

USER nr hvor nr er et tal mellem 0 og 15

Det er ikke muligt at skifte brugernummer, hvis man har åbne filer på det gamle brugernummer.

Hvis man undlader argumentet til USER, udskrives det aktuelle brugernummer.

Ønsker man at udskrive en oversigt over filerne på diskdrev nummer 1 bruges kommandoen

DIR 1

Hvis man ønsker udskriften for diskdrev nummer 2, udskiftes ettallet naturligvis blot med et 2-tal.

Kommandoen DIR kan desuden benyttes i programsætninger.

Eksempel
Hvis man vil udskrive indholdet på disk nummer 1 og 2 på skriveren, kan det gøres med følgende program :

```
0010 SELECT OUTPUT "printer"
0020 FOR i:=1 TO 2 DO DIR i
```

9.4 Datastrømskommandoer

Datastrømskommandoerne er for næsten alles vedkommende gennemgået i afsnit 6.3.

Her behandles kommandoen til kopiering af indholdet af en fil.

Eksempel

```
PREFIX ""
COPY "1/HANOI","2/HANOI"
```

Bemærkning
Præfixet sættes til den tomme streng. Det betyder, at de angivne filnavne ikke får sat anden tekst foran. Kommandoen kopierer filen HANOI fra disk nummer 1 over i en ny fil med navnet HANOI på disk nummer 2.

De to filnavne behøver naturligvis ikke være ens.

Man kan desuden kopiere filer på samme enhed.

Eksempel

```
PREFIX "1/"
DELETE "GLDATA"
COPY "NYEDATA","GLDATA"
```

Bemærkning
Med disse kommandoer har man fået kopieret en fil, så man har en gammel version af filen, hvis den nye bliver behandlet forkert i ens program. Dermed er alle data ikke tabt.

Resten af datastrømskommandoerne er som sagt beskrevet i afsnit 6.3, så vi bringer blot et resume af dem her:

LIST filnavn

udskriver et program i tekstformat på enten en ydre enhed eller en diskettefil. Programmet kan læses ind igen med kommandoen

ENTER filnavn

idet man dog skal bemærke, at program- og data-lageret ikke slettes i forbindelse med ENTER (ønskes det slettet, angives NEW-kommandoen før ENTER-kommandoen).

Eksempel
LIST "listfil"
programmet kan senere indlæses med kommandoerne

NEW
ENTER "listfil"

SAVE filnavn

gemmer programlageret på en diskettefil. Formatet i filen svarer til maskinens interne format. Det indlæses igen med kommandoen

LOAD filnavn

Eksempel
SAVE "savefil"
programmet kan senere indlæses med kommandoen
LOAD "savefil"

Man bør tilstræbe at bruge SAVE/LOAD ved normal arkivering af programmer på disken, da kommandoerne fungerer betydeligt hurtigere og filerne normalt fylder mindre end ved LIST/ENTER.

En fil fjernes med kommandoen

DELETE filnavn

og ønsker man at omdøbe en fil, dvs. ændre navnet men ikke indholdet, er kommandoen

RENAME gl filnavn,nyt filnavn

Eksempel
RENAME "/1/oversigt","logon"

Bemærk, at man ikke skal angive diskens nummer i det nye filnavn.

Det er muligt at skifte det aktuelle skrivernummer med kommandoen:

PRINTER 2

Hvis man undlader argumentet til PRINTER, udskrives det aktuelle skrivernummer.

9.5 BYE

Med kommandoen

BYE

forlader man RcComal80 og returnerer til styresystemet.

10. Grafik

RcComal80 på Partner og PICCOLINE er udstyret med en række kommandoer, der via det grafiske 'styresystem' GSX gør det muligt at tegne grafiske billeder enten på skærmen, grafikprintere eller plottere.

Samtlige grafikkommandoer er:

- CIRCLE (13.10)
- CLEAR (13.11)
- CLOSE GRAPHICS (13.13)
- DRAW (13.29)
- DRAWTO (13.30)
- GPARM (13.51)
- GSX (13.52)
- LOCATE (13.67)
- MOVE (13.71)
- MOVETO (13.72)
- OPEN GRAPHICS (13.76)
- PALETTE (13.81)
- PENCOLOR (13.83)
- TEXT (13.121)
- WINDOW (13.130)

En detaljeret gennemgang af grafikkommandoerne findes i kapitel 13, og dette kapitel skal udelukkende opfattes som en introduktion til emnet.

10.1 Opstart af grafik.

Før man kan benytte grafikken på Partner eller PICCOLINE skal det grafiske 'styresystem' GSX hentes ind i maskinen. Dette kan gøres på to måder:

- Fra menusystemet ved at udføre menuindgangen 'Grafik'

- Hvis man ikke betjener sig af menusystemet kan følgende kommando udføres fra TMP inden RcComal80 startes:
  GRAPHICS

Herefter er det muligt at starte grafikken op.

10.2 Grafik i RcComal80

Selve fremgangsmåden for grafik i RcComal80 er følgende:

1. Man vælger den grafiske enhed, dvs. skærm, plotter eller lignende.
2. Det grafiske vindues ramme defineres
3. Tegning indenfor vinduet
4. Afslutning af det aktuelle billede.

10.2.1 Valg af grafisk enhed

I RcComal80 kan man tegne på en grafisk enhed (dvs. skærm, plotter eller skriver) ad gangen. Valget af den aktuelle grafiske enhed gøres med sætninger der har formen

OPEN GRAPHICS 1

Tallet efter OPEN GRAPHICS henviser til den aktuelle grafiske enhed.

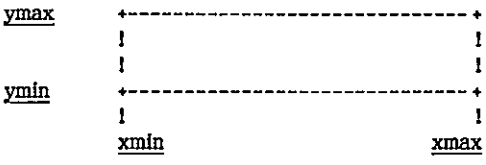
Partner og PICCOLINE er standard udstyret med følgende værdier:

Nr	Ydre enhed
1	Monochrom skærm (sort/hvid skærm)
2	Farveskærm
11	RC621-plotter
21	RC603-skriver eller RC602-skriver

Læg mærke til, at disse tal er uafhængige af RcComal80. Tallene konfigureres i GKONFIG eller Installationsmenuen, som ændrer i filen, der hedder ASSIGN.SYS (se Installationsvejledningen for den pågældende maskine).

10.2.2 Definition af det grafiske vindue

Før man starter på at tegne på den aktuelle grafiske enhed, skal man med nogle tal angive, grænserne for ens (kommende) billede.



Herefter angives vinduet på følgende måde

WINDOW xmin ,xmax ,ymin ,ymax

Eksempel
WINDOW -10,10,0,100

10.2.3 Tegning af streger

Nu kan man tegne indenfor grafikvinduet.

Grafik-operationerne foregår altid ud fra det punkt, man er nået til ved forrige operation (det løbende punkt). Man kan vælge mellem blot at flytte det løbende punkt, eller at tegne en streg (og samtidig flytte det løbende punkt). Man kan enten flytte eller tegne

- relativt, dvs. man i vinduets koordinater angiver, hvor langt der skal flyttes eller tegnes ud fra det løbende punkt, eller
- absolut, dvs. man i vinduets koordinater angiver, hvorhen der skal flyttes tegnes.

Nedenstående skema giver en oversigt over tegne- og flytte-kommandoerne.

	Tegning	Flytning af det løbende punkt
Absolut	DRAWTO	MOVETO
Relativt	DRAW	MOVE

Om man foretrækker absolutte eller relative angivelser afhænger af opgavens karakter, og det er altid muligt at løse det aktuelle problem på begge måder. Nedenstående 2 eksempler giver den samme tegning.

Eksempel 1
```
0010 OPEN GRAPHICS 1
0020 WINDOW 0,10,0,10
0030 MOVETO 2,2
0040 DRAWTO 7,2
0050 DRAWTO 7,7
0060 DRAWTO 2,7
0070 DRAWTO 2,2
```

Eksempel 2

```
0010 OPEN GRAPHICS 1
0020 WINDOW 0,10,0,10
0030 MOVETO 2,2
0040 DRAW 5,0
0050 DRAW 0,5
0060 DRAW -5,0
0070 DRAW 0,-5
```

10.2.4 Afslutning af det aktuelle billede

Når man er færdig med at tegne grafikbilledet, afslutter man med kommandoen

```
CLOSE GRAPHICS
```

Dette bevirker enten

- at skærmen slettes, og "stilles tilbage" til almindelig tekst-tilstand, eller
- at billedet på skriveren tegnes.

Foretager man en ny OPEN GRAPHICS-kommando uden at have udført CLOSE GRAPHICS, udfører systemet selv CLOSE GRAPHICS-kommandoen.

11. Brug af pakker

Pakker består af en række procedurer og funktioner, der kan være programmeret i RcComal80, PolyPascal eller assembler (også kaldet maskinkode). Uanset hvilket sprog pakken er skrevet i, ser dens procedurer og funktioner fra RcComal80 ud - og kan benyttes på præcis samme måde - som almindelige RcComal80 procedurer og funktioner.

Der er en række fordele ved at bruge pakker skrevet i assembler eller PolyPascal. Dels er de hurtige at udføre, og dels giver de adgang til flere faciliteter end pakker skrevet i RcComal80. Desuden giver de mulighed for at lave biblioteker og for at udnytte lageret bedre. Disse fordele uddybes i afsnit 11.1.

Til gengæld er det sværere at skrive og afteste PolyPascal- og assemblerprogrammer. Specielt pakker skrevet i assembler vil derfor næppe blive skrevet af den "almindelige" RcComal80 bruger. Men enhver RcComal80 bruger vil kunne benytte pakker, skrevet af andre, i egne programmer. Bemærk: Man kan normalt ikke rette i pakker fra RcComal80 - man kan kun bruge dem. Undtaget herfra er de RcComal80-pakker, der ikke er beskyttet.

Dette kapitel beskriver, hvorledes man bruger pakker, mens kap. 12 beskriver, hvorledes man programmerer pakker i hvert af de 3 sprog.

11.1 Fordele ved pakker

11.1.1 Hastighed

Assembler er **hurtigere** at udføre end RcComal80, fordi assemblerkode er skrevet i maskinens eget sprog, og derfor kan udføres omgående, hvorimod et RcComal80-program fortolkes af RcComal80-fortolkeren.

PolyPascal-pakker er under udarbejdelsen af pakken blevet oversat fra PolyPascal-sproget og til maskinkode. Derfor er PolyPascal-pakker lige så hurtige at udføre som assemblerpakker.

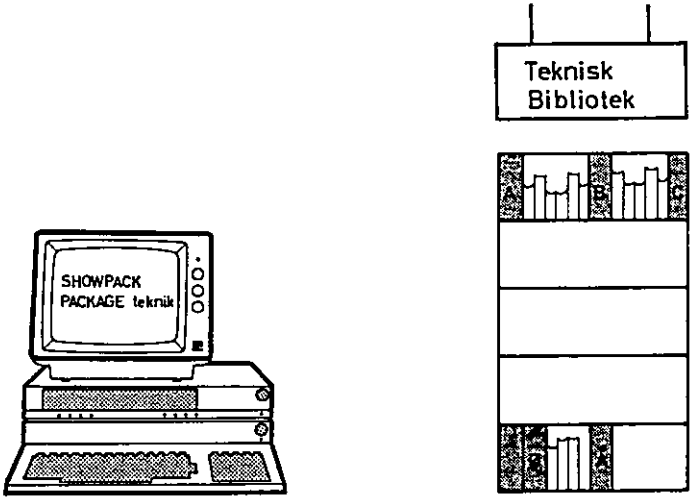
11.1.2 Flere faciliteter

Pakker skrevet i assembler eller i PolyPascal, giver mulighed for at udnytte maskinens faciliteter fuldt ud, f.eks. ved at lave systemkald til det underliggende styresystem eller ved at kunne addressere hele lageret.

Hvis der opstår fejl i en PolyPascal-pakke, eller en assemblerpakke, kan RcComal80 systemet ikke altid begrænse følgerne af fejlen. Det betyder, at det, ved brug af PolyPascal- eller assemblerpakker, er muligt at "slå benene væk under sig selv", således at maskinen må genstartes. Det er i modsætning til almindelig brug af RcComal80, hvor man befinder sig i beskyttede omgivelser og bliver advaret, hvis man forsøger at lave noget galt.

11.1.3 Biblioteker

Ligesom et fagbibliotek f.eks. opstiller og inddeler sine bogsamlinger ud fra hensigtsmæssige temaer, kan man ved hjælp af pakker så at sige skabe sig et **pakkebibliotek**, der under forskellige temaer rummer en række samlinger af nyttige underprogrammer, der kan lånes og benyttes fra ethvert RcComal80 program.



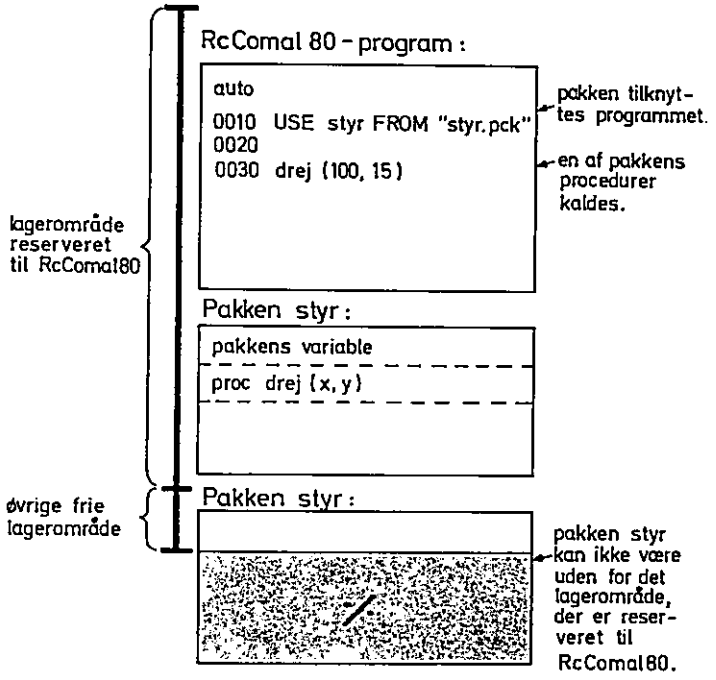
Pakker i RcComal80 er beregnet til at gøre underprogrammer (dvs. procedurer og funktioner) endnu mere anvendelige: En stor opgave kan løses på en overskuelig måde ved brug af underprogrammer. Når opgaven skal løses, kan man opdele den i en række mindre og mere overskuelige opgaver. Derefter kan man benytte procedurer eller funktioner, som underprogrammer, der løser delopgaverne.

Genbrug af underprogrammer er således meget udbredt, og derfor er det en god ide, at samle underprogrammer pakkevis i et bibliotek. En pakke er altså ofte en temapakke, der under samme tema indeholder en samling underprogrammer, der løser typiske delopgaver inden for et bestemt område - f.eks. en matematikpakke, der kan løse forskellige typer ligninger, eller en mekanik-styrepakke, der kan udføre forskellige former for styring af en motor.

Det er vha. procedurer og funktioner i RcComal80 muligt at opnå en del af de fordele, der er ved biblioteker.

11.1.4 Bedre udnyttelse af lageret

Pakker giver mulighed for at udnytte Partner eller PICCOLINE med et stort lager endnu bedre, når man udfører RcComal80 programmer. RcComal80 finder nemlig selv ud af, om der er plads til at placere en PolyPascal-pakke eller en assemblerpakke uden for det lagerområde, som er reserveret til brugerens RcComal80-program. Hvis det ikke er tilfældet, placeres pakken inden for dette lagerområde.
RcComal80-pakker bliver altid placeret indenfor det lagerområde, der er reserveret til RcComal80.



Pakker giver mulighed for at have faciliteter i RcComal80, som kun fylder i lageret, når brugeren ønsker de skal være tilstede.
Pakker fylder mere eller mindre alt efter, hvilket sprog de er skrevet i:

RcComal80-pakker	fylder ca. det samme som det tilsvarende program i RcComal80.
PolyPascal-pakker	består af 2 dele, nemlig PolyPascal-systemet, som optager ca. 12 Kb, og den oversatte pakke.
assembler-pakker	består af maskinkode og fylder normalt ikke særlig meget.

11.2 Kommandoer ved brug af pakker

Til at kunne bruge pakker, er følgende nøgleord:

DISCARD	(13.26)
DOCU	(13.28)
LISTPACK	(13.64)
LOADPACK ... FROM ...	(13.66)
SAVEPACK ... ON ...	(13.107)
SHOWPACK	(13.111)
USE ... FROM ...	(13.125)

I dette afsnit vil vi se nærmere på, hvordan disse kommandoer bruges. I kap. 12 beskrives de nøgleord, der kan benyttes til at programmere RcComal80 pakker.

11.2.1 Tilgængelighed og brug

En pakke, der skal bruges i et RcComal80-program, gøres tilgængelig ved at give en USE-sætning eller en USE-kommando til den fil, der indeholder pakken.
Ved udførelse af en USE-sætning eller en USE-kommando, sker der følgende: Filen med pakken hentes ind i programlageret. Pakken får det navn, der er angivet som parameter til USE-sætningen uanset om pakken på filen hedder noget andet. Derefter udføres pakkens initialiseringsprocedure (se kap. 12).
Bemærk: Man kan bruge (USE) lige så mange pakker, som der er plads til i lageret. En pakke kan dog kun USE's, hvis den ligger på en fil på disketten.
Man kan have flere udgaver af den samme pakke, hvis man i hver af USE-sætningerne angiver et nyt pakkenavn.

Når en pakke er gjort tilgængelig, kan man benytte alle de procedurer og funktioner, som pakken stiller til rådighed for RcComal80 brugere.

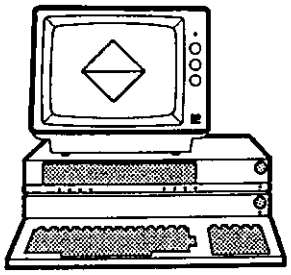
Eksempel
Vi forestiller os, at der på disketten findes en fil: COPY603.PCK
På filen ligger en pakke copy603, der indeholder to procedurer og funktioner, som kan udskrive hhv. hele eller en del af det nuværende grafik-skærmbillede på en Rc603-skriver.

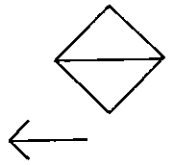
Hvis man vil gøre copy603-pakken tilgængelig, kan man f.eks. udføre følgende program (ved at give kommandoen RUN til programmet). I dette program er der også et eksempel på, hvordan en pakke-procedure (skærmkopi og skærmudsnit) kaldes. Det sker på præcis samme måde, som kald af en almindelig procedure eller funktion i RcComal80.

Bemærk, at programmet kun kan udføres, hvis det grafiske styresystem (GRAPHICS) er hentet ind i maskinen.

```
0010 USE copy603 FROM "copy603.pck"    // pakken gøres tilgængelig
0020
0030 PROC trekant(x,y)
0040   DRAW 2*x,0
0050   DRAW -x,2*y
0060   DRAW -x,-2*y
0070 ENDPROC trekant
0080
0090 OPEN GRAPHICS
0100 WINDOW 0,100,0,100
0110
0120 MOVETO 40,46
0130 xkoor:= 20
0140 ykoor:= 20
0150 trekant(xkoor,ykoor)
0160 ykoor:= -ykoor
0170 trekant(xkoor,ykoor)
0180
0190 skærmkopi                         // udfør pakkeproceduren
0195                                   // skærmkopi
0200 skærmudsnit(200,100,200,80,10)    // udfør pakkenproceduren
0210                                   // skærmudsnit
```

Skærmbilledet under udførelsen af programmet ses på følgende figur, som også viser, hvad henholdsvis skærmkopi og skærmudsnit tegner på skriveren:





11.2.2 Oversigt over pakkens indhold

Efter brug af USE, er pakkens procedurer og funktioner tilgængelige, og man kan få en liste over dem ved at bruge kommandoen:

```
LISTPACK copy603
```

Kommandoen vil give følgende udskrift:

```
PACKAGE copy603
  PROC skærmkopi
  PROC skærmudsnit(startx,starty,antalx,antaly,indryk)
ENDPACKAGE copy603
```

Hvis man vil have mere at vide om pakken eller en af dens procedurer eller funktioner, kan man få dens kommentarer (documentation) udskrevet med kommandoen:

```
DOCU copy603                          eller
DOCU copy603.skærmkopi                eller
DOCU copy603.skærmudsnit
```

Endelig kan man, hvis pakken er skrevet i RcComal80 og hvis man kender et eventuelt password, loade pakken ind vha. kommandoen:

```
LOADPACK lillegsx
```

Når pakken er loadet, vil den ligge i programområdet, og man kan så frit liste og editere i pakken.

Når man er færdig, kan pakken gemmes igen vha. kommandoen:

```
SAVEPACK lillegsx
```

11.2.3 Oversigt over indlæste pakker

Hvis man vil have en oversigt over, hvilke pakker der for tiden er i programlageret, kan man give kommandoen:

```
SHOWPACK
```

I vores tilfælde, vil resultatet så være:

```
PACKAGE copy603
PACKAGE lillegsx
```

Ved hjælp af kommandoerne SHOWPACK, LISTPACK og DOCU kan man således få en fuldstændig oversigt over de biblioteker med procedurer og funktioner, der er i programlageret på et givet tidspunkt.

Bemærk: Selv om en pakke er i programlageret, er det ikke sikkert, at den er tilgængelig. Hvis man vha. en kommando USE'er en pakke og derefter udfører et program, vil pakken være i programlageret, men den er ikke længere tilgængelig. Hvis man retter i et program, hvor der er en pakke tilgængelig, vil pakken ikke længere være tilgængelig, men den er stadig i programlageret.

11.2.4 Fjernelse af pakker

Hvis man ønsker at fjerne pakkerne fra programlageret, bruges kommandoen

```
DISCARD
```

Man kan ikke fjerne en enkelt pakke, man skal fjerne dem alle.

Ved fjernelse af pakkerne sker der følgende for hver af pakkerne:
Hvis pakken indeholder en exitprocedure, udføres denne (se kap. 12). Derefter fjernes pakken fra programlageret.

12. Programmering af pakker til RcComal80

Pakker til RcComal80 kan skrives i RcComal80, PolyPascal eller assembler. Dette kapitel er derfor inddelt i afsnit svarende til disse sprog. Først er der et generelt afsnit om programmering af pakker, der skal læses uanset hvilken slags pakker, man ønsker at lave. Sidst er der et afsnit om avancerede pakker i PolyPascal eller assembler.

Desuden bør kapitel 11 om brug af pakker læses, inden man begynder at lave sine egne pakker.

12.1 Generelt om pakker

En pakke er et program, der har et bestemt format, der gør det muligt at finde de procedurer og funktioner, der kan bruges fra pakken. Når man vil lave en pakke er det derfor nemmest først at lave et almindeligt program (i RcComal80, PolyPascal eller assembler) og afteste dette program, inden programmet laves til en pakke.

Til en pakke bør der laves en skriftlig dokumentation så brugerne af pakken kan se, hvordan pakken bruges, og hvilke fejlreaktioner, de kan komme ud for. Dokumentationen bør også oplyse om, der er specielle ting, man skal gøre, når man bruger pakken.

I hvert af de følgende afsnit er der eksempler på pakker skrevet i de forskellige sprog. Disse eksempler findes som kildetekster på disk 4/4 i PICCOLINE SW1400 release 3.1 eller på distributionsdisketten for RcComal80 release 2.1. Ved at se på disse eksempler er det lettere at lave sine egne pakker.

12.2 RcComal80-pakker til RcComal80

En RcComal80-pakke er et RcComal80-program, der indeholder nogle specielle pakkesætninger og er gemt på diskette med kommandoen SAVEPACK.

Eksempel
Pakken lillegsx er en RcComal80-pakke, der for overskuelighedens skyld kun har to procedurer, polyline og polylinetype:

```
0010 PACKAGE lillegsx
0020   // se beskrivelse af GSX sætning i RcComal80-manual
0030   PUBLIC polyline, polylinetype
0040   DIM kontrol(6),ind(10),pktind(1,2),ud(10),pktud(101,2)
0050
0060   PROC polyline(ant,REF pkti(,))
0070     kontrol(1):= 6
0080     kontrol(2):= ant
0090     kontrol(4):= 0
0100     GSX kontrol,ind,pkti,ud,pktud
0110   ENDPROC polyline
0120
0130   PROC polylinetype(type)
0140     kontrol(1):= 15
0150     kontrol(2):= 0
0160     kontrol(4):= 1
0170     ind(1):= type
0180     GSX kontrol,ind,pktind,ud,pktud
0190   ENDPROC polylinetype
0200 ENDPACKAGE lillegsx
```

Pakken skal starte med en linie indeholdende sætningen PACKAGE og skal afsluttes med ENDPACKAGE. PUBLIC-sætninger angiver, hvilke af pakkens procedurer og funktioner, der kan kaldes af brugeren af pakken. Der må gerne være flere navne i en PUBLIC-sætning (som i eksemplet), og flere PUBLIC-sætninger i pakken. PUBLIC-sætninger må stå vilkårlige steder i pakken og behøver ikke stå før erklæringerne af de procedurer og funktioner, der gøre PUBLIC (det betyder at linie 30 godt kunne indsættes som linie 195).

Initialiseringsrutinen er de sætninger mellem PACKAGE og ENDPACKAGE, der ikke er en del af en procedure eller funktion, dvs. linierne 20-50 og 120.

Hvis man ønsker en exitrutine, angives det med sætningen:

EXITPROC navn

hvor navn skal være navnet på en procedure uden parametre. Der må kun være een EXITPROC-sætning i en pakke. Pakken lillegsx, i eksemplet ovenfor, har ingen exitrutine.

Eksempel

Aftestning af pakken lillegsx, fra eksemplet ovenfor, kan foretages ved at tilføje linierne:

```
0210 // test af lillegsx pakke
0220 USE lillegsx
0230 OPEN GRAPHICS 1
0240 WINDOW 0,100,0,100
0250 DIM pktind(101,2)
0260 // indlæsning af 4 punkter til pktind
0270 ..
....
0400 polyline(4,pktind)
....
0500 polylinetype(3)
....
0600 CLOSE GRAPHICS
```

Når pakken er aftestet, kan den gemmes med kommandoen SAVEPACK, som gemmer selve pakken, men ikke de øvrige sætninger.

Eksempel

SAVEPACK lillegsx

gemmer pakken lillegsx på filen LILLEGSX.PCK

SAVEPACK lillegsx ON "storegsx"

gemmer pakken lillegsx på filen STOREGSX.PCK

Når pakken er gemt med SAVEPACK, kan den bruges i et andet RcComal80 program, som beskrevet i kapitel 11 (se USE). Pakken kan ikke listes, når den bruges. Linienumrene i et program, der bruger pakken, er uafhængige af linienumrene i pakken.

En RcComal80-pakke, der er gemt med en SAVEPACK-kommando, kan hentes ind lageret, så den kan listes og editeres ved kommandoen LOADPACK. LOADPACK laver automatisk en NEW-kommando, dvs. et eksisterende program i lageret fjernes.

Eksempel
LOADPACK lillegsx
    henter pakken fra filen LILLEGSX.PCK
LOADPACK lillegsx FROM "storegsx"
    henter pakken lillegsx fra filen STOREGSX.PCK

Hvis man ønsker at forhindre andre i at ændre ens RcComal80-pakker, kan man give pakken et løsen ved at indsætte følgende sætning (inden pakken gemmes med kommandoen SAVEPACK):

0057 PASSWORD sesam

PASSWORD-sætningen (der må kun være een i en pakke) kan stå et vilkårligt sted i pakken. Hvis der er en PASSWORD-sætning i en pakke, kan LOADPACK kun udføres, hvis man angiver det rigtige løsen. Men pakken kan stadig bruges (USE's).

Det er muligt at bruge andre pakker indeni en RcComal80-pakke. Man skal være opmærksom på, at hvis to RcComal80-pakker i det samme program bruger den samme pakke PAK, så får de to pakker hver sin kopi af PAK.

Hvis en RcComal80-pakke bruger en anden pakke, og pakkehovedet for denne anden pakke ændres, skal:
1) RcComal80-pakken hentes ind i lageret med kommandoen LOADPACK
2) tilføjes en linie med USE af RcComal80-pakken
3) kommandoen RUN udføres
4) RcComal80-pakken gemmes igen med kommandoen SAVEPACK

ellers kan man risikere, at der opstår fejl, når man senere bruger RcComal80-pakken.

Det er ikke tilladt at bruge sætningerne GLOBAL og IMPORT i en RcComal80-pakke.

Dokumentation for en pakke eller PUBLIC procedurer og funktioner, der skrives ud med kommandoen DOCU, skal skrives på samme måde, som dokumentation for almindelige procedurer og funktioner. Dvs. at dokumentationen skrives med to skråstreger efterfulgt af den teksten, altså
//.....

12.3 Programmering af PolyPascal-pakker til RcComal80

PolyPascal-pakker til RcComal80 skal skrives i PolyPascal version 3.10. Det er derfor nødvendigt at have kendskab og adgang til dette sprog, hvis du vil forstå dette afsnit og skrive PolyPascal-pakker.

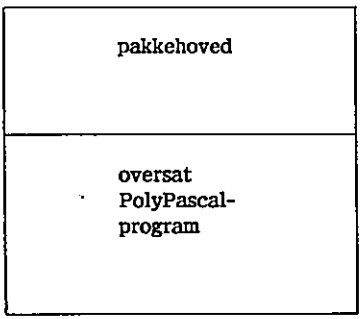
Der skal gøres opmærksom på at programmering i PolyPascal giver adgang til alle faciliteter i maskinen såsom styresystemkald og alle lagerceller, hvilket giver risiko for "at slå benene væk under sig selv", således at maskinen må genstartes. Det står i modsætning til programmering i RcComal80, hvor man befinder sig i beskyttede omgivelser og bliver advaret, hvis man forsøger at lave noget galt.

Når man skal bruge en pakke skrevet i PolyPascal er det nødvendigt, at PolyPascal's run-time omgivelser er med i pakken. Dette bevirker, at pakker skrevet i PolyPascal bliver forholdsvis store (mindst 12K), selv om selve pakken ikke er ret stor. Det kan derfor ske, at der ikke er plads til pakken. Ofte kan det være en fordel at skrive pakken i RcComal80 eller assembler, idet pakker skrevet i disse sprog fylder "meget" mindre.

De eksempler, der omtales i dette afsnit findes alle som kildetekster på en af distributionsdisketterne. Eksemplerne gør det nemmere at lave sine egne pakker, idet det er muligt at genbruge en del fra disse pakker, specielt vedrørende hoved på en pakke og overførsel af parametre.

12.3.1 Opbygning af en PolyPascal-pakke

En PolyPascal-pakke er en fil, der består af et oversat PolyPascal-program flettet sammen med et pakkehoved. Pakkehovedet beskriver de snitflader programmet har overfor RcComal80, når det bruges som en pakke.



Pakkehovedet definerer

- at det er en pakke skrevet i PolyPascal
- for hver af de procedurer og funktioner, der kan bruges fra RcComal80: erklæring af proceduren/funktionen, adressen på koden for proceduren/funktionen, adressen på dokumentationen hørende til proceduren/funktionen.
  (gælder også for initialiserings- og exitrutinen)

De følgende afsnit beskriver de tekniske detaljer i, hvordan man laver PolyPascal-pakken. Her skitseres, hvad der skal gøres for at omdanne et PolyPascal-program til en PolyPascal-pakke, der kan bruges fra RcComal80:

PolyPascal-programmets hovedprogram skal laves til en procedure uden parametre.

Alle procedurer og funktioner, der skal kunne kaldes fra RcComal80, skal have ændret deres erklæring, sådan at de er erklæret som procedurer uden parametre. Hvis de oprindelig havde parametre, skal der tilføjes kodestumper, som sørger for at overføre parametre fra og til RcComal80. Tilsvarende for et funktionsresultat. (hvis procedurerne/funktionerne stadig skal kunne kaldes fra PolyPascal-programmet, skal der således være 2 versioner af dem).

Alle procedurer og funktioner, der skal kunne kaldes fra RcComal80, skal afsluttes med en returnering til RcComal80. Der findes en speciel hjælpeprocedure til dette formål.

Der skal laves et nyt hovedprogram i PolyPascal-programmet. Dette program skal generere en fil som indeholder pakkehovedet for den ønskede pakke. Der findes en inklude-fil på distributionsdisketten, som indeholder en lang række hjælperutiner til at generere denne fil.

Når ovennævnte ændringer er udført, skal det ændrede PolyPascal-program oversættes og køres. Derved bliver der genereret en pakkehovedfil, og en fil med et oversat PolyPascal-program. Disse 2 filer flettes sammen vha. PASPAK programmet.

I de følgende afsnit kommer en mere detaljeret beskrivelse af, hvad der skal gøres.

12.3.2 Konventioner for en PolyPascal-pakke

Der er nogle konventioner, som et PolyPascal-program skal overholde, for at det kan bruges som en pakke af RcComal80:

1) programmets hovedprogram skal lave en fil, hvis indhold er et pakkehoved, der bl.a. beskriver, hvilke procedurer og funktioner, pakken stiller til rådighed

2) denne fil med pakkehovedet skal flettes sammen med CMD-filen, der indeholder det oversatte program

3) der skal erklæres variable, der fylder 4 ord. Variablene bruges ved kald af en procedure eller funktion i pakken

4) procedurer og funktioner, som pakken stiller til rådighed, skal selv sørge for at overføre parametre mellem RcComal80 og pakken

5) procedurer og funktioner, som pakken stiller til rådighed, skal selv sørge for at returnere til RcComal80

For at gøre det nemmere at overholde disse konventioner findes en fil POLYPAS.PAS og et program PASPAK.CMD på distributionsdisketterne.

POLYPAS.PAS skal inkluderes i PolyPascal-programmet, der skal være en pakke (se nedenfor). En udskrift af POLYPAS.PAS findes i appendix H. POLYPAS.PAS indeholder:
- procedurer og konstanter, der gør det nemmere at lave filen med pakkehovedet
- statements, der sørger for at erklære de nødvendige variable (se punkt 3 ovenfor)
- procedurer, som gør det let at flytte parameterværdier fra RcComal80 til pakken og tilbage
- procedurer, der sørger for at returnere til RcComal80 fra en pakke
- andre anvendelige rutiner (beskrives nedenfor)

Programmet PASPAK.CMD fletter en fil med et pakkehoved for en pakke og en fil med det oversatte program sammen til en fil, der kan bruges som pakke. PASPAK skal have en parameter, der angiver filnavnet for de filer, der skal flettes. Det antages at hovedfilen har filtype HOV og programfilen har filtype CMD. Pakkefilen får samme navn, som de to andre, men med filtype PCK.

Et PolyPascal-program, der skal være en PolyPascal-pakke, skal have et hovedprogram, der laver en fil med et pakkehoved, når programmet udføres. Pakkehovedet består af tre dele, hvor den midterste del definerer, de procedurer og funktioner, pakken stiller til rådighed. Der er en række procedurer på POLYPAS.PAS, som gør det nemt at udskrive filen med hovedet; disse procedurer forklares i afsnit 12.3.6.

Eksempel
Hovedprogrammet for pakken MUS består af en række kald af procedurer i POLYPAS.PAS (filen MUS.PAS ligger på distributionsdisketten):

```
BEGIN
  openfile('mus.hov');
  skriv_versionsno;
  skriv_packtype;
  skriv_offset(ofs(dok_mus));          (* dokumentation          *)
  skriv_offset(ofs(mouseinit));        (* adresse initrutine     *)
  skriv_offset(0);                     (* ingen exitrutine       *)

                                       (* PROC musinit           *)
  skriv_navn('MUSINIT');               (* procedurenavnet        *)
  skriv_proc;                          (* procedure              *)
  skriv_offset(ofs(dok_init));         (* dokumentation          *)
  skriv_offset(ofs(mouseinit));        (* offset for proceduren  *)
  skriv_byte(0);                       (* 0 parametre            *)

                        (* FUNC musstatus(REF dx, dy, keyinf)    *)
  skriv_navn('MUSSTATUS');             (* funktionsnavnet        *)
  skriv_realfunc;                      (* reel funktion          *)
  skriv_offset(ofs(dok_status));       (* dokumentation          *)
  skriv_offset(ofs(xstatus));          (* offset for funktionen  *)
  skriv_byte(3);                       (* 3 parametre            *)
  skriv_navn('DX');                    (* parameternavn          *)
  skriv_typeogdim(refpar+realpar,0);   (* REF tal                *)
  skriv_navn('DY');                    (* parameternavn          *)
  skriv_typeogdim(refpar+realpar,0);   (* REF tal                *)
  skriv_navn('KEYINF');                (* parameternavn          *)
  skriv_typeogdim(refpar+realpar,0);   (* REF tal                *)

  skriv_byte(0);                       (* ikke flere procedurer  *)
  skriv_reserver;                      (* adresse på variable    *)
  skriv_packtype;                      (* afslutning af hoved    *)
  closefile;
END.
```

Desuden skal følgende linie indsættes efter den første linie af MUS.PAS
(*$I POLYPAS *)

Den første linie i hovedprogrammet angiver navnet på den fil, hovedet skrives på. Filtypen skal være 'HOV'.

Linie to og tre af hovedet vil være de samme for alle PolyPascal-pakker. De giver pakken et versionsnummer, og fortæller at pakken er skrevet i PolyPascal.

Den fjerde linie angiver om, der i pakken er noget dokumentation for selve pakken, der kan udskrives med kommandoen DOCU. Det beskrives sidst i dette afsnit, hvorledes man laver dokumentationen i pakken.

Den femte linie angiver adressen for initialiseringsrutinen. Denne rutine kaldes automatisk, når det angives at en pakke skal benyttes (se USE i kapitel 11 om brug af pakker). Initialiseringsrutinen benyttes til at initialisere eventuelle variable. Der skal altid være en initialiseringsrutine og denne må ikke have parametre.

Linie seks angiver adressen på exitrutinen. Exitrutinen kaldes automatisk, når en pakke fjernes fra lageret (sker ved kommandoerne DISCARD, NEW og LOAD). Exitrutinen kan ikke have parametre. Der behøver (i modsætning til initialiseringsrutinen) ikke være en exitrutine. I så fald sættes adressen til nul.

Linie 4 til 6 laves ved kald af proceduren skriv_offset (se 12.3.6).

Den midterste del af pakkehovedet definerer de procedurer og funktioner, som pakken stiller til rådighed. I pakken MUS er der kun en procedure (musinit) og en funktion (musstatus).

For hver procedure eller funktion angives:

- navnet (skriv_navn)
- typen for rutinen (skriv_proc, skriv_realfunc eller skriv_stringfunc)
- adresse på en eventuel dokumentation for proceduren (skriv_offset)
- adresse på proceduren (skriv_offset)
- antal parametre (skriv_byte)
- de enkelte parametre

En parameter beskrives ved to linier:
- navnet (skriv_navn)
- typen og dimensionen af parameteren. Udskrives med skriv_typeogdim.

Til at beskrive typen af en parameter, er der defineret nogle konstanter i POLYPAS.PAS:

valuepar	betyder VALUE-parameter
refpar	betyder REF-parameter
realpar	betyder en numerisk parameter
stringpar	betyder en streng parameter

Typen af parameteren fremkommer ved en sum af en af de to første (valuepar eller refpar) og en af de to sidste (realpar eller stringpar).

Når parameteren ikke er en tabel, er dimensionen 0. Hvis parameteren er en tabel, skal parametertypen være refpar (den må ikke være valuepar).

De sidste fire linier i hovedet skal være de samme for alle PolyPascal-pakker.

Når POLYPAS.PAS inkluderes i PolyPascal-programmet, bliver de variable, der skal bruges ved kald af procedurer og funktioner i pakken, erklæret.

Alle procedurer og funktioner, der angives i pakkehovedet (se ovenfor), skal erklæres som procedurer uden parametre, også selvom de skal kunne kaldes som funktioner fra RcComal80, og må ikke have lokale erklæringer af variable.

Alle procedurer og funktioner, der angives i pakkehovedet skal selv hente eventuelle parameterværdier fra RcComal80 og aflevere REF-parameterværdier og funktionsresultater tilbage til RcComal80. Der findes rutiner i POLYPAS.PAS, der gør dette nemt.

De følgende fire procedurer hhv. henter og gemmer hhv. reelle tal og strengparametre. Parameter no angiver i alle tilfælde nummeret på parameteren.

getrealpar(no : byte; VAR tal : real)
henter reel parameter no og konverterer tallet til PolyPascal-format i tal.

putrealpar(no : byte; tal : real)
konverterer tal til RcComal80-format og gemmer det i adressen for parameter nummer no.

getstringpar(no : byte; VAR mystr : stringmax)
henter strengparameter nummer no og konverterer til PolyPascal-format i mystr.

putstringpar(no : byte; mystr : stringmax)
konverterer mystr til RcComal80-format og gemmer i adressen for parameter nummer no.

Eksempel
Funktionen musstatus i pakken MUS bruger proceduren putrealpar til at ændre sine REF-parametre. Anden parameteren dy ændres således:

```
(* tal indeholder dy på PolyPascal-format *)
putrealpar(2,tal);
```

Alle procedurer, der angives i pakkehovedet skal afsluttes med et kald af proceduren returnfar, der returnerer til RcComal80. En funktion afsluttes med et kald af proceduren return_realresult eller return_stringresult, der sætter funktionsresultatet op og returnerer til RcComal80. Alle tre procedurer er erklæret i POLYPAS.PAS

returnfar
afslutter kaldet af proceduren og returnerer til RcComal80.

return_realresult(tal : real)
afslutter et kald af en reel funktion, returnerer funktionsværdien og returnerer til RcComal80.

return_stringresult(mystr : stringmax)
afslutter et kald af en strengfunktion, returnerer funktionsværdien og returnerer til RcComal80.

Dokumentation for en pakke, en procedure eller en funktion angives med en adresse i pakkehovedet. Adressen udpeger et sted i PolyPascal-programmet, hvor der er erklæret en typedefineret strengkonstant. Dokumentationen kan udskrives med kommandoen DOCU.

Eksempel
Dokumentationen for pakken MUS er erklæret således (stringmax er en type erklæret i POLYPAS.PAS; stringmax = STRINGÆ255Å):

```
CONST
  dok_mus : stringmax =
    'brug af mus uden grafik';
```

12.3.3 Oversættelse af pakker

Pakker skrevet i PolyPascal laves som følger:

1) først laves en programfil med p-kommandoen i PolyPascal

2) så udføres denne programfil på styresystem-niveau

3) til sidst udføres programmet PASPAK med filnavnet som parameter

12.3.4 Fejlhåndtering

Initialiseringsrutinen i PolyPascal-pakke skal altid indeholde et kald af proceduren:

init_errorhandler

Når denne procedure er kaldt, vil eventuelle hårde fejl i PolyPascal (kørsels- eller I/O-fejl, som f.eks. division med 0) betyde, at der returneres til RcComal80 med fejlnummer sat til 190 (fejl i PolyPascal-pakke). Hvis fejlnummer 190 er sat vil RcComal80 stoppe programudførelsen og udskrive en fejlmeddelelse. Når fejlnummer 190 forekommer, er det ikke muligt at bruge pakken igen, før den er re-initialiseret (ved USE eller RUN).

Fejlnummer 190 betyder, at kald af procedurer på kommando-niveau og udførelse af kommandoen CON ikke er tilladt i RcComal80. Det er ikke tilladt at behandle fejlnummer 190 i en PROC-HANDLER.

Der er i inklude-filen på distributionsdisketten defineret en konstant **ppaserr=190.**

Det er ikke muligt at have sin egen fejlbehandlingsrutine i PolyPascal-pakken til behandling af hårde fejl.

Derimod er der mulighed for selv at tage vare på fejl, som hører til RcComal80-programmet, men som først opdages i selve PolyPascal-pakken.
Fra PolyPascal-pakken er der så mulighed for at kalde proceduren comalerror fra POLYPAS.PAS. Alle tal mellem 191 og 199 er reserveret til brug som fejlnumre i PolyPascal-pakker. Ved at ændre i GENERRM.CSV kan

man definere fejltekster svarende til numrene 191 til 199. Dette bør selvfølgelig beskrives i den tilhørende skriftlige dokumentation for pakken. Proceduren comalerror har følgende format:

comalerror(errorkode : integer);
sætter fejlnummeret i RcComal80, så RcComal80-programmet stopper, når der returneres, og udskriver en fejlmeddelelse.

Bemærk:
Under konvertering af tal og strenge fra RcComal80-format til PolyPascal-format vil fejlnummeret i RcComal80 blive sat til 191, f.eks. hvis tallet ikke kan konverteres pga. at RcComal80's talområde er større end PolyPascal's.
Der er i inklude-filen på distributionsdisketten defineret en konstant konverr=191.

12.3.5 Gode råd og advarsler

Cursorplacering

RcComal80 holder hele tiden rede på, hvor cursoren er placeret på skærmen. Hvis pakken ændrer på cursorplaceringen ved udskrift på eller indlæsning fra skærmen, vil cursorens placering være ændret, når der returneres til RcComal80; ofte vil cursoren stå oveni udskriften på skærmen.

Under alle omstændigheder bør det fremgå af dokumentationen for pakken, om problemet kan forekomme.

Filer

Hvis en pakke-rutine åbner en fil, skal pakken sørge for at lukke den igen, idet den ellers vil blokere en fælles diskettestation.

Tegnsæt

Hvis en pakke-rutine ændrer maskinens tegnsæt, skal pakken sørge for at reeablere det oprindelige tegnsæt (f.eks. ved exitrutinen).

Interruptvektorer

Hvis en pakke ændrer interruptvektorerne, skal pakken sørge for at reetablere dem (f.eks. i exitrutinen).

12.3.6 Procedurer til at lave pakkehoved

POLYPAS.PAS indeholder følgende procedurer til at lave pakkehovedet til en PolyPascal-pakke.

openfile(filnavn : navne)
åbner en pakkehovedfil, som hedder filnavn. De øvrige procedurer opererer på denne fil.

closefile
lukker pakkehovedfilen.

skriv_byte(tal : byte)
hjælpeprocedure, der udskriver en byte på filen.

skriv_navn(varnavn : navne)
procedure, der først udskriver en byte med længden af navnet, og dernæst de enkelte tegn i navnet.

skriv_offset(offset : integer)
udskriver parameteren i to bytes.

skriv_packtype
udskriver at pakketypen er PolyPascal.

skriv_versionsno
udskriver versionsnummer.

skriv_proc
bruges til at udskrive, at det er en procedure.

skriv_realfunc
bruges til at udskrive, at det er en reel funktion.

skriv_stringfunc
bruges til at udskrive, at det er en strengfunktion.

skriv_typeogdim(partype, pardim : byte)
skriver type og dimensionen af en parameter.

skriv_reserver
skriver adressen på de variable, der skal erklæres for at kunne behandle kald af rutiner i pakken.

12.3.7 Andre procedurer i POLYPAS.PAS

comalprgfunckey
programmerer funktionstasterne til standardværdierne, dvs. det samme som når RcComal80 starter op.

De følgende fire procedurer er hjælpeprocedurer til getrealpar, putrealpar, getstringpar og putstringpar (er beskrevet i 12.3.2). Disse hjælpeprocedurer er anvendelige ved avancerede pakker, som bruger tabeller som parametre, og behandles i afsnit 12.5.

henttal(segadr, ofsadr : integer; VAR tal : real)
henter tallet på RcComal80-format, som ligger på adressen de to første parametre angiver, og konverterer det til PolyPascal-format i tal.

gemtal(segadr, ofsadr : integer; tal : real)
konverterer tal til RcComal80-format og gemmer det i adressen, som segadr og ofsadr angiver.

hentstring(segadr, ofsadr : integer; VAR mystr : stringmax)
henter en streng parameter fra adressen, som segadr og ofsadr angiver, og konverterer til PolyPascal-format i mystr.

gemstring(segadr, ofsadr : integer; mystr : stringmax)
konverterer mystr til RcComal80-format og gemmer i adressen som segadr og ofsadr.

12.4 Programmering af assemblerpakker til RcComal80

Pakker i assembler til RcComal80 skal skrives til ASM86 eller RASM86. Dette afsnit forudsætter derfor kendskab til et af disse assemblersprog.

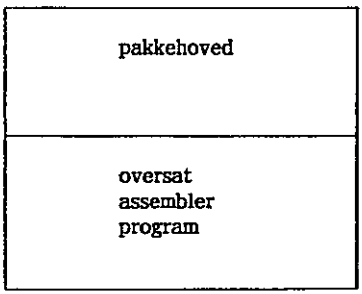
Programmering i assembler giver adgang til alle faciliteter i maskinen såsom styresystemkald og alle lagerceller. Dette giver risiko for "at slå benene væk under sig selv", således at maskinen må genstartes. Det er i modsætning til programmering i RcComal80, hvor man befinder sig i beskyttede omgivelser og bliver advaret, hvis man forsøger at lave noget galt. Det er sværere at finde fejl i et assemblerprogram, fordi programmet ofte fortsætter efter en fejl, så det er svært at se grunden til, at det gik galt.

Hvis man vil programmere i assembler på en PICCOLINE eller en Partner, er det nødvendigt at anskaffe CCP/M-86 Manual Set (ref. 3), som bl.a. indeholder "Programmer's Utilities Guide for CP/M-86", som dokumenterer ASM86. Desuden indeholder manualsættet "Concurrent CP/M-86 Programmer's Reference Guide", som beskriver de forskellige styresystemkald m.m. Denne er også i andre sammenhænge meget anvendelig. Det kan også være nødvendigt at anskaffe "Programmer's Guide" til den maskine, der bruges (ref. 5 og 6). Det skal bemærkes, at alle disse manualer er på engelsk.

De eksempler, der omtales i dette afsnit findes alle som kildetekster (filtype A86) på en af distributionsdisketterne. Dette gør det nemmere at lave sine egne pakker, idet det er muligt at genbruge en del fra disse eksempler.

12.4.1 Opbygning af en assemblerpakke

En assemblerpakke er opbygget på samme måde som en PolyPascal-pakke. Det er en fil, der består af et oversat assemblerprogram flettet sammen med et pakkehoved. Pakkehovedet beskriver de snitflader programmet har overfor RcComal80, når det bruges som en pakke.



Pakkehovedet definerer
- at det er en pakke skrevet i assembler
- for hver af de assemblerrutiner, der kan bruges som procedure eller funktion fra RcComal80: erklæring af proceduren/funktionen, adressen på koden for assemblerrutinen, adressen på dokumentationen hørende til assemblerrutinen.
  (gælder også for initialiserings- og exitrutinen)

De følgende afsnit beskriver de tekniske detaljer i, hvordan man laver assemblerpakken. Her skitseres på overordnet form, hvad der skal gøres for at lave en assemblerpakke, der kan bruges fra RcComal80:

Alle assemblerrutiner, der skal kunne kaldes fra RcComal80, skal starte med en label, så det er muligt at lokalisere rutinen. Hvis rutinen skal kunne kaldes som en procedure eller funktion med parametre, skal der tilføjes assemblerkode, som sørger for at overføre parametre fra og til RcComal80. Tilsvarende for et funktionsresultat. (Hvis rutinerne stadig skal kunne kaldes i assemblerprogrammet, skal der således være 2 versioner af dem).
Der findes på en af distributionsdiskettene en fil med en række nyttige hjælperutiner til overførelse af parametrene.

Alle assemblerrutiner, der skal kunne kaldes fra RcComal80, skal afsluttes med en returnering til RcComal80. Der findes en speciel hjælpeprocedure til dette formål.

Der skal defineres et pakkehoved forrest i assemblerpakkens codesegment. Pakkehovedet definerer, hvordan assemblerrutinerne ser ud fra RcComal80.

Når ovennævnte ændringer er udført, skal assemblerprogrammet oversættes og derefter skal der ud fra det oversatte program genereres en CMD-fil. Der findes på distributionsdisketten et submitprogram til at udføre dette.

I de følgende afsnit kommer en mere detaljeret beskrivelse af, hvad der skal gøres.

12.4.2 Konventioner for en assemblerpakke

Der er nogle konventioner, som et assemblerprogram skal overholde, for at det kan bruges som en pakke af RcComal80:

1) der skal forrest i codesegmentet defineres et pakkehoved, der bl.a. angiver, hvilke procedurer og funktioner pakken stiller til rådighed

2) i datasegmentet skal der reserveres plads til basepage og 4 ord, der skal bruges ved kald af en procedure eller funktion i pakken

3) de rutiner fra assemblerpakken, der skal bruges som procedurer og funktioner fra RcComal80, skal selv sørge for at overføre parametre mellem pakken og RcComal80

4) de rutiner fra assemblerpakken, der skal bruges som procedurer og funktioner fra RcComal80, skal selv sørge for at returnere til RcComal80

For at gøre det nemmere at overholde disse konventioner findes en fil ASSEMBL.A86 på en af distributionsdisketterne.

Denne fil skal inkluderes i assemblerprogrammet, som skal være en pakke. En udskrift af ASSEMBL.A86 findes i appendix H. ASSEMBL.A86 indeholder:
- konstanter, der gør det nemmere at lave pakkehovedet
- kode, der sørger for at reservere de nødvendige ord i datasegmentet
- rutiner, der gør det let at flytte parameterværdier fra RcComal80 til pakken og tilbage
- en rutine, der sørger for at returnere til RcComal80 fra en pakke
- en samling af andre anvendelige rutiner, som kan kaldes fra assemblerkoden (beskrives nedenfor)

Pakkehovedet skal stå forrest i assemblerpakkens codesegment og består af tre dele, hvor den midterste del definerer de procedurer og funktioner, pakken stiller til rådighed.

Eksempel
Pakken PROGRAM har en procedure kommando, der gør det muligt at kalde et program, der findes på en fil på disketten. Pakkehovedet for pakken PROGRAM ser således ud (filen findes på en af distributionsdisketterne).

```
CSEG
VER       DW    VERSNO              ; versionsnummer
PACKTYPE  DW    ASSEMBLER           ; assemblerpakke
DOC       DW    OFFSET DOKPROGRAM; ingen dokumentation
INIT      DW    OFFSET FINITO       ; offset initrutine
EXIT      DW    0                   ; ingen exitrutine

                                    ; procedure kommando(comstr$);
          DB    8,'KOMMANDO'        ; procedurenavnet
          DB    PROC                ; procedure
          DW    0                   ; ingen dokumentation
          DW    OFFSET XCOM         ; offset for proceduren
          DB    1                   ; 1 parameter
          DB    6, 'COMSTR'         ; parameternavn
          DB    VALUE+STRING,0      ; simple_string
```

```
DB        0                 ; ikke flere procedurer
DW        OFFSET RESERVER   ; offset for de 4 ord i dataseg
DW        ASSEMBLER         ; afslutning af pakkens hoved
INCLUDE   ASSEMBL.A86       ;
```

Bemærk: DW afsætter og definerer ord, mens DB afsætter og definerer bytes.

Navne foran DW (f.eks. VER og PACKTYPE) og resten af linierne efter semikolon virker som kommentarer og er kun med for at lette læseligheden.

De første tre linier af pakkehovedet vil være de samme for alle assemblerpakker. De sørger for at pakkehovedet skrives forrest i code-segmentet, giver pakken et versionsnummer, og fortæller at pakken er skrevet i assembler. Versionsnummeret og pakketypen skal bruges, når pakken hentes ind i RcComal80.

Den fjerde linie angiver, om der i pakken er noget dokumentation, der kan udskrives med kommandoen DOCU. Det beskrives sidst i dette afsnit, hvorledes dokumentationen laves.

Den femte linie angiver adressen for initialiseringsrutinen, der kaldes automatisk, når det angives i et RcComal80-program, at en pakke skal benyttes (se USE i kapitel 11). Der skal altid være en initialiseringsrutine. Initialiseringsrutinen kan ikke have parametre.

Linie seks angiver adressen på exitrutinen. Exitrutinen kaldes automatisk, når en pakke fjernes fra lageret (sker ved kommandoerne DISCARD, NEW og LOAD). Exitrutinen kan ikke have parametre. Der behøver (i modsætning til initialiseringsrutinen) ikke være en exitrutine (i så fald sættes adressen bare til nul).

Den midterste del af pakkehovedet definerer de procedurer og funktioner, som pakken stiller til rådighed. I pakken PROGRAM er der kun en procedure (denne hedder KOMMANDO).

For hver procedure eller funktion angives:

- navnet (DB 8,'KOMMANDO') (antal tegn og de enkelte tegn)
- typen for rutinen. (her er det en procedure)
- adresse på en eventuel dokumentation for proceduren
- adresse på rutinen
- antal parametre (her 1)
- de enkelte parametre

En parameter beskrives ved to linier:
- navnet (DB 6,'COMSTR')
- typen og dimensionen af parameteren (DB VALUE+STRING,0)

Til at beskrive typen af en parameter, er der defineret nogle konstanter i filen ASSEMBL.A86:

VALUE	betyder VALUE-parameter
REF	betyder REF-parameter
REAL	betyder en numerisk parameter
STRING	betyder en streng parameter

Typen af parameteren er et tal, der angives som en sum af en af de to første (VALUE eller REF) og en af de to sidste (REAL eller STRING).

Når parameteren ikke er en tabel, er dimensionen 0. Hvis parameteren er en tabel, skal parametertypen være REF (den må ikke være VALUE).

En funktion har samme format bortset fra at typen på rutinen skal være enten REALFUNC eller STRINGFUNC.

De sidste fire linier i hovedet er de samme for alle assemblerpakker.

Når ASSEMBL.A86 inkluderes i assemblerpakken, bliver der reserveret den nødvendige plads i datasegmentet for pakken (se punkt 2 i starten af dette afsnit).

Alle procedurer og funktioner, der angives i pakkehovedet, skal selv hente eventuelle parameterværdier fra RcComal80 og aflevere REF-parameterværdier og funktionsresultater tilbage til RcComal80. Der findes fire rutiner i ASSEMBL.A86, der gør dette nemt:

REALPAR henter en reel parameter fra RcComal80, og konverterer værdien til et 16 bits heltal, der kan være i et register.

Registre ved kald:
CL indeholder nummeret på parameteren

Registre efter kald:
BX indeholder heltallet
Der er kun garanti for at registrene CS, DS og SS har bevaret deres værdi.

Fejlmuligheder:
Under konverteringen kan der opstå RcComal80-fejl 101 "ULOVLIG HELTALSVÆRDI".

Eksempel
I pakken BITOP flytter funktionen BITAND sine to reelle parametre over i to ord (bitrep1 og bitrep2) i pakkens datasegment:

```
MOV     CL,1
CALL    REALPAR
MOV     BITREP1,BX
MOV     CL,2
CALL    REALPAR
MOV     BITREP2,BX
```

STRPAR henter en strengparameter fra RcComal80.

Registre ved kald:
CL indeholder nummeret på parameteren
DI indeholder adressen, hvor tegnene skal ligge

Registre efter kald:
Der er kun garanti for at registrene CS, DS og SS har bevaret deres værdi.

Fejlmuligheder:
ingen

Eksempel
I pakken program flyttes strengen fra parameter 1 fra RcComal80's datasegment over i pakkens datasegment:

```
MOV     CL,1
MOV     DI,OFFSET CLBUF1
CALL    STRPAR
```

REALRES konverterer et 16-bits heltal til RcComal80-format og returnerer det som funktionsværdi til RcComal80. Efter kaldet må registerene AX og CX ikke ændres, inden der returneres til RcComal80).

Registre ved kald:
BX indeholder det 16-bits heltal, der skal konverteres

Registre efter kald:
AX indeholder segment for tallet på RcComal80-format
CX indeholder offset for tallet på RcComal80-format
Der er kun garanti for at registrene CS, DS og SS har bevaret deres værdi.

Fejlmuligheder:
ingen

Eksempel
Alle funktioner i pakken BITOP afsluttes med følgende to ordrer, der returnerer funktionsværdi og returenerer til RcComal80:

```
CALL    REALRES
JMP     FINITO
```

PUTREALPAR konverterer et tal og returnerer det til en REF-parameter.

Registre ved kald:
BX indeholder det 16-bits heltal, der skal konverteres
CL indeholder nummeret på REF-parameteren

Registre efter kald:
Der er kun garanti for at registrene CS, DS og SS har bevaret deres værdi.

Fejlmuligheder:
ingen

Eksempel
Proceduren pin i pakken PORTIO afsluttes med følgende to ordrer, der ændrer parameterværdien for parameteren d (idet register BX inden disse to ordrer, har fået den rette værdi):

```
MOV     CL,2
CALL    PUTREALPAR
```

Alle procedurer og funktioner, der angives i pakkehovedet, og initialiseringsrutinen og en eventuel exitrutine skal afsluttes specielt, så der returneres til RcComal80 programmet. Rutinen FINITO i ASSEMBL.A86 gør dette nemt.

FINITO sørger for at returnere til RcComal80. FINITO sørger for at fjerne de ting, pakken har stakket på assemblerstakken, men ikke har fjernet igen.

Registre ved kald:

DS skal indeholde adressen på datasegmentet for pakken
AX skal indeholde segment for et funktionsresultat, hvis det er en funktion, der returneres fra
CX skal indeholde offset for et funktionsresultat, hvis det er en funktion, der returneres fra
DX skal desuden indeholde længden af strengen, hvis det er en strengfunktion

Registre efter kald:

Returnerer ikke til pakken

Fejlmuligheder:

ingen

Eksempel

I pakken PROGRAM er initialiseringsrutinen lig FINITO, dvs. der udføres ingen initialisering, men springes direkte tilbage til RcComal80 (men der skal være en initialiseringsrutine).

Rutinen ERROR kan sætte RcComal80's fejlnummer, så udførelsen af RcComal80 programmet stopper, når der returneres. RcComal80 vil så udskrive fejlnummeret eller en fejlmeddelelse.

Registre ved kald:

AX indeholder fejlnummeret

Registre efter kald:

Der er kun garanti for at registrene CS, DS og SS har bevaret deres værdi.

Fejlmuligheder:

ingen

Eksempel

I pakken PROGRAM kaldes rutinen ERROR, hvis der opstår fejl, når proceduren kommando vil udføre det program, som dets parameter angiver (det kan f.eks. skyldes at parameteren ikke er et korrekt filnavn, eller at der ikke er plads nok):

```
MOV     AX,ASSERR       ;
CALL    ERROR           ; error(assemblerpakke);
```

ASSERR er en konstant (180). Den er defineret i ASSEMBL.A86, og kan bruges ved fejl i assemblerpakken. Desuden er alle tal mellem 181 og 189 reserveret til brug som fejlnumre i assemblerpakker. Ved at ændre i GENERRM.CSV kan man definere fejltekster svarende til numrene 181 til 189. Specielle fejltekster defineret i en pakke bør selvfølgelig beskrives i den tilhørende skriftlige dokumentation for pakken.

COMALRUT kan kalde de RcComal80 rutiner, der er til rådighed for assemblerpakken. Disse rutiner beskrives nærmere i afsnit 12.4.5.

Registre ved kald:

BX indeholder nummeret på rutinen (0 til 32)
SI indeholder første parameter, hvis rutinen har mindst en parameter
DI indeholder anden parameter, hvis rutinen har to parametre

Registre efter kald:

SI indeholder resultatet, hvis rutinen returnerer et resultat
Der er kun garanti for at registrene CS, DS og SS har bevaret deres værdi.

Fejlmuligheder:

Hvis der opstår fejl under udførelsen, sættes fejlnummeret automatisk i RcComal80. Disse fejlnumre vil være et af de allerede eksisterende i RcComal80. Se de enkelte rutiner i afsnit 12.4.5 angående hvilke fejl, der kan forekomme.

Eksempel
I pakken PROGRAM kaldes RcComal80 rutine nummer 31, som sørger for at programmere funktionstasterne. Rutinen har ingen parametre.

```
MOV     BX,31           ;
CALL    COMALRUT        ; comal_program_funktionkeys;
```

Dokumentation for en pakke, en procedure eller en funktion angives med en adresse i pakkehovedet. Adressen udpeger et sted i codesegmentet, hvor der er et ord, der angiver antallet af tegn i dokumentationen, efterfulgt af en række bytes med selve dokumentationen.

Eksempel
Pakken PROGRAM har følgende dokumentation:

```
DOKPROGRAM DW       LENGTH DPROGRAM
DPROGRAM   DB       'kan kalde programmer'
```

Dokumentationen kan udskrives fra RcComal80 med kommandoen DOCU.

12.4.3 Oversættelse af pakker

Pakker skrevet i ASM86 skal oversættes med ASM86, og derefter skal der genereres en CMD-fil v.hj.a. GENCMD. Til sidst bør CMD-filen omdøbes til at have filtypen PCK.

Til at gøre alt dette findes en SUBMIT-fil ASSPAK.SUB, så en oversættelse af assemblerprogrammet PROGRAM.A86 sker ved

```
SUBMIT ASSPAK PROGRAM
```

Pakker kan også oversættes med RASM86, og der kan genereres en CMD-fil med LINK86, hvis du har disse to programmer til rådighed.

ASM86 og GENCMD ligger på en af distributionsdisketterne for maskinen. RASM86 og LINK86 er en del af SW1603 (DR assembler plus tools, ref. 9).

12.4.4 Gode råd og advarsler

Stakplads

Når en pakkeprocedure eller -funktion kaldes vil den bruge samme assemblerstak, som den RcComal80 bruger, hvis der ikke er defineret et staksegment i pakken. Hvis der bruges meget stakplads, kan det blive nødvendigt, at pakken definerer sit eget staksegment. RcComal80-fortolkeren sørger så for at skifte til dette staksegment ved kaldet af en rutine i pakken og skifte tilbage til RcComal80's staksegment efter returneringen til RcComal80.

Cursorplacering

RcComal80 holder hele tiden rede på, hvor cursoren er placeret på skærmen. Hvis pakken ændrer på cursorplaceringen ved udskrift på eller indlæsning fra skærmen, vil cursorens placering være ændret, når der returneres til RcComal80; ofte vil cursoren stå oveni udskriften på skærmen.

Under alle omstændigheder bør det fremgå af dokumentationen for pakken, om problemet kan forekomme.

Filer

Hvis en pakke-rutine åbner en fil, skal pakken huske at lukke den igen, idet den ellers vil blokere en fælles diskettestation.

Tegnsæt

Hvis en pakke-rutine ændrer maskinens tegnsæt, skal pakken sørge for, at reetablere det oprindelige tegnsæt (f.eks. exitrutinen).

Interruptvektorer

Hvis en pakke ændrer interruptvektorerne, skal pakken sørge for at reetablere dem (f.eks. i exitrutinen).

12.4.5 RcComal80-rutiner

Dette afsnit beskriver de RcComal80-rutiner, som det er muligt at kalde fra en assemblerpakke. Rutinerne kaldes med rutinen COMALRUT (se 12.4.2) med parametre som beskrevet nedenfor.

Registre ved kald:
BX indeholder nummeret på rutinen (0 til 32)
SI indeholder første parameter, hvis rutinen har mindst en parameter
DI indeholder anden parameter, hvis rutinen har to parametre

Registre efter kald:
SI indeholder resultatet, hvis rutinen returnerer et resultat
Der er kun garanti for at registrene CS, DS og SS har bevaret deres værdi.

Fejlmuligheder:
Hvis der opstår fejl under udførelsen, sættes fejlnummeret automatisk i RcComal80. Disse fejlnumre vil være et af de allerede eksisterende i RcComal80. Se de enkelte rutiner nedenfor angående hvilke fejl, der kan forekomme.

Parametrene SI og DI og et eventuelt funktionsresultat i SI vil, hvis de repræsenterer et tal på RcComal80-format, være en adresse i RcComal80's datasegment (hvor de 8 bytes for tallet ligger). Såfremt de repræsenterer et 16-bits heltal, indeholder de 16-bits heltallet.

Rutine nr	og navn	Beskrivelse
0	FLOAT	Konverterer det 16 bits heltal, som SI indeholder til RcComal80-format. Efter kaldet vil SI pege på værdien på RcComal80-format.
1	FIX	Konverterer det tal på RcComal80-format, som SI peger på til 16 bits heltal. Efter kaldet vil BX indeholde det 16 bits heltal. Fejlkoden 101, som betyder at tallet er for stort eller for lille til at blive konverteret, kan blive sat.
2	STR	Konverterer det tal SI peger på til en streng af tegn, så tallet let kan skrives ud. Efter kaldet indeholder BX længden af strengen og DX peger på det første tegn.
3	ADD	Adderer de to tal på RcComal80-format, som SI og DI peger på. Tallene, som SI og DI peger på ændres ikke, men SI ændres så den efter kaldet peger på et tal, der er summen af de to tal. Fejlkoden 106, hvis der er arithmetisk overløb, kan blive sat.
4	SUB	Subtraktion, registre bruges som ved ADD.
5	MULT	Multiplikation, registre bruges som ved ADD.
6	DIVI	Division, registre bruges som ved ADD. Desuden er der mulighed for fejlkode 104 (Division med 0).
7	MOD	Heltalsmodulus, registre bruges som ved ADD.
8	DIV	Heltalsdivision, registre bruges som ved ADD. Desuden er der mulighed for fejlkode 104 (Division med 0).
9	POWER	Potensopløftning, registre bruges som ved ADD.
10	EQ	Sammenligning, registre bruges som ved ADD. Efter kaldet peger SI på tallet 1 på RcComal80-format, hvis de to tal er ens, ellers peger SI på tallet 0.
11	NE	Ikke-ens, ellers som EQ.
12	LT	Mindre-end, ellers som EQ.
13	LE	Mindre-end-eller-lig-med, ellers som EQ.
14	GT	Større-end, ellers som EQ.

15 GE Større-end-eller-lig-med, ellers som EQ.

16 OR Logisk or, registre bruges som ved ADD. Efter kaldet peger SI på tallet 1 på RcComal80-format, hvis bare et af tallene er forskellige fra 0, ellers peger SI på tallet 0.

17 AND Logisk and, registre bruges som ved ADD. Efter kaldet peger SI på tallet 1 på RcComal80-format, hvis begge tal er forskellige fra 0, ellers peger SI på tallet 0.

18 NOT Logisk not. Ved kaldet peger SI på parameteren. Efter kaldet peger SI på tallet 1 på RcComal80-format, hvis parameteren er lig 0, ellers peger SI på tallet 0.

19 ABS Absolut værdi. SI peger på parameteren. Efter kaldet peger SI på den absolutte værdi.

20 SIGN Fortegnsværdi. SI peger på parameteren. Efter kaldet peger SI på tallet 1 på RcComal80-format, hvis parameteren er positiv, tallet 0 hvis parameteren er 0, og tallet -1 hvis parameteren er negativ.

21 NEG Negation, ellers som ABS.

22 INT Den hele del, det nærmeste heltal der er mindre end eller lig parameteren. Ellers som ABS.

23 SQR Kvadratrod, ellers som ABS. Kan returnere fejlkode 103 (Kvadratrod af negativt tal).

24 EXP Den naturlige exponentialfunktion, ellers som ABS.

25 LOG Den naturlige logaritmefunktion, ellers som ABS. Kan returnere fejlkode 102 (Log til negativt tal).

26 TAN Tangensfunktion, ellers som ABS. Regner med radianer.

27 COS Cosinusfunktion, ellers som TAN.

28 SIN Sinusfunktion, ellers som TAN.

29 ATN Arctangensfunktion (den omvendte funktion til tangens), ellers som TAN.

30 PI Efter kaldet peger SI på tallet pi. Ingen parametre.

31 FUNKKEY Programmerer funktionstaster, som de skal være i RcComal80.

32 ERR Sætter RcComal80's fejlnummer til indholdet af register SI. Dette kan også laves ved rutinen ERROR (se afsnit 12.4.2), som har AX som parameter.

Bemærk: ved funktioner, der opererer på tal på RcComal80-format vil de tal, som SI og DI peger på, ikke blive ændret, men SI vil blive ændret til at pege på det sted, hvor resultatet ligger. Alle disse tal vil ligge i RcComal80's datasegment.

Ved kald af flere funktioner efter hinanden kan det være nødvendigt at gemme et mellemresultat. Det gøres ved at flytte værdien midlertidigt over i pakkens datasegment, da det ellers kan blive ødelagt ved næste funktionskald. Efter kaldet skal værdien flyttes tilbage til RcComal80's datasegment. Adressen, der har offset 8 mindre end den adresse, der returneres i SI i rutinerne 0 og 2 til 30 kan bruges til dette formål.

Eksempel
sin(par1)+cos(par2) kan udregnes således

```
MOV     BP,RESERVER+6
MOV     SI,6ÆBPÅ ; SI:= adresse på par1
MOV     BX,28    ;
CALL    COMALRUT; sin
CALL    TOPACK   ; flyt resultatet over i pakkens
                 ;          datasegment
MOV     BP,RESERVER+6
MOV     SI,8ÆBPÅ ; SI:= adresse på par2
MOV     BX,27    ;
CALL    COMALRUT; cos
PUSH    SI       ;
MOV     DI,SI    ;
SUB     DI,8     ;
PUSH    DI       ;
CALL    FROMPACK; flyt resultatet af sin-kaldet fra
                 ; pakkens datasegment til RcComal80's
                 ; datasegment
POP     DI       ;
POP     SI       ;
MOV     BX,3     ;
CALL    COMALRUT; add
```

hvor TOPACK og FROMPACK kan laves således:

```
TOPACK:
    PUSH    DS          ;
    POP     ES          ;
    MOV     DI,OFFSET GEMMESTED
    MOV     CX,8        ;
    MOV     BP, RESERVER+6
    MOV     DS,4ÆBPÅ    ;
    REP     MOVSB       ; movebytes(RcComal80's datasegm adr SI,
    PUSH    ES          ;           pakkens datasegm adr gemmested,
    POP     DS          ;           8 bytes);
    RET                 ;

FROMPACK:
    MOV     BP, RESERVER+6
    MOV     ES,4ÆBPÅ    ;
    MOV     SI,OFFSET GEMMESTED
    MOV     CX,8        ; movebytes(pakkens datasegm gemmested,
    REP     MOVSB       ;           RcComal80's datasegm adr DI,
    RET                 ;           8 bytes);
```

Eksempel
exp(sin(par1)) kan udregnes som følger (her er det ikke nødvendigt at flytte resultater, da der kun er en resultat-værdi af gangen):

```
    MOV     BP,RESERVER+6
    MOV     SI,6ÆBPÅ ; SI:= adresse på par1
    MOV     BX,28
    CALL    COMALRUT; sin
    MOV     BX,24
    CALL    COMALRUT; exp
```

12.5 Programmering af avancerede pakker til RcComal80

Når man laver pakker i PolyPascal eller assembler, kan det ske at beskrivelserne i de foregående afsnit, ikke er fyldestgørende. Dette kan f.eks. være tilfældet, hvis man har tabeller, som parametre. Dette kapitel indeholder derfor den fuldstændige beskrivelse af, hvordan pakker skrevet i PolyPascal og assembler fungerer. Det vil være en fordele at sætte sig ind i koden i de to inklude-filer POLYPAS.PAS og ASSEMBL.A86, inden man vover sig ud i, at lave sine egne rutiner til overføring af parametre m.v.

12.5.1 Ved kald af pakkerutine

Når en af pakkens procedurer eller funktioner kaldes skiftes til pakkens codesegment og datasegment. Hvis pakken har et staksegment, skiftes også til dette, ellers bruges det samme staksegment som RcComal80 bruger.

Når en af procedurerne eller funktionerne kaldes er stakkens udseende (Bemærk: når stakken vokser bliver staktop-pointeren mindre):

adresse relativt til staktoppen på kald-tidspunktet (i bytes)	indhold
stacktop	offset for returadresse
stacktop+2	segment for returadresse
stacktop+4	adresse på reserverede ord i pakkens datasegment
stacktop+6	offset for tabel af RcComal80-rutiner
stacktop+8	segment for tabel af RcComal80-rutiner
stacktop+10	segment for parametre
stacktop+12	offset for evt. første parameter
stacktop+14	offset for evt. anden parameter
stacktop+16	...

De to øverste ord skal bruges, når der skal returneres til RcComal80 fra pakken.

Det næste ord (indholdet af stacktop+4) udpeger fire reserverede ord i pakkens datasegment. De to første af disse bruges ved selve kaldet af proceduren eller funktionen i pakken. De to sidste er pointere ind i stakområdet. Det tredie ord er således en pegepind, som peger på det stakord, der er efter den sidste parameter (hvis der er to parametre, er indholdet således stacktop+16). Det fjerde ord indeholder stacktop+6, dvs. det peger på det ord, der indeholder offset for RcComal80-rutiner. Ved hjælp af dette fjerde ord er det altid muligt (også selv om pakkeproceduren eller -funktionen har fyldt noget mere på stakken) at finde returadressen på stakken, at kalde en af de RcComal80-rutiner der er til rådighed, og at hente eller gemme en parameter.

De to næste ord på stakken indeholder adgang til disse forskellige RcComal80-rutiner. Disse rutiner er mest anvendelige i assemblerpakker og er beskrevet i afsnit 12.4.5. Nogle af dem er også gjort direkte tilgængelige i PolyPascal-pakker, f.eks. rutine 31 og 32, som kan kaldes comalprgfunckey og comalerror. De øvrige kan også kaldes fra PolyPascal-pakker, men de indeholder operationer, der allerede er i PolyPascal, og det kræver, at der laves tilsvarende "krumspring", som i comalprgfunckey og comalerror.

De resterende ord gør det muligt at få fat i parametrene. I inklude-filerne POLYPAS.PAS og ASSEMBL.A86 findes rutiner til at hente og gemme simple variable (ikke tabeller). Hvis man ønsker at anvende tabeller som parametre, er det nødvendigt, at lave sine egne rutiner, der henter og gemmer disse. For at lave disse rutiner skal man kende formatet for, hvordan RcComal80 gemmer tal, strenge og tabeller. Desuden skal der laves rutiner, der konverterer disse til det format, man ønsker. Til denne konvertering kan man tage udgangspunkt i de rutiner, der findes i inklude-filerne. Det næste afsnit beskriver formatet for tal, strenge og tabeller i RcComal80, og dernæst følger et afsnit med nogle eksempler.

12.5.2 Format for tal, strenge og tabeller

Reelle tal

Reelle tal fylder 8 bytes på RcComal80-format. I samlingen af RcComal80-rutiner, der kan benyttes fra assemblerpakker, findes der rutiner til at regne med reelle tal på RcComal80-format og rutiner til at konvertere mellem RcComal80-format og 16 bits heltal (se afsnit 12.4.5).

De 8 bytes anvendes som følger (hver linie gælder for en byte):

fortegn	1.ciffer
2.ciffer	3.ciffer
4.ciffer	5.ciffer
6.ciffer	7.ciffer
8.ciffer	9.ciffer
10.ciffer	11.ciffer
12.ciffer	13.ciffer
exponent	

Fortegn og de 13 cifre fylder hver en nibble, dvs. fire bits. Fortegnet er negativt, hvis den øverste bit (den længst til venstre) i nibblen er 1. De enkelte cifre er gemt på BCD-format, dvs. værdien af nibblen er cifferværdien. Exponenten er en hel byte, der angiver exponenten relativt til 128, dvs. en exponent på 128 svarer til, at 1.ciffer står lige efter kommaet.

Eksempel
-987 gemmes som:

1000	1001
1000	0111
0000	0000
0000	0000
0000	0000
0000	0000
0000	0000
1000 0011	

hvor 10000011 binært er lig 131, dvs. 128+3.

Strengparametre

I RcComal80 ligger en streng på følgende format (hver linie gælder for en byte, dvs. længderne fylder hver to bytes (et ord) og de enkelte tegn en byte):

maksimale_længde
aktuelle_længde
første tegn
andet tegn
...
...
sidste tegn

Dvs. en streng fylder altid maksimale_længde + 4 bytes.

Eksempel
Hvis der i RcComal80 er skrevet følgende program:

```
0010 DIM str$ OF 10
0020 str$ :="ABEKAT"
0030 ...
```

Efter udførelsen af linie 0020 er værdien for str$ gemt således:

10
6
A
B
E
K
A
T
udefineret
udefineret
udefineret
udefineret

De to tal er gemt som 16 bits heltal. Tegnene er gemt ved deres ASCII-værdi. Bytes markeret med udefineret indeholder værdier, som ikke har nogen mening.

Tabel parametre

En tabel parameter består af en række simple elementer, som enten er tal eller strenge. Elementerne er gemt, som beskrevet i de to foregående afsnit. Udover denne række af simple værdier (de aktuelle data), er det nødvendigt at have en beskrivelse af selve dimensioneringen af tabellen (en såkaldt dope-vektor). Værdien af en tabel er en pegepind til et område, der består af to ord, der udpeger hhv. dopevektor og de aktuelle data:

reference til dope-vektor
reference til aktuelle data

Begge referencer fylder et ord (to bytes).

En dope-vektor har følgende format:

no. of dimensions
max index n'te dimension
min index n'te dimension
"size" n'te dimension
...
max index første dimension
min index første dimension
"size" første dimension

Alle felter undtagen det første fylder et ord (to bytes). no. of dimensions fylder kun en byte.

Se eksemplet nedenfor vedrørende "size".

Data for en tabel ligger grupperet på følgende måde: først ligger alle tabelindgange, hvor første index er "min index for første dimension", dernæst alle tabelindgange hvor første index er "min index for første dimension" + 1, osv. og til sidst alle tabelindgange hvor første index er "max index for første dimension".

Indenfor hver gruppe er de grupperet efter andet index, hvis der er flere dimensioner osv.

Eksempel

```
DIM a(1:7, 1:5)          // taltabel med to dimensioner
```

Data for a (hvert element fylder 8 bytes, da det er en taltabel):

```
a(1,1)
a(1,2)
a(1,3)
a(1,4)
a(1,5)
a(2,1)
a(2,2)
...
...
a(7,4)
a(7,5)
```

Dope-vektor for a

2	antal dimensioner
5	max index anden dimension
1	min index anden dimension
8	"size" for elementer
7	max index første dimension
1	min index første dimension
40	"size" for gruppe af elementer med samme første index

"Size" for første dimension er 40, da gruppen af elementer med ens første index er 5 elementer a 8 bytes lig med 40 bytes. Det betyder, at gruppen af elementer med første index lig 4 starter (4-1)*40 = 120 bytes efter starten på data.

"Size" for anden dimension er 8, da elementerne indenfor en gruppe med samme andet index fylder 8 bytes, da de er tal.

Teksttabeller ligger på samme måde, men basiselementerne ligger på den måde, der er beskrevet i forrige afsnit.

12.5.3 Eksempler

De to følgende eksempler beskriver, hvorledes det er muligt at hente elementet a(i,j) fra eksemplet i forrige afsnit i hhv. en PolyPascal-pakke og en assemblerpakke. I begge tilfælde skal man først beregne adressen på elementet, og derefter konvertere tallet til hhv. PolyPascal-format eller assemblerformat.

Eksempel
Dette eksempel beskriver, hvorledes elementet a(i,j) hentes i en PolyPascal-pakke, idet a er parameter nummer no, og resultatet skal returneres i variablen tal:

```
stackadr := memwÆ seg(reserver) : ofs(reserver)+6 A;
comaldataseg := memwÆ sseg : stackadr+4 A;
dataadr := memwÆ sseg : stackadr+4+2*no A;
dopevektor := memwÆ comaldataseg : dataadr A;
aktueldata := memwÆ comaldataseg : dataadr+2 A;
IF memÆ comaldataseg : dopevektor A <> 2 THEN
          myerror('dimension passer ikke');

dopevektor:=dopevektor+1;
IF (j > memwÆ comaldataseg : dopevektor A) OR
   (j < memwÆ comaldataseg : dopevektor+2 A) THEN
          myerror('index fejl');
secondindex:=    (j-memwÆ    comaldataseg    :    dopevektor+2    A) *
                      memwÆ comaldataseg : dopevektor+4 A;

dopevektor:=dopevektor+6;
IF (i > memwÆ comaldataseg : dopevektor A) OR
   (i < memwÆ comaldataseg : dopevektor+2 A) THEN
          myerror('index fejl');
firstindex:=     (i-memwÆ     comaldataseg     :     dopevektor+2     A) *
                      memwÆ comaldataseg : dopevektor+4 A;

elementadr := aktueldata+firstindex+secondindex;
henttal(comaldataseg, elementadr, tal);
```

Hvor procedure myerror er defineret således:

```
procedure myerror (tt: stringmax);
begin
  writeln(tt);
  comalerror(192);
end;
```

Eksempel
Dette eksempel beskriver, hvorledes elementet a(i,j) hentes i en assemblerpakke, idet parameter nummer for a er i AL, og i og j ligger i registrene CX og DX, og resultat skal returneres i register BX:

```
PUSH    DS                  ;
MOV     BP, RESERVER+6
MOV     DS,4ÆBPÅ            ; DS:=Comal's datasegment
XOR     AH,AH               ;
ADD     AX,AX               ;
ADD     BP,AX               ;
MOV     BX,4ÆBPÅ            ; offset for a
PUSH    BX                  ;
MOV     BX,ÆBXÅ             ; dopevektor
MOV     AL,ÆBXÅ             ;
CMP     AL,2                ; IF dim = 2 THEN
JNE     MYERROR2            ;
INC     BX                  ;
MOV     AX,ÆBXÅ             ;
CMP     DX,AX               ; IF j <= max_2_index THEN
JA      MYERROR2            ;
MOV     AX,2ÆBXÅ            ;
CMP     DX,AX               ; IF j >= mim_2_index THEN
JB      MYERROR2            ;
SUB     DX,AX               ;
MOV     AX,4ÆBXÅ            ;
MUL     DX                  ; s2 := (j-mim_2_index)*size_2
PUSH    AX                  ;
ADD     BX,6                ;
MOV     AX,ÆBXÅ             ;
CMP     CX,AX               ; IF i <= max_1_index THEN
JA      MYERROR3            ;
MOV     AX,2ÆBXÅ            ;
CMP     CX,AX               ; IF i >= min_1_index THEN
JB      MYERROR3            ;
SUB     CX,AX               ;
MOV     AX,4ÆBXÅ            ;
MUL     CX                  ; s1 := (i-min_1_index)*size_1
```

```
    POP     CX              ;
    ADD     AX,CX           ; s := s1+s2
    POP     BX              ;
    MOV     CX,2ÆBXÅ        ;
    ADD     AX,CX           ; s := s + dataadr_for_a
    MOV     SI,AX           ;
    MOV     BX,1            ;
    CALL    COMALRUT        ; fix(s)
    POP     DS              ;
```

Hvor de 2 specielle error-rutiner er defineret således:

```
MYERROR3:
    POP     CX
MYERROR2:
    POP     CX
    POP     DS
    MOV     AX,181
    CALL    ERROR
    JMP     FINITO
```

13. RcComal80 nøgleord

Dette afsnit skal betragtes som et referenceafsnit, som beskriver alle nøgleordene i RcComal80, dvs. sætninger, kommandoer, funktioner, strukturer og operatorer. Dog er de almindelige regneoperatorer (+, -, *, / og ^) og sammenligningsoperatorerne (=, <>, <, >, <= og >=) ikke beskrevet.

Beskrivelsen af nøgleordene har følgende format:

13.x RcComal80 nøgleord

Type
Format
Operator prioritet
Anvendelse
Virkemåde
Bemærkninger
Eksempler

hvor

x : afsnitsnummeret

RcComal80 nøgleord : et eller flere reserverede RcComal80 ord

Type : om nøgleordet er en sætning, kommando, struktur, funktion eller operator

Format : den formaliserede syntaks for nøgleordet. Den følger følgende regler:

- Store bogstaver skal indtastes direkte
- Bløde parenteser () skal indtastes direkte
- Krøllede parenteser { } giver valgfrihed mellem de opskrevne muligheder
- Kantede parenteser [] angiver, at det indklammede ikke skal, men kan indtastes
- Tre punktummer ... angiver, at det foregående argument kan gentages
- Symboler der er understregede beskrives senere i afsnittet

Operator prioritet	:	et tal, der angiver i hvilken rækkefølge operatorerne udføres (se afsnit 4.2.1 og 4.6.1)
Anvendelse	:	en kort beskrivelse af nøgleordet
Virkemåde	:	en beskrivelse af virkemåden for mere indviklede RcComal80 strukturer eller sætninger
Bemærkninger	:	bemærkninger angående brugen af nøgleordet, advarsler, fejlmeldinger o.lign.
Eksempler	:	illustrering af nøgleordets virkefelt med kortere eller længere eksempler. Yderligere eksempler findes i de foregående kapitler samt i appendix E.

BEMÆRK: Alle underpunkter anvendes ikke nødvendigvis ved beskrivelse af et nøgleord.

13.1 ABS

RcComal80 funktion

Format
ABS (nudtr)

nudtr : et vilkårligt numerisk udtryk

Anvendelse
Funktionen anvendes til at beregne den absolutte værdi (den numeriske værdi) af et nudtr. Den absolutte værdi er et ikke-negativt tal.

Eksempel

```
0010 INPUT "Indtast 2 tal  ": tal1, tal2
0020 PRINT "Forskellen mellem ";tal1;"og ";tal2;
0030 PRINT "er ";ABS(tal1-tal2)
0040 END

RUN
Indtast 2 tal :-7 16
Forskellen mellem -7 og 16 er 23
END
AT 0040
```

13.2 AND

RcComal80 operator

Format
nudtr1 AND nudtr2

nudtr1 : et vilkårligt numerisk udtryk opfattet som logisk udtryk
nudtr2 : et vilkårligt numerisk udtryk opfattet som logisk udtryk

Et logisk udtryk opfattes som falsk, hvis det er lig nul; ellers opfattes det som sandt.

Operatorprioritet = 7 (se afsnit 4.6.1)

Anvendelse
Den logiske operator AND er sand (sættes lig 1), hvis både nudtr1 og nudtr2 er sande (forskellige fra nul). Hvis enten nudtr1 eller nudtr2 er falsk (lig nul), bliver resultatet også falsk (nul).

Eksempel

```
0010 DIM logisk$(0:1) OF 5
0020 logisk$(FALSE):="FALSK"; logisk$(TRUE):="SAND"
0030 ZONE 10
0040 PRINT "AND";TAB(7);"*","FALSK","SAND"
0050 FOR i:=1 TO 30 DO PRINT "-";
0060 PRINT
0070 FOR udsagn:=FALSE TO TRUE DO
0080   PRINT logisk$(udsagn);TAB(7);"*",
0090   PRINT logisk$(udsagn AND FALSE),
0100   PRINT logisk$(udsagn AND TRUE)
0110 NEXT udsagn

RUN
AND   *  FALSK     SAND
------------------------------
FALSK *  FALSK     FALSK
SAND  *  FALSK     SAND
END
AT 0110
```

13.3 AT

RcComal80 funktion

Format
AT (nudtr1,nudtr2)

nudtr1 : et numerisk udtryk med værdi i intervallet 1 <= nudtr1 <= 80
nudtr2 : et numerisk udtryk med værdi i intervallet 1 <= nudtr2 <= 25

Anvendelse
Funktionen benyttes i PRINT- eller INPUT-sætninger til at flytte udskriften til en bestemt position på skærmen.

Bemærkninger
1. Skærmen adresseres på følgende måde med AT(x,y) :

2. AT kan kun benyttes i forbindelse med udskrift på skærmen.

3. Hvis nudtr2 er lig 25, skal statuslinien nederst på skærmen være fjernet (se appendix C).

Eksempel 1

```
0010 PRINT CHR$(12) // slet skærmen
0020 INPUT AT(30,12),"Kassestørrelse : ": kasse
0030 IF kasse>10 THEN kasse:=10
0040 IF kasse<1 THEN kasse:=1
0050 MARGIN 0
0060 FOR xkoord:=1 TO 80 DO
0070   PRINT AT(xkoord,12-kasse);"*";
0080   PRINT AT(xkoord,12+kasse);"*";
0090 NEXT xkoord
0100 FOR ykoord:=12-kasse TO 12+kasse DO
0110   PRINT AT(1,ykoord);"*";AT(80,ykoord);"*";
0120 NEXT ykoord
```

Eksempel 2

```
0010 PRINT CHR$(12) // slet skærm
0020 PRINT AT(5,2);"COSINUS OG"
0030 PRINT AT(5,3);"SINUS FUNKTIONEN"
0040 FOR j:=2 TO 80 DO PRINT AT(j,11);"-";// vandret streg
0050 FOR i:=1 TO 21 DO PRINT AT(40,i);"I";// lodret streg
0060 FOR rd:=-PI TO PI STEP 0.1 DO
0070   PRINT AT(rd*10+40,-SIN(rd)*10+11);"S"
0080   PRINT AT(rd*10+40,-COS(rd)*10+11);"C"
0090 NEXT rd
0100 PRINT AT(1,22);
0110 END
```

13.4 ATN

RcComal80 funktion

Format
ATN(nudtr)

nudtr : Et vilkårligt numerisk udtryk

Anvendelse
Funktionen udregner den vinkel (udtrykt i radianer) hvis tangens er lig nudtr, dvs. at funktionen er den omvendte funktion til tangens (TAN).

Eksempel

```
0010 // Ud fra ATN kan de øvrige arcus funktioner udregnes
0020 FUNC asin(x)// Arcus sinus
0030   IF ABS(x)>=1 THEN
0040     IF ABS(x)=1 THEN RETURN SGN(x)*PI/2
0050   ELSE
0060     RETURN ATN(x/SQR(1-x*x))
0070   ENDIF
0080 ENDFUNC asin
0090
0100 FUNC acos(x)// Arcus cosinus
0110   IF x=0 THEN
0120     RETURN PI/2
0130   ELSE
0140     IF ABS(x)<=1 THEN RETURN ATN(SQR(1-x*x)/x)+PI*(x<0)
0150   ENDIF
0160 ENDFUNC acos
0170
0180 FUNC acot(x)// Arcus cotangens
0190   IF x<>0 THEN
0200     RETURN ATN(1/x)+PI*(x<0)
0210   ENDIF
0220 ENDFUNC acot
0230
0240 ZONE 20
0250 PRINT "i","asin(i)","acos(i)"
0260 FOR i:=0 TO 1 STEP 0.1 DO PRINT i,asin(i),acos(i)
0270 PRINT
0280 PRINT "i","acot(i)","atan(i)"
0290 FOR i:=0.1 TO 0.9 STEP 0.1 DO PRINT i,acot(i),ATN(i)
0300 END
```

```
RUN
i               asin(i)                 acos(i)
0               0                       1.570796326794
0.1             0.1001674211615         1.470628905634
0.2             0.2013579207902         1.369438406005
0.3             0.3046926540152         1.26610367278
0.4             0.4115168460677         1.159279480726
0.5             0.5235987755979         1.047197551196
0.6             0.6435011087933         0.9272952180014
0.7             0.7753974966105         0.7953988301838
0.8             0.9272952180014         0.6435011087927
0.9             1.119769514998          0.4510268117957
1               1.570796326794          0

i               acot(i)                 atan(i)
0.1             1.471127674304          9.966865249115E-002
0.2             1.373400766946          0.1973955598499
0.3             1.279339532318          0.2914567944779
0.4             1.190289949683          0.380506377112
0.5             1.107148717794          0.4636476090008
0.6             1.030376826524          0.5404195002706
0.7             0.9600703624053         0.6107259643893
0.8             0.8960553845712         0.6747409422236
0.9             0.8379812250082         0.7328151017866
```

13.5 AUTO

RcComal80 kommando

Format

AUTO [{ lnr [,] / , lnr spring / lnr,lnr spring }]

lnr : første linienummer der skal indtastes
lnr spring : forskellen mellem linienumrene under indtastningen

Anvendelse
Kommandoen anvendes til automatisk linienummerering af et program, der skal indtastes.

Bemærkninger
1. Alle programlinier skal indledes med et linienummer. Hvis flere linier skal indtastes, simplificerer AUTO opgaven.

2. Man kommer ud af AUTO-tilstanden ved at trykke ESC på tastaturet

3. Hvis der optræder fejl under indtastningen, angives fejlen i skærmens øverste højre hjørne, og cursoren forbliver på linien.

4. Kombinationerne af parametrene til kommandoen AUTO, har følgende betydning:

AUTO	Linienummereringen starter ved linie 0010 og fortsætter med spring på 0010
AUTO lnr	Linienummereringen starter ved linie lnr og fortsætter med spring på lnr
AUTO lnr,	Linienummereringen starter ved linie lnr og fortsætter med spring på 0010
AUTO ,lnr spring	Linienummereringen starter ved linie 0010 og fortsætter med spring på lnr spring
AUTO lnr,lnr spring	Linienummereringen starter ved linie lnr og fortsætter med spring på lnr spring

Eksempel	Kommentar
AUTO	0010 udskrives på skærmen. Når linien er indtastet udskrives 0020, derefter 0030, 0040, osv.
AUTO 50	Linienummereringen bliver 0050, 0100, 0150, 0200 osv.
AUTO 50,	Linienummereringen bliver 0050, 0060, 0070, 0080 osv.
AUTO ,100	Linienummereringen bliver 0010, 0110, 0210, 0310 osv.
AUTO 50,100	Linienummereringen bliver 0050, 0150, 0250, 0350 osv.

13.6 BYE

RcComal80 sætning/kommando

Format
BYE

Anvendelse
Sætningen/kommandoen anvendes til at returnere til styresystemet.

Eksempel
Med nedenstående logon-program (se appendix F) kan ikke alle og enhver starte RcComal80 op.

```
0010 PROC fejl HANDLER
0020   PRINT "Fejl";ERR
0030   BYE
0040 ENDPROC fejl
0050 ENABLE fejl
0060 DIM kode$ OF 10
0070 INPUT "Indtast den hemmelige kode ":kode$
0080 IF kode$<>"qwerty" THEN
0090   PRINT "Forkert kode"
0100   BYE
0110 ENDIF
0120 PRINT "Velkommen"
0130 NEW
```

13.7 CASE - WHEN - ENDCASE

RcComal80 struktur

Format
CASE udtryk0 OF
WHEN udtryk1a ,udtryk1b...
sætningsliste-1

[WHEN udtrykna ,udtryknb]...]
[sætningsliste-n]

[OTHERWISE]
[sætningsliste-0]
ENDCASE

udtryk0 : nøgleudtrykket, enten et numerisk udtryk eller et strengudtryk. De øvrige udtryk.. skal være af samme type som udtryk0
sætningsliste : RcComal80 sætninger

Anvendelse
Konstruktionen benyttes til at få udført en ud af flere sætningsblokke afhængig af værdien af et udtryk.

Virkemåde
1. Værdien af udtryk0 i CASE-OF sætningen udregnes.

2. Værdien af udtrykkene i WHEN udtryk (,udtryk) sætningerne udregnes en for en, indtil der findes en værdi lig værdien i trin 1. Hvis sammenfaldet optræder i den i'te WHEN-sætning, vil sætningsliste-i blive udført.

3. Når sætningsliste-i er udført, og hvis sætningsliste-i ikke foretager et hop ud af konstruktionen, hoppes der til første sætning efter ENDCASE.

4. Hvis der ikke udregnes et WHEN udtryk der svarer til udtrykket i trin 1 udføres sætningsliste-0 (sætningslisten efter OTHERWISE). Hvis OTHERWISE ikke er angivet, udskrives feji 0115: ULOVLIG CASE VÆRDI.

Bemærkning
1. Konstruktionen kan ikke udføres som kommando.

Eksempel 1

```
0010 // eksempel på brug af case
0020 FOR i:= 1 TO 5 DO
0030   CASE i OF
0040   WHEN 1,3
0050     PRINT i;" (1 eller 3)"
0060   WHEN 5
0070     PRINT i;" (5)"
0080   OTHERWISE
0090     PRINT i
0100   ENDCASE
0110 NEXT i
0120 END
```

```
RUN
1  (1 eller 3)
2
3  (1 eller 3)
4
5  (5)
END
AT 0120
```

Eksempel 2

```
0010 DIM måned$ OF 3
0020 WHILE TRUE DO
0030   INPUT "Månednavn (3 tegn er nok) : ": måned$
0040   CASE måned$ OF
0050   WHEN "jan", "mar", "maj", "jul", "aug", "okt", "dec"
0060     PRINT måned$;" har 31 dage."
0070   WHEN "apr", "jun", "sep", "nov"
0080     PRINT måned$;" har 30 dage."
0090   WHEN "feb"
0100     PRINT måned$;" har for det meste 28 dage."
0110   OTHERWISE
0120     PRINT "Denne måned eksisterer ikke"
0130   ENDCASE
0140 ENDWHILE
0150 END
```

```
RUN
Månednavn (3 tegn er nok) : april
apr har 30 dage.
Månednavn (3 tegn er nok) : nøv
Denne måned eksisterer ikke
Månednavn (3 tegn er nok) : ESC
STOP
AT 0030
```

13.8 CHAIN

RcComal80 sætning

Format
CHAIN filnavn

filnavn : navnet på en diskettefil angivet i anførselstegn

Anvendelse
Sætningen anvendes til at afslutte et program samt automatisk LOADe og udføre et nyt program, der er SAVE'd i en diskettefil.

Bemærkninger
1. Datalageret slettes når CHAIN-sætningen udføres
2. Programudførelsen i det angivne program (filnavn) starter i programlinien med det laveste linienummer
3. Programnavnet angivet med filnavn skal være gemt med SAVE.
4. Når man har CHAIN'et til et nyt program, vil det nye filnavn blive angivet i statuslinien, når programmet stopper.

Eksempel	Kommentar
0010 // Program a 0020 PRINT "Program a" 0030 END SAVE "programa"	Nu er programmet gemt under navnet programa.
NEW	Slet program- og datalageret.
0010 // Program b 0020 PRINT "Program b" 0030 CHAIN "programa" SAVE "programb"	Nyt program, der kalder programa.
RUN Program b Program a END AT 0030	Bemærk, at filnavnet i status-linien er ændret til programa.

13.9 CHR$

RcComal80 funktion

Format
CHR$(nudtr)

nudtr : vilkårligt numerisk udtryk mellem 0 og 255 (incl)

Anvendelse
Funktionen returnerer med det tegn, som svarer til ASCII-værdien af nudtr.

Bemærkninger
1. Sammenhængen mellem nudtr og tegnene er vist i appendix D.

2. Funktionen må anvendes i et vilkårligt strengudtryk.

3. Hvordan skærmen reagerer på forskellige CHR$-koder fremgår af appendix C.

Eksempel

```
0010 // Programmet udskriver det normale tegnsæt på skærmen
0020 MARGIN 64
0030 ZONE 8
0040 PRINT CHR$(12)// Blanker skærmen
0050 FOR ch:= 32 TO 127 DO PRINT ch;CHR$(ch),
0060 PRINT
0070 PRINT CHR$(7)// Bib
0080 END

RUN
32      33 !    34 "    35 §    36 $    37 %.   38 &    39 '
40 (    41 )    42 *    43 +    44 ,    45 -    46 .    47 /
48 0    49 1    50 2    51 3    52 4    53 5    54 6    55 7
56 8    57 9    58 :    59 ;    60 <    61 =    62 >    63 ?
64 @    65 A    66 B    67 C    68 D    59 E    70 F    71 G
72 H    74 I    74 J    75 K    76 L    77 M    78 N    79 O
80 P    81 Q    82 R    83 S    84 T    85 U    86 V    87 W
88 X    89 Y    90 Z    91 Æ    92 Ø    93 Å    94 Ü    95 _
96 `    97 a    98 b    99 c    100 d   101 e   102 f   103 g
104 h   105 i   106 j   107 k   108 l   109 m   119 n   111 o
112 p   113 q   114 r   115 s   116 t   117 u   118 v   119 w
120 x   121 y   122 z   123 æ   124 ø   125 å   126 ü   127
END
AT 0080
```

13.10 CIRCLE

RcComal80 grafik sætning/kommando

Format
CIRCLE radius ,buestart ,bue

radius : et positivt numerisk udtryk
buestart,bue : et numerisk udtryk, der angiver et radianantal

Anvendelse
Sætningen/kommandoen anvendes til at tegne en elipse eller elipsebue med udgangspunkt i det løbende punkt. Tegningen af elipsen starter i det punkt på elipsen, der svarer til vinklen buestart (udtrykt i radianer). Ud fra dette punkt tegnes en elipsebue beskrevet ved bue, hvor bue normalt er et tal mellem -2*PI og 2*PI.

Bemærkninger
1. Hvis bue er enten 2*PI eller -2*PI tegnes en hel elipse.

2. Hvis bue er positiv, tegnes buen mod uret, hvis bue er negativ, tegnes buen med uret.

Eksempel

```
0010 PROC pie(n,REF a(),rad) CLOSED
0020   sum:=0
0030   FOR i:=1 TO n DO sum:=sum+a(i)
0040   andel:=0
0050   MOVETO rad,0
0060   FOR i:=1 TO n DO
0070     CIRCLE rad,andel*2*PI,a(i)/sum*2*PI
0080     andel:=andel+a(i)/sum
0090     xnu:=GPARM(0); ynu:=GPARM(1)
0100     DRAWTO 0,0
0110     MOVETO xnu,ynu
0120   NEXT i
0130 ENDPROC pie
0140
0150 INPUT "Antal værdier : ": ant
0160 DIM værdier(ant)
0170 FOR i:=1 TO ant DO INPUT "Indtast værdi :": værdier(i)
0180 OPEN GRAPHICS 1
0190 xudstr:=GPARM(101)*GPARM(104)
```

```
0200 yudstr:=GPARM(102)*GPARM(105)
0210 WINDOW -xudstr,xudstr,-yudstr,yudstr
0220 radius:=xudstr
0230 IF radius>yudstr THEN radius:=yudstr
0240 EXEC pie(ant,værdier,radius*0.9)
0250 END
```

13.11 CLEAR

RcComal80 grafik sætning/kommando

Format
CLEAR

Anvendelse
Sætningen sletter grafikbilledet.

13.12 CLOSE

RcComal80 sætning/kommando

Format
CLOSE [[FILE] strømnr]

strømnr : et numerisk udtryk, hvis værdi angiver nummeret på en datastrøm, 1 <= strømnr <= 5

Anvendelse
Sætningen/kommandoen bruges til at lukke en eller alle datastrømme.

Bemærkninger
1. Anføres CLOSE uden FILE strømnr lukkes alle åbne strømme i programmet.

2. CLOSE benyttes til at lukke en datastrøm, så den eventuelt kan åbnes igen i en anden mode.

Eksempel

```
0010 OPEN FILE 1,"datafil",WRITE
0020 RANDOMIZE
0030 FOR i:=1 TO 10 DO WRITE FILE 1: RND
0040 CLOSE FILE 1 // Luk efter brug
0050 END
```

13.13 CLOSE GRAPHICS

RcComal80 grafik sætning/kommando

Format
CLOSE GRAPHICS

Anvendelse
Sætningen/kommandoen afslutter den aktuelle grafikbehandling.

Bemærkning
1. Hvis man tegner grafik på f.eks. en skriver, udskrives billedet først, når CLOSE GRAPHICS udføres.

13.14 CON

RcComal80 kommando

Format
CON

Anvendelse
CONtinue-kommandoen benyttes til at genstarte udførelsen af et program, der er stoppet af programsætningerne STOP, END, ved ESCape eller ved fejl.

Bemærkninger
1. Programudførelsen genoptages med sætningen umiddelbart efter den sætning, hvor programmet standsede.

2. Der må ikke slettes, indføjes eller rettes linier i programmet før CON.

3. Hvis man retter i et program og derefter skriver CON, fås fejludskriften 0095: CON IKKE TILLADT.

4. CON må ikke benyttes til genoptagelse af programudførelsen, hvis årsagen til standsningen af programudførelsen findes i en pakke.

13.15 CONTINUE

RcComal80 sætning

Format
CONTINUE

Anvendelse
Bruges kun i en PROC-HANDLER. Sætningen bevirker, at programudførelsen fortsætter med sætningen efter den sætning, der forårsagede kaldet af PROC-HANDLER.

Bemærkning
1. Sætningen kan ikke udføres som kommando.

Eksempel

```
0010 PROC nuldiv HANDLER
0020   IF ERR=104 THEN
0030     PRINT "*** Division med nul"
0040     CONTINUE // Fortsæt hovedprogram
0050   ENDIF
0060 ENDPROC nuldiv
0070
0080 ENABLE nuldiv
0090 i := 1/0 // Fremprovoker fejl
0100 PRINT "Slut"
0110 END

RUN

*** Division med nul
Slut
END
AT 0110
```

13.16 COPY

RcComal80 kommando

Format
COPY filnavn1 , filnavn2

filnavn1 : navnet på en datastrøm, der skal kopieres fra. Navnet angives i anførselstegn
filnavn2 : navnet på en datastrøm, der skal kopieres over i. Navnet angives i anførselstegn

Anvendelse
Kommandoen kopierer en datastrøm over i en anden datastrøm.

Bemærkninger
1. Kommandoen kan bruges til at flytte filer mellem 2 disketteenheder, mellem diskette og kassettebånd eller fra en diskettefil til en ydre enhed (skærm, printer).

2. Hvis filnavn2 angiver en diskettefil, må den ikke eksistere på forhånd, ellers udskrives fejl 0213: FIL EKSISTERER ALLEREDE.

3. COPY benytter det frie brugerareal under kopieringen. Hvis man derfor ikke skal bruge det program, der ligger i program- og data-lageret, bør man skrive NEW før COPY.

4. COPY må ikke anvendes til at flytte SAVEde filer til eller fra kassettebånd. Her skal man bruge SAVE og LOAD.

Eksempel
Med følgende kommando, kan man udskrive en fil, der er gemt med kommandoen LIST eller en datafil, der er skrevet med PRINT FILE, på printeren:

```
COPY "eksempela", "printer"
```

13.17 COS

RcComal80 funktion

Format
COS (nudtr)

nudtr : et numerisk udtryk, der angiver et radianantal

Anvendelse
Funktionen udregner cosinus til en vinkel udtrykt i radianer.

Eksempel

```
0010 PRINT "Programmet omformer grader til radianer ";
0020 PRINT "og udregner cosinus"
0030 PRINT
0040 INPUT "Indtast en vinkel (i grader) :": vinkel
0050 radianer:= vinkel*PI/180
0060 PRINT vinkel;"grader svarer til ";radianer;" Radianer."
0070 PRINT "COS( ";vinkel;") = ";COS(radianer)
0080 END

RUN
Programmet omformer grader til radianer og udregner cosinus

Indtast en vinkel (i grader) :60
60 grader svarer til 1.047197551196 Radianer.
COS( 60 ) = 0.5000000000007

END
AT 0080
```

13.18 CREATE

RcComal80 sætning/kommando

Format
CREATE filnavn , længde

filnavn : navnet på den diskettefil (angivet i anførselstegn), der skal dannes
længde : vilkårligt positivt numerisk udtryk, begrænset af diskettens kapacitet

Anvendelse
Sætningen/kommandoen benyttes til at oprette en fil med direkte tilgang på en diskette. længde angiver størrelsen af filen i 1024 tegns blokke.

Bemærkninger
1. Sætningen benyttes kun til at reservere plads til filer med direkte tilgang, da sekventielle filer automatisk oprettes og udvides

2. Forsøger man at oprette en fil, der allerede eksisterer, fås fejl 0213: FIL EKSISTERER ALLEREDE.

3. Er disketten skrivebeskyttet fås fejl 0209: DISKETTE SKRIVEBESKYTTET.

4. Er der ikke plads til flere filer på disketten fås fejl 0215: DISKETTE FULD.

Eksempel

```
0010 PRINT "Filopretter"
0020 PRINT
0030 DIM navn$ OF 20
0040 INPUT "Filnavn          :": navn$
0050 INPUT "Poststørrelse    :": poststørrelse
0060 INPUT "Antal poster     :": antposter
0070 længde:=poststørrelse*antposter DIV 1024 + 1
0080 CREATE navn$,længde
0090 PRINT "Filen er oprettet og fylder ";længde;"k bytes"
0100 END
```

```
RUN
Filopretter

Filnavn          :register
Poststørrelse    :40
Antal poster     :100
Filen er oprettet og fylder 4 k bytes
END
AT 0100
```

13.19 DATA

RcComal80 sætning

Format

DATA { n konst / s konst } [, { n konst / s konst }] ...

n konst : numerisk konstant
s konst : streng konstant

Anvendelse

Sætningen bruges til at opbevare værdier i selve programmet som derefter kan læses ved hjælp af READ-sætningen.

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. Ved starten af et programs udførelse (RUN) gennemgås programmet sekventielt. Alle DATA-sætninger, der ikke findes i en lukket procedure eller funktion, kædes sammen til en dataliste og en datapegepind sættes til at pege på det første element i den første DATA-sætning. Ved starten af en lukket procedure eller en lukket funktions udførelse og inden initialisering af en RcComal80-pakke, sker det tilsvarende indenfor den pågældende struktur.

3. Både tal- og streng-konstanter må optræde i samme DATA-sætning.

Eksempel

```
0010 DIM spørg$ OF 80, svar$ OF 80, korr$ OF 80
0020 EXEC overskrift
0030 WHILE NOT EOD DO
0040   READ spørg$,korr$
0050   antalsvar:=0
0060   REPEAT
0070     PRINT spørg$
0080     INPUT " ? ":svar$;
0090     IF svar$=korr$ THEN
0100       PRINT TAB(40);"** Korrekt **"
0110     ELSE
0120       PRINT TAB(40);">>>> Forkert !"
0130     ENDIF
0140     antalsvar:=antalsvar+1
0150   UNTIL antalsvar>=3 OR svar$=korr$
0160   IF svar$<>korr$ THEN
0170     PRINT "Det korrekte svar er ";korr$
0180   ENDIF
0190 ENDWHILE
```

```
0200 PROC overskrift CLOSED
0210   DIM txt$ OF 80
0220   // En lukket procedure har en lokal DATA-liste
0230   WHILE NOT EOD DO
0240     READ txt$
0250     PRINT txt$
0260   ENDWHILE
0270   DATA "Oversættelsesøvelse",""
0280   DATA "Oversæt følgende ord til dansk"
0290 ENDPROC overskrift
0300 DATA "table","bord","dog","hund","duck","and"

RUN
Oversættelsesøvelse
Oversæt følgende ord til dansk
table ? bord                    ** Korrekt **
dog ? and                       >>>> Forkert !
dog ? høne                      >>>> Forkert !
dog ? ælling                    >>>> Forkert !
Det korrekte svar er hund
duck ? and                      ** Korrekt **
END
AT 0300
```

13.20 DATE$

RcComal80 funktion

Format
DATE$

Anvendelse
DATE$ er en funktion, der returnerer den aktuelle dato.

Bemærkninger
1. DATE$ har formen AAAA.MM.DD, hvor AAAA er året, MM er måneden og DD er dagen. F.eks. er 1987.01.20 den 20. januar i 1987.

13.21 DEL

RcComal80 kommando

Format

DEL { lnr1 [,] | , lnr2 | lnr1, lnr2 }

lnr1 : linienummeret på første sætning, der skal slettes
lnr2 : linienummeret på sidste sætning, der skal slettes

Anvendelse
Kommandoen anvendes til at slette en eller flere programlinier i det program, der ligger i programlageret.

Bemærkninger
1. Kombinationerne af parametrene til DEL-kommanoden har følgende betydning:

DEL lnr1	sletter linien med linienummer lnr1
DEL lnr1,	sletter alle linierne hvor linienummeret >= lnr1.
DEL ,lnr2	sletter alle linierne hvor linienummeret <= lnr2
DEL lnr1,lnr2	sletter alle linierne hvor lnr1 <= linienummeret <= lnr2.

Eksempel
Hvis vi i det følgende forudsætter, at programlageret indeholder følgende program:

```
0010 PRINT
0020 PRINT
0030 PRINT
0040 PRINT
0050 PRINT
0060 PRINT
```

Da vil:

del 20	slette linie 0020
del 40,	slettte linierne 0040, 0050, 0060
del ,20	slette linierne 0010, 0020
del 30,40	slette linierne 0030, 0040

13.22 DELETE

RcComal80 sætning/kommando

Format

DELETE filnavn

filnavn : Navnet på en diskettefil, der ønskes slettet

Anvendelse
Sætningen/kommandoen sletter filen filnavn fra diskettens katalog.

Bemærkninger
1. Hvis filen ikke eksisterer, udskrives der ikke en fejlmeddelelse.

2. Er disketten skrivebeskyttet udskrives fejl 0209: DISKETTE SKRIVEBESKYTTET.

3. DELETE-kommandoen har ingen virkning på kassettebåndsfiler.

Eksempel

```
0010 // Programmet sletter de angivne filnavne
0020 PROC fejl HANDLER
0030   PRINT "*** ";filnavn$;" er ikke slettet"
0040   CASE SYS(0) OF
0050   WHEN 209
0060     PRINT "*** Diskette skrivebeskyttet"
0070     CONTINUE
0080   WHEN 214
0090     PRINT "*** Fil eksisterer ikke"
0100     CONTINUE
0110   WHEN 212
0120     PRINT "*** Fejl i filnavn"
0130     CONTINUE
0140   WHEN 222
0150     PRINT "*** Fil skrivebeskyttet"
0160     CONTINUE
0170   OTHERWISE
0180     // ingen speciel aktion
0190   ENDCASE
0200 ENDPROC fejl
0210
```

```
0220 DIM unit$ OF 1,navn$ OF 11,filnavn$ OF 14
0230 ENABLE feji
0240 PRINT "Programmet sletter filer på disketter"
0250 REPEAT
0260   PRINT
0270   INPUT "Unitnr : ": unit$
0280   INPUT "Filnavn :": navn$
0290   filnavn$:= "/"+unit$+"/"+navn$
0300   DELETE filnavn$
0310 UNTIL FALSE
```

Eksempel
DELETE kan også afgives som kommando:

```
DELETE "tips.csv"
```

13.23 DIM

RcComal80 sætning/kommando

Format

DIM { array / streng / teksttabel } [, { array / streng / teksttabel }] ...

array	:	navn(grænse , grænse ...)
streng	:	navn$ OF nudtr
teksttabel	:	navn$(grænse , grænse ...) OF nudtr
navn	:	navn, der følger reglerne for en identifikator
grænse	:	[nudtr1 :] nudtr2
nudtr	:	positive numeriske udtryk
nudtr1, nudtr2	:	numeriske udtryk, nudtr1 < nudtr2

Anvendelse

Sætningen/kommandoen anvendes til at erklære reelle talsæt (vektorer og matricer), strenge og teksttabeller.

Bemærkninger

1. Det er ikke tilladt at redimensionere en allerede dimensioneret variabel.

2. En DIM-sætning må gerne optræde lokalt i en lukket (CLOSED) procedure eller funktion. Hver gang proceduren eller funktionen kaldes, oprettes variablen påny, og når man returnerer fra proceduren eller funktionen fjernes den igen.

3. Efter at DIM er udført, har de reelle elementer værdien 0 og strenge længden 0.

4. Angives den nedre grænse ikke, sættes den til 1.

5. Antallet af dimensioner er begrænset af maskinens lager, samt af liniebredden.

6. Opbygning:
   DIM V(5) Dimensionerer et n-dimensionalt array (vektor) med 5 tal: V(1) V(2) V(3) V(4) V(5)

DIM M(5,2:4) Dimensionerer et to-dimensionalt array (matrix) med 5 rækker og 3 søjler:
M(1,2) M(1,3) M(1,4)
M(2,2) M(2,3) M(2,4)
M(3,2) M(3,3) M(3,4)
M(4,2) M(4,3) M(4,4)
M(5,2) M(5,3) M(5,4)

DIM S$ OF 15 Dimensionerer en strengvariabel, der kan indeholde op til 15 tegn.

DIM T$(4) OF 30 Dimensionerer en teksttabel, der kan indeholde op til 4 strenge, hver på ind til 30 tegn:
T$(1) T$(2) T$(3) T$(4)

7. Man kan kun tildele en værdi til en del af en tekstvariabel, hvis hele tekstvariablen har en værdi i forvejen.

13.24 DIR

RcComal80 sætning/kommando

Format
DIR [nudtr]

nudtr : positivt numerisk udtryk

Anvendelse
Sætningen/kommandoen udskriver en liste over filerne på diskette nummer nudtr. nudtr=1,2,3... svarer til CCP/M unit A,B,C,...

Bemærkning
1. Normalt udskrives listen på skærmen. Dette kan dog ændres med SELECT OUTPUT sætningen/kommandoen.

2. Angives nudtr ikke, forventes det, at PREFIX er sat til enten 1/ 2/ osv., og DIR vil da udskrive filerne på prefix-enheden

3. Oversigten over filnavnene afsluttes med en angivelse af den resterende plads på disketten.

4. DIR-kommandoen kan ikke bruges til at få udskrevet en liste over filerne på et kassettebånd. I stedet kan man f.eks. LOADe en fil, der ikke findes. RcComal80 vil da udskrive navnene på de filer, der springes over undervejs og på denne måde lave en liste over filerne på kassettebåndet.

Eksempel 1

	Kommentar
SELECT OUTPUT "printer"	Udskrift af navnene på diskette nr
DIR 1	1's filer på printeren

Eksempel 2

```
0010 // programmet udskriver filerne på en diskette
0020 // i alfabetisk orden
0030
0040 PROC sort(venstre, højre) CLOSED
0050   GLOBAL x$,k$,a$
0060   // sort sorterer elementer i rk k$(venstre)...k$(højre)
0070   i:= venstre; j:= højre
0080   x$:= k$((i+j) DIV 2)
0090   REPEAT
0100     WHILE k$(i)<x$ DO i:=i+1
```

```
0110      WHILE x$<k$(j) DO j:=j-1
0120      IF i<=j THEN
0130        a$:=k$(i); k$(i):=k$(j); k$(j):=a$
0140        i:=i+1; j:=j-1
0150      ENDIF
0160    UNTIL i>j
0170    IF venstre<j THEN EXEC sort(venstre,j)
0180    IF i<højre THEN EXEC sort(i,højre)
0190  ENDPROC sort
0200
0210  INPUT "Listning af hvilken unit ?": unit
0220  MARGIN 16
0230  ZONE 16
0240  SELECT OUTPUT "kat$$$"
0250  DIR unit // Udskriv kataloget i filen kat$$$
0260  SELECT OUTPUT "console"
0270
0280  DIM k$(100) OF 11,x$ OF 11,a$ OF 11
0290  OPEN 1,"kat$$$", READ
0300  n:= 1
0310  WHILE NOT EOF(1) DO
0320    INPUT FILE 1: k$(n)
0330    IF k$(n)<>"KAT$$$      " AND NOT EOF(1) THEN n:=n+1
0340  ENDWHILE
0350  n:=n-3
0360  PRINT "Unit nr ";unit
0370  CLOSE 1
0380  DELETE "kat$$$"// fjernes efter brug
0390
0400  // nu ligger filnavne i k$(1)...k$(n) (usorteret)
0410
0420  EXEC sort(1,n)// sorter elementerne
0430  MARGIN 80
0440  FOR i:= 1 TO n DO PRINT k$(i),
0450  PRINT
0460  END
```

13.25 DISABLE

RcComal80 sætning

Format
DISABLE

Anvendelse
Sætningen gør en eventuel ENABLE'd PROC-HANDLER inaktiv og lader systemet selv håndtere kommende fejl.

Bemærkning
1. Systemet udfører automatisk en DISABLE, hvis en ny PROC-HANDLER ENABLE's.

Eksempel

```
0010 PROC skrivnr HANDLER
0020   PRINT "*** Fejl nr :";ERR
0030   CONTINUE
0040 ENDPROC skrivnr
0050
0060 ENABLE skrivnr
0070 i:= SQR(-7) // Fremprovoker fejl
0080 DISABLE // Normal fejlbehandling
0090 i:= LOG(0)
0100 END

RUN

*** Fejl nr :103
AT 0090
ERROR : 0106
```

13.26 DISCARD

RcComal80 kommando

Format
DISCARD

Anvendelse
Kommandoen fjerner alle pakker fra programlageret. Inden pakkerne fjernes, udføres pakkernes exitprocedurer.

Bemærkning
1. Systemet udfører automatisk en DISCARD, hvis kommandoerne NEW eller LOAD bruges.

2. Det er ikke muligt at fjerne en enkelt pakke. DISCARD fjerner alle pakker.

13.27 DIV

RcComal80 operator

Format
nudtr1 DIV nudtr2

nudtr1 : vilkårligt numerisk udtryk
nudtr2 : vilkårligt numerisk udtryk forskellig fra nul

Operator prioritet = 3 (se afsnit 4.6.1)

Anvendelse
Operatoren foretager en heltalsdivision. Dvs. der udføres en division til et helt tal.

Bemærkning
1. Definitionen på a DIV b er: INT(a/ABS(b))*SGN(b)

Eksempel

```
0010 FUNC divi(a,b)
0020   RETURN INT(a/ABS(b))*SGN(b)
0030 ENDFUNC divi
0040
0050 RANDOMIZE
0060 ZONE 20
0070 PRINT "A","B","A DIV B","divi(A,B)"
0080 FOR i:=1 TO 10 DO
0090   REPEAT
0100     tal1:=RND*1000-500; tal2:=RND*1000-500
0110   UNTIL INT(tal2)<>0
0120   PRINT tal1,tal2,tal1 DIV tal2,divi(tal1,tal2)
0130 NEXT i
0140 END
```

13.28 DOCU

RcComal80 kommando

Format
DOCU navn1 [.navn2]

navn1 : navn på en pakke, procedure eller funktion der er kendt i programlageret
navn2 : navn på en PUBLIC procedure eller funktion i den pakke, der hedder navn1

Anvendelse
Kommandoen anvendes til at udskrive kommentarer (dokumentation) som står efter definitionen af en procedure, funktion, pakke, pakke-procedure eller en pakke-funktion.

Bemærkninger
1. Hvis de angivne navne ikke er kendte i programlageret, udskrives der ikke noget.

2. Hvis man vil se kommentarer hørende til en pakke-procedure eller en pakke-funktion skal man angive både navn1 og navn2. Ellers skal man alene angive navn1.

3. Dokumentation for en pakke, pakke-procedure eller en pakke-funktion skal laves af den, der laver pakken.

4. Dokumentation for en procedure eller en funktion eller i en RcComal80-pakke består af de linier med kommentarer, der følger lige efter erklæringen af proceduren, funktionen eller pakken.

Eksempel

```
USE strkonv
LISTPACK strkonv
  PACKAGE strkonv
     PROC lower(REF mystr$)
     PROC upper(REF mystr$)
  ENDPACKAGE

DOCU strkonv
  PACKAGE strkonv
  konverterer strenge til små eller store bogstaver

DOCU strkonv.lower
  PROC lower (REF mystr$)
  konvereterer strengparameteren til små bogstaver
```

13.29 DRAW

RcComal80 grafik sætning/kommando

Format
DRAW xkoor , ykoor

xkoor, ykoor : numeriske udtryk

Anvendelse
Sætningen/kommandoen tegner en linie fra det løbende punkt (x,y) til punktet (x+xkoor,y+ykoor). DRAW tegner altså relativt fra det løbende punkt.

Eksempel

```
0010 PRINT "Sierpinski-kuver."
0020 PRINT
0030 INPUT "Fra hvilken orden :": fra
0040 INPUT "Til hvilken orden :": til
0050
0060 OPEN GRAPHICS
0070 WINDOW 0,100,0,100
0080 h:=25; x0:=50; y0:=75
0085 FOR i:=1 TO fra-1 DO x0:=x0-h; h:=h/2; y0:=y0+h
0090 FOR i:=fra TO til DO
0100   x0:=x0-h; h:=h/2; y0:=y0+h
0110   MOVETO x0,y0
0120   EXEC a(i)
0130   DRAW h,-h
0140   EXEC b(i)
0150   DRAW -h,-h
0160   EXEC c(i)
0170   DRAW -h,h
0180   EXEC d(i)
0190   DRAW h,h
0200 NEXT i
0210
0220 PROC a(i)
0230   IF i>0 THEN
0240     EXEC a(i-1)
0250     DRAW h,-h
0260     EXEC b(i-1)
0270     DRAW 2*h,0
0280     EXEC d(i-1)
0290     DRAW h,h
0300     EXEC a(i-1)
0310   ENDIF
0320 ENDPROC a
```

```
0330
0340 PROC b(i)
0350   IF i>0 THEN
0360     EXEC b(i-1)
0370     DRAW -h,-h
0380     EXEC c(i-1)
0390     DRAW 0,-2*h
0400     EXEC a(i-1)
0410     DRAW h,-h
0420     EXEC b(i-1)
0430   ENDIF
0440 ENDPROC b
0450
0460 PROC c(i)
0470   IF i>0 THEN
0480     EXEC c(i-1)
0490     DRAW -h,h
0500     EXEC d(i-1)
0510     DRAW -2*h,0
0520     EXEC b(i-1)
0530     DRAW -h,-h
0540     EXEC c(i-1)
0550   ENDIF
0560 ENDPROC c
0570
0580 PROC d(i)
0590   IF i>0 THEN
0600     EXEC d(i-1)
0610     DRAW h,h
0620     EXEC a(i-1)
0630     DRAW 0,2*h
0640     EXEC c(i-1)
0650     DRAW -h,h
0660     EXEC d(i-1)
0670   ENDIF
0680 ENDPROC d
```

13.30 DRAWTO

RcComal80 grafik sætning/kommando

Format
DRAWTO xkoor , ykoor

xkoor,ykoor : numeriske udtryk

Anvendelse
Sætningen/kommandoen tegner en linie fra det løbende punkt (x,y) til punktet (xkoor,ykoor). DRAWTO tegner altså absolut fra det løbende punkt.

Eksempel

```
0010 INPUT "Tast tal mellem 1 og 50:": n
0020 OPEN GRAPHICS 1
0030 WINDOW -n,n,-n,n
0040 FOR i:=0 TO n DO
0050    EXEC streg(-n,i,-n+i,n)
0060    EXEC streg(i,n,n,n-i)
0070    EXEC streg(n,-i,n-i,-n)
0080    EXEC streg(-i,-n,-n,i-n)
0090 NEXT i
0110
0120 PROC streg(xfra,yfra,xtil,ytil)
0130    MOVETO xfra,yfra
0140    DRAWTO xtil,ytil
0150 ENDPROC streg
```

13.31 EDIT

RcComal80 kommando

Format

EDIT [{ lnr1 [,] }]
       { , lnr2 }
       { lnr1, lnr2 }

lnr1 : linienummeret på den første sætning, hvori der skal rettes
lnr2 : linienummeret på den sidste sætning, hvori der skal rettes

Anvendelse
Kommandoen anvendes til at rette i hele eller dele af det program, der ligger i programlageret.

Bemærkninger
1. Systemet udskriver den første sætning i området og cursoren fremkommer lige efter linienummeret. Brugeren kan nu rette den pågældende sætning ved hjælp af systemets editeringsfaciliteter (se afsnit 2). Når de ønskede rettelser er foretaget tastes vognretur, sætningen syntaksanalyseres og lagres. Derefter udskrives den næste sætning, der så kan rettes. Osv.

2. Kombinationerne af parametrene til EDIT-kommandoen har følgende betydning:

EDIT	Giver brugeren mulighed for at rette i hele programmet i programlageret.
EDIT lnr1	Giver brugeren mulighed for at rette linien med linienummeret = lnr1
EDIT lnr1,	Giver brugeren mulighed for at rette de linier, hvor linienummeret >= lnr1
EDIT ,lnr2	Giver brugeren mulighed for at rette de linier, hvor linienummeret <= lnr2
EDIT lnr1, lnr2	Giver brugeren mulighed for at rette de linier, hvor lnr1 <= linienummeret <= lnr2

3. En alternativ måde at rette i et program er først at udskrive dele af det med kommandoen LIST og derefter benytte tastaturets 4 pile samt øvrige editeringsfaciliteter til at rette i programmet med. Men husk, at hver rettet linie først bliver lagret, når man har trykket vognretur.

13.32 ENABLE

RcComal80 sætning

Format
ENABLE navn

navn : Navnet på en PROC-HANDLER

Anvendelse
Sætningen aktiverer en PROC-HANDLER.

Bemærkninger
1. Sætningen kan ikke udføres som kommando.

2. En PROC-HANDLER gøres inaktiv ved hjælp af DISABLE-sætningen.

Eksempel

```
0010 PROC fejlproc HANDLER
0020   IF ERR>=200 THEN
0030     PRINT "*** I/O-Fejl NR ";SYS(0)
0040     PRINT "*** Strømnr ";SYS(1)
0050   ELSE
0060     PRINT "*** Fejl nr ";SYS(0)
0070   ENDIF
0080   REPEAT
0090     INPUT "*** Funktion: F(ortsæt, G(entag, S(lut :": O$
0100     IF O$="F" THEN CONTINUE
0110     IF O$="G" THEN RETRY
0120   UNTIL O$="S"
0130 ENDPROC fejlproc
0140
0150 DIM O$ OF 1
0160 ENABLE fejlproc
```

13.33 END

RcComal80 sætning

Format
END

Anvendelse
Benyttes til at markere afslutningen af et program.

Bemærkninger
1. Sætningen kan ikke udføres som kommando.

2. RcComal80 kræver ikke, at programmet indeholder en END sætning, da programudførelsen standser, når programmets sidste sætning (med det højeste linienummer) er udført.

3. Programmet må gerne indeholde flere END-sætninger. Når en af disse mødes udskrives
   ```
   END
   AT XXXX
   ```
   hvor XXXX er linienummeret på END-sætningen.

Eksempel
```
...
0090 PRINT "Slut på program"
0100 END
RUN
Slut på program
END
AT 0100
```

13.34 ENTER

RcComal80 kommando

Format
ENTER filnavn

filnavn : navnet på en diskettefil angivet i anførselstegn

Anvendelse
Kommandoen bruges til at læse et program, der er gemt ved hjælp af kommandoen LIST, ind i lageret. Programlinierne på diskettefilen bliver sammenflettet med programlinierne i programlageret. Filen kan også indeholde linier med kommandoer.

Bemærkninger
1. Hvis programlagerets program og diskettefilens program har fælles linienumre, vil programlagerets linie blive slettet og diskettefilens linie blive indsat.

2. De linienumre, der eksisterer i programlageret, men ikke i diskettefilen, vil ikke blive slettet ved ENTER.

3. Hvis programmet i maskinens lager ikke ønskes benyttet, bør kommandoen NEW gives før ENTER kommandoen.

4. For at et program kan læses ind ved hjælp af ENTER, skal det være gemt ved hjælp af LIST.

5. Det er ikke muligt, at filen indeholder kommandoen ENTER.

Eksempel	Kommentar
20 // program a	
30 a:=7	Et program laves og gemmes
list "eksempela"	under navnet "eksempela"
new	Sletter programlageret
10 // program b	Nye programlinier indtastes
30 B:=3	
list	
0010 // program b	og listes på skærmen
0030 b:=3	
enter "eksempela"	"eksempela" indlæses oveni program b
list	
0010 // program b	Slutresultatet: en blanding af
0020 // program a	program a og program b, hvor
0030 a:=7	linie 0030 er overskrevet.

13.35 EOD

RcComal80 funktion

Format
EOD

Anvendelse
EOD (End Of Data) er en logisk funktion uden argumenter, som har værdien SAND (1) når det sidste element i datalisten er læst, ellers har den værdien FALSK (0).

Eksempel

```
0010 ZONE 10
0020 PRINT "EOD = ";EOD
0030 WHILE NOT EOD DO
0040   READ tal
0050   PRINT "EOD = ";EOD, "TAL = ";tal
0060 ENDWHILE
0070 DATA 7, 9, 13
0080 END

RUN
EOD = 0
EOD = 0  TAL = 7
EOD = 0  TAL = 9
EOD = 1  TAL = 13
END
AT 0080
```

13.36 EOF

RcComal80 funktion

Format
EOF(strømnr)

strømnr : et udtryk, hvis værdi er lig nummeret på en datastrøm,
1 <= strømnr <= 5

Anvendelse
EOF (End Of File) er en logisk funktion, som antager værdien SAND (1) når man forsøger at læse udover det sidste dataelement i en datastrøm, ellers har den værdien FALSK (0).

Bemærkning
1. Funktionen har kun mening i forbindelse med READ FILE, INPUT FILE og GET$.

Eksempel

```
0010 // åben filen, og skriv tallene fra 1 til 10 ud i den
0020 OPEN 1, "datafil", WRITE
0030 FOR i:=1 TO 10 DO WRITE FILE 1: i
0040 CLOSE 1// luk filen igen
0050
0060
0070 // åben filen igen, og læs tallene nok engang
0080 OPEN 1, "datafil", READ
0090 ZONE 20
0100 PRINT "EOF", "tal"
0110 REPEAT
0120   READ FILE 1: tal
0130   PRINT EOF(1), tal
0140 UNTIL EOF(1)
0150 CLOSE 1
0160
0170 DELETE "datafil"// slet datafilen efter brug
```

```
RUN
EOF          tal
0            1
0            2
0            3
0            4
0            5
0            6
```

```
0               7
0               8
0               9
0               10
1               10
END
AT 0170
```

13.37 ERR

RcComal80 funktion

Format
ERR

Anvendelse
Bruges kun i PROC-HANDLER. Funktionen returnerer med nummeret for sidste RUN-time fejl (se appendix B).

Bemærkning
1. Funktionen svarer præcis til SYS(0)

Eksempel

```
0010 PROC esctest HANDLER
0020   IF ERR=100 THEN
0030     escset:=TRUE
0040     CONTINUE
0050   ENDIF
0060 ENDPROC esctest
0070
0080 ENABLE esctest
0090 escset:=FALSE
0100 antal:=1; max:=50
0110 DIM alder(max),højde(max)
0120 REPEAT
0130   INPUT "Alder ? (ESC=slut) ":alder(antal);
0140   IF NOT escset THEN
0150     INPUT " Højde ? ":højde(antal)
0160     antal:=antal+1
0170   ENDIF
0180 UNTIL escset
0190 antal:=antal-1
0200 DISABLE
0210 // Nu er alder og højde indtastet
```

13.38 ERRTEXT$

RcComal80 funtion

Format
ERRTEXT$(nudtr)

nudtr : Et positivt numerisk udtryk

Anvendelse
Funktionen returnerer fejlmeddelelsen, der svarer til et givet fejlnummer (se appendix B).

13.39 Etikette

RcComal80 sætning

Format
navn :

navn : Et navn der følger reglerne for variabelnavne

Anvendelse
Sætningen anvendes til at navngive et bestemt sted i programmet, hvortil andre programsætninger kan referere.

Bemærkninger
1. Man kan referere til etiketter I sætningerne GOTO og RESTORE.

2. En variabel og en etikette må ikke have det samme navn.

3. En etikette kaldes også for en label.

Eksempel

```
0010 max:=50
0020 DIM alder(max),højde(max)
0030 FOR nr:=1 TO 50 DO
0040    INPUT "Alder ? (-1=slut) ":alder(nr);
0050    IF alder(nr)=-1 THEN GOTO behandling
0060    INPUT " Højde ? ":højde(nr)
0070 NEXT nr
0080 nr:=51 // Vi trækker 1 fra om et øjeblik
0090 behandling:
0100 nr:=nr-1
0110 // Nu er alder og højde indtastet
```

13.40 EXEC

RcComal80 sætning/kommando

Format
[EXEC] [packnavn.]navn [(par [,par] ...)]

packnavn : navnet på en pakke
navn : navnet på en procedure
par : navnet på en simpel variabel, et array, en strengvariabel, en teksttabel, et strengudtryk eller et numerisk udtryk

Anvendelse
Sætningen anvendes til at kalde en procedure.

Virkemåde
1. Proceduren med navnet navn fremfindes. De evt. aktuelle parametre i EXEC bliver overført til de eventuelle formelle parametre i PROC-sætningen. Hvis antallet eller typen af parametrene ikke passer, udskrives fejl 0112: PARAMETER FEJL eller fejl 0109: TYPE KONFLIKT.

2. Programudførelsen fortsætter i underprogrammet indtil RETURN eller ENDPROC mødes.

3. Derefter fortsætter programudførelsen med næste sætning efter EXEC.

Bemærkninger
1. EXEC kan eventuelt udelades.

2. Sætningen kan ikke udføres som kommando, hvis man har rettet i programmet siden sidste RUN.

3. Se desuden kapitel 8: Procedurer og funktioner.

4. Hvis en procedure i en pakke har samme navn, som en variabel, etikette eller en anden procedure etc., er det ikke muligt at kalde pakke-proceduren ved dens navn. Man må så skrive pakkens navn og et punktum foran procedurens navn (punktum-notation). Der må ikke være blanke mellem pakkenavnet og punktummet eller mellem punktummet og procedure-navnet.

Eksempel
Kald af procedure med to parametre og kald af pakkeprocedure med punktum-notationen:

```
0010 min(12,5)
0020 USE program
0030 program.kommando("pip m:=a:mine.dat")
```

13.41 EXITPROC

RcComal80 sætning

Format
EXITPROC navn

navn : Navn på en procedure eller funktion

Anvendelse
Sætningen anvendes kun i strukturen PACKAGE-ENDPACKAGE. Sætningen benyttes til at angive, hvilken af pakkens procedurer, der skal udføres når pakken fjernes. Den angivne procedure udføres automatisk ved fjernelse af pakken (NEW eller DISCARD).

Bemærkning
1. Der må kun være een EXITPROC-sætning i hver PACKAGE-ENDPACKAGE.

13.42 EXP

RcComal80 funktion

Format
EXP(nudtr)

nudtr : vilkårligt numerisk udtryk

Anvendelse
Funktionen udregner værdien af e (=2,718281828459) opløftet til potensen nudtr.

Eksempel
Ved hjælp af EXP kan man udregne cosinus hyperbolsk og sinus hyperbolsk.

```
0010 FUNC cosh(x)//cosinus hyperbolsk
0020   RETURN (exp(x)+1/exp(x))/2
0030 ENDFUNC cosh
0040 FUNC sinh(x)//sinus hyperbolsk
0050   RETURN (exp(x)-1/exp(x))/2
0060 ENDFUNC sinh
```

13.43 FALSE

RcComal80 konstant

Format
FALSE

Anvendelse
FALSE er en konstant med værdien 0.

Eksempel

```
0010 DIM måned$ OF 3, svar$ OF 3
0020 REPEAT
0030   INPUT "Skriv en måneds navn >": svar$
0040   fundet:= FALSE
0050   RESTORE
0060   WHILE NOT fundet AND NOT EOD DO
0070     READ måned$, dage
0080     IF måned$=svar$ THEN
0090       fundet:= TRUE
0100       PRINT måned$;" har ";dage;" dage."
0110     ENDIF
0120   ENDWHILE
0130   IF NOT fundet THEN PRINT "Den måned kender jeg ikke"
0140 UNTIL FALSE// uendelig løkke
0150 DATA "jan",31,"feb",28,"mar",31,"apr",30,"maj",31,"jun",30
0160 DATA "jul",31,"aug",31,"sep",30,"okt",31,"nov",30,"dec",31
```

```
RUN
Skriv en måneds navn >jul
jul har 31 dage.
Skriv en måneds navn >jam
Den måned kender jeg ikke
Skriv en måneds navn >jan
jan har 31 dage.
```

13.44 FOR

RcComal80 sætning

Format
FOR tvar:= stværdi TO slværdi [STEP trværdi]DO simpel sætning

tvar	:	tællevariabel, en simpel numerisk variable
stværdi	:	numerisk udtryk, startværdi
slværdi	:	numerisk udtryk, slutværdi
trværdi	:	numerisk udtryk, trinværdi
simpel sætning	:	en simpel RcComal80 sætning, dvs. CIRCLE, CLEAR, CLOSE, CLOSE GRAPHICS, CONTINUE, CREATE, DELETE, DRAW, DRAWTO, END, EXEC, GOTO, GPARM, GSX, INPUT, INPUT FILE, LOCATE, MOVE, MOVETO, OPEN FILE, OPEN GRAPHICS, PALETTE, PENCOLOR, PRINT, PRINT FILE, PRINT USING, READ, READ FILE, RESTORE, RETRY, RETURN, SELECT OUTPUT, STOP, TEXT, tildeling, WINDOW, WRITE FILE.

Anvendelse
Sætningen benyttes til at udføre en simpel sætning et bestemt antal gange.

Virkemåde
1. stværdi, slværdi og trværdi udregnes. Hvis trværdi ikke er angivet, sættes den til +1.

2. tvar sættes lig stværdi

3. Hvis trværdi er positiv (negativ) og tvar er større (mindre) end slværdi er slutbetingelsen opfyldt og programmet fortsætter med næste sætning.

4. simpel sætning udføres.

5. tvar sættes lig tvar plus trværdi og trin 3 udføres igen.

Bemærkninger
1. Sætningen kan ikke udføres som kommando.

2. Ønskes mere end en simpel sætning udført, skal konstruktionen FOR-NEXT benyttes.

Eksempel

```
0010 ZONE 10
0020 FOR i: = 1 TO 10 STEP 2 DO PRINT i,
0030 END

RUN
1          3          5          7          9
END
AT 0030
```

13.45 FOR - NEXT

RcComal80 struktur

Format
FOR tvar:= stværdi TO slværdi [STEP trværdi] DO
  sætningsliste
NEXT tvar

tvar	:	tællevariabel, en simpel numerisk variable
stværdi	:	numerisk udtryk, startværdi
slværdi	:	numerisk udtryk, slutværdi
trværdi	:	numerisk udtryk, trinværdi
sætningsliste	:	RcComal80 sætninger

Anvendelse
Konstruktion benyttes til at repetere udførelsen af en sætningsliste et bestemt antal gange.

Virkemåde
1. stværdi, slværdi og trværdi udregnes. Hvis trværdi ikke er angivet, sættes den lig +1.

2. tvar sættes lig stværdi.

3. Hvis tværdi er positiv (negativ), og tvar er større (mindre) end slværdi, er slutbetingelsen opfyldt, og programmet fortsætter med første sætning efter NEXT-sætningen.

4. sætningsliste udføres.

5. tvar sættes lig tvar plus trværdi, og trin 3 udføres igen.

Bemærkninger
1. Strukturen kan ikke udføres som kommando.

2. FOR-NEXT kan indeholde andre FOR-NEXT sætninger.

3. Hvis NEXT mangler fås feji 0096: FEJL I PROGRAMSTRUKTUR.

4. Hvis man hopper ind i en FOR-NEXT løkke udenom FOR-sætningen fås feji 0116: ULOVLIGT HOP.

13.46 FUNC - ENDFUNC

RcComal80 struktur

Format

FUNC navn [(par [,par] ...)] [CLOSED]
  sætningsliste
ENDFUNC navn

par:
```
        [REF]  nvar
       {[REF]  svar          }
        [REF]  ntabl ([,] ... )
          REF  stabl ([,] ... )
```

navn : navn på funktionen (max. 16 tegn)
nvar : vilkårligt numerisk variabel navn
svar : vilkårligt strengvariabel navn
ntabl : vilkårligt array-navn
stabl : vilkårligt teksttabel navn
sætningsliste : RcComal80 sætninger

Anvendelse

Konstruktionen benyttes til at definere en funktion.

Bemærkninger

1. navn skal være forskellig fra alle andre navne på procedurer, funktioner, etiketter, numeriske variable og strengvariable.

2. En funktion kan gøres lukket (CLOSED), så variable, der optræder i funktionen, er lokale og ikke berører variable i resten af programmet (de globale variable). Ønsker man i en CLOSED funktion at benytte en variabel, der optræder i resten af programmet, benyttes en GLOBAL- eller IMPORT-sætning. De lokale variable, der optræder i funktionen, slettes når man forlader funktionen.

3. Anføres REF foran variabelnavnet i parameterlisten (den formelle parameter) vil variablen i kaldet af funktionen (den reelle parameter) blive ændret samtidig med, at den formelle parameter ændres i funktionen. På denne måde kan man returnere resultater via parametrene.

4. Værdien af funktionen tildeles med RETURN-sætningen, hvorved man tillige returnerer fra funktionen.

5. Hvis der ikke mødes en RETURN-sætning under udførelsen af funktionen, udskrives fejl 0113: FUNKTIONSVÆRDI UDEFINERET

6. Hvis en pakke, der også bruges udenfor, skal bruges i en CLOSED funktion, er det nødvendigt at importere (v.hj.a. IMPORT eller GLOBAL) pakken og/eller de procedurer og funktioner, pakken stiller til rådighed. Hvis pakken importeres, kan pakkeprocedurerne og pakkefunktionerne kaldes ved punktum-notationen (se EXEC 13.40).

Eksempel

```
0010 FUNC fahrenheit(celcius)
0020   // Funktionen omformer celcius til fahrenheit
0030   RETURN celcius*9/5+32
0040 ENDFUNC fahrenheit
0050
0060 MARGIN 80
0070 ZONE 10
0080 PRINT "Celcius","Fahrenheit"
0090 FOR c:=0 TO 100 STEP 20 DO PRINT c,fahrenheit(c)

RUN
Celcius    Fahrenheit
0              32
20             68
40             104
60             140
80             176
100            212
END
AT 0090
```

13.47 FUNC - EXTERNAL

RcComal80 sætning

Format
FUNC navn [(par [, par] ...)] EXTERNAL filnavn

par:
```
  [REF]  nvar
{ [REF]  svar                }
  REF    ntabl ( [,] ... )
  REF    stabl ( [,] ... )
```

navn : Navn på funktionen (max 16 tegn)
nvar : Vilkårligt numerisk variabelnavn
svar : Vilkårligt strengvariabelnavn
ntabl : Vilkårligt array-navn
stabl : Vilkårligt tekstabelnavn
filnavn : Navn på den diskettefil, hvori den externe funktion er gemt med kommandoen SAVE

Anvendelse
Sætningen benyttes til at erklære en extern funktion.

Bemærkninger

1. Den externe funktion skal være SAVE't i filnavn og første linie i dette program skal være et funktionshoved for en lukket (CLOSED) funktion. Den lukkede funktion må ikke indeholde GLOBAL-sætninger.

2. Parameterbeskrivelsen i FUNC-EXTERNAL skal svare nøjagtig til parameterbeskrivelsen i funktionshovedet i den SAVE'de funktion.

3. Senere i programmet kan man referere til den externe funktion som man refererer til en almindelig FUNC-ENDFUNC. Ved hvert funktionskald af den externe funktion bliver den indlæst fra disketten og fjernet igen efter brug.

4. Den externe funktion kan indeholde andre procedurer og funktioner, både åbne, lukkede samt externe.

5. Hvis en extern funktion kalder sig selv, indlæses den ikke igen.

6. Stopper programudførelsen under udførelsen af en extern funktion, er det kun denne, man kan se med LIST-kommandoen. SAVE-kommandoen gemmer da også kun den externe funktion, hvorimod RUN genstarter hovedprogrammet. NEW sletter både hovedprogram og den externe funktion.

7. Externe funktioner gør det muligt at oprette biblioteker af funktioner, der kan benyttes af flere forskellige programmer.

8. Se desuden afsnit 8.5 om externe procedurer.

13.48 GET$

RcComal80 funktion

Format
GET$ (strømnr, nudtr)

strømnr : et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm
nudtr : et vilkårligt positivt numerisk udtryk

Anvendelse
Funktionen anvendes til at læse et bestemt antal tegn (nudtr) fra en diskettefil eller en ydre enhed.

Bemærkning
1. Funktionen benyttes til at læse "formatløst" i en datastrøm.

Eksempel 1
Dette lille program udskriver indholdet af en diskettefil. Brugeren har mulighed for at vælge en ren hexadecimal udskrift, eller vælge mellem tekst- og hexadecimal-udskrift.

```
0010 DIM unit$ OF 1, navn$ OF 11, hextal$ OF 16
0020 DIM buf$ OF 512, svar$ OF 3
0030 hextal$:= "0123456789ABCDEF"
0040 INPUT "Unitnr : ": unit$
0050 INPUT "Filnavn: ": navn$
0060 REPEAT
0070   INPUT "Tekst og Hex-udskrift : ": svar$
0080 UNTIL svar$="ja" OR SVAR$="nej"
0090 tekst:= (svar$="ja")
0100 OPEN 1, "/"+unit$+"/"+navn$, READ
0110 recno:= 1
0120 WHILE NOT EOF(1) DO
0130   buf$:= GET$(1,512)
0140   PRINT "Fil : ";navn$;TAB(40);"recordnr : ";recno
0150   FOR i:= 1 TO 512 DO
0160     IF (i-1) MOD 30=0 THEN
0170       PRINT
0180       PRINT USING "§§§§ ":i;
0190     ENDIF
0200     IF i MOD 2=1 THEN PRINT " ";
```

```
0210      IF tekst AND (buf$(i:i)>=" " AND buf$(i:i) <="å") THEN
0220        PRINT ".";buf$(i:i);
0230      ELSE
0240        i1:=ord(buf$(i:i)) DIV 16 + 1
0250        i2:=ord(buf$(i:i)) MOD 16 + 1
0260        PRINT hextal$(i1:i1);hextal$(i2:i2);
0270      ENDIF
0280    NEXT i
0290    PRINT
0300    PRINT
0310    recno:= recno+1
0320  ENDWHILE
0330  CLOSE
0340  END
```

Eksempel 2
Kopieringsprogram

```
0010 DIM ifil$ OF 20, ufil$ OF 20, buf$ OF 512
0020 INPUT "INPUTFIL: ": ifil$
0030 INPUT "OUTPUTFIL: ": ufil$
0040 MARGIN 0
0050 OPEN 1, ifil$, READ
0060 OPEN 2, ufil$, WRITE
0070 WHILE NOT EOF(1) DO
0080   PRINT FILE 2: GET$(1,512);
0090 ENDWHILE
0100 CLOSE
0110 END
```

13.49 GLOBAL

RcComal80 sætning

Format
GLOBAL navn [,navn] ...

navn : navn på en variabel, array, streng, strengtabel, etikette, procedure eller funktion.

Anvendelse
Sætningen anvendes kun i FUNC-ENDFUNC og PROC-ENDPROC, der er CLOSED. Sætningen benyttes til at angive hvilke globale variable der skal kunne refereres til i en lukket procedure eller funktion.

13.50 GOTO

RcComal80 sætning

Format
GOTO navn

navn : Navnet på en etikette (se 13.39)

Anvendelse
Sætningen anvendes til at foretage et hop i programudførelsen.

Bemærkninger
1. Når GOTO udføres, bliver den næste sætning, der skal udføres, sætningen efter etiketten navn.

2. Det er ikke tilladt at hoppe ind i strukturerne IF-THEN-ELSE-ENDIF, CASE-ENDCASE, FOR-NEXT, REPEAT-UNTIL, WHILE-ENDWHILE, FUNC-ENDFUNC, PROC-ENDPROC og PACKAGE-ENDPACKAGE.

3. Man må ikke hoppe ud af FUNC-ENDFUNC, PROC-ENDPROC og PACKAGE-ENDPACKAGE.

Eksempel

```
0010 DIM svar$ OF 3
0020 WHILE TRUE DO
0030    INPUT "Ønsker du at fortsætte (ja/nej) : ": svar$
0040    IF svar$="nej" THEN GOTO slut
0050 ENDWHILE
0060 slut:
0070 END

RUN

Ønsker du at fortsætte (ja/nej) : ja
Ønsker du at fortsætte (ja/nej) : nej
END
AT 0070
```

13.51 GPARM

RcComal80 grafik funktion

Format
GPARM(nudtr)

nudtr : et numerisk udtryk

Anvendelse
Funktionen returnerer information om det grafik-system, der er åbnet.

Funktion	Information
GPARM(0)	Aktuel x-koordinat
GPARM(1)	Aktuel y-koordinat
GPARM(2)	Aktuelt vindues nedre x-grænse
GPARM(3)	Aktuelt vindues øvre x-grænse
GPARM(4)	Aktuelt vindues nedre y-grænse
GPARM(5)	Aktuelt vindues øvre y-grænse
GPARM(101)-GPARM(145)	Indholdet af GSX's parameterblok for ydre grafiske enhed (se ref. 4). De mest relevante oplysninger er:
GPARM(101)	Opløsning i x-retning. Hvis værdien f.eks. er 719, betyder det at der ialt kan adresseres 720 punkter.
GPARM(102)	Opløsning i y-retning. Hvis værdien f.eks. er 359, betyder det, at der ialt kan adresseres 360 punkter.
GPARM(103)	Skaleringsmuligheder: 0: Enheden er i stand til at producere præcist skalerede billeder 1: Enheden kan ikke producere præcist skalerede billeder (f.eks. skærme)
GPARM(104)	Bredden af et x-step i mikrometre
GPARM(105)	Højden af et y-step i mikrometre
GPARM(114)	Antal farver, der kan vises samtidig (mindst 2)

Bemærkninger
1. Indholdet af GPARM(101)-GPARM(145) er indholdet af intout efter man har foretaget en GSX Open Workstation-operation (se GSX Programmer's Guide for uddybning af emnet).

2. Funktionen må kun kaldes efter OPEN GRAPHICS er udført, ellers fås fejlen 0217: IKKE ÅBEN/ALLEREDE ÅBEN.

13.52 GSX

RcComal80 grafik sætning/kommando

Format
GSX kontrol ,ind ,pktind ,ud ,pktud

kontrol,ind,ud : endimensionale tal-tabeller
pktind ,pktud : todimensionale tal-tabeller

Anvendelse
Sætningen/kommandoen benyttes til at udføre GSX-kald direkte.

Bemærkninger
1. De 5 tabeller kontrol ,ind ,pktind ,ud ,pktud svarer til henholdsvis contrl, intin, ptsin, intout, ptsout beskrevet i GSX-86 Manual Set.

2. Inddata-parametre ved GSX-kaldet:

   kontrol-tabellen skal have følgende indhold:
   kontrol(1) : GSX-funktionsnummer
   kontrol(2) : antal punktpar i pktind
   kontrol(4) : antal værdier i ind
   kontrol(6-n) : afhængig af GSX-funktionsnummeret

   ind : en tabel, der beskriver inddata-parametre (afhængig af GSX-funktionsnummeret).

   pktind : en tabel, der indeholder koordinatpar
   pktind(1,1) : første punkts x-koordinat
   pktind(1,2) : første punkts y-koordinat
   pktind(2,1) : andet punkts x-koordinat
   pktind(2,2) : andet punkts y-koordinat
   etc.

   Bemærk koordinaterne skal angives indenfor det aktuelle grafik-vindue (se 13.130 WINDOW).

3. Uddata-parametre ved GSX-kaldet:

   kontrol-tabellen vil få følgende indhold:
   kontrol(1) : GSX-funktionsnummer
   kontrol(3) : antal punktpar i pktud
   kontrol(5) : antal værdier i ud
   kontrol(6-n) : afhængig af GSX-funktionsnummeret

   ud : en tabel, der beskriver uddata-parametre (afhængig af GSX-funktionsnummeret).

pktud	: en tabel, der indeholder koordinatpar
pktud(1,1)	: første punkts x-koordinat
pktud(1,2)	: første punkts y-koordinat
pktud(2,1)	: andet punkts x-koordinat
pktud(2,2)	: andet punkts y-koordinat
etc.	

Bemærk koordinaterne bliver angivet indenfor det aktuelle grafik-vindue (se 13.130 WINDOW).

4. De væsentligste GSX-kald er:

POLYLINE

kontrol(1)	: 6
kontrol(2)	: antal koordinatpar
kontrol(4)	: 0

Beskrivelse
Kaldet bevirker, at der tegnes en polylinie på den grafiske enhed. Udgangspunktet for polylinien er den første koordinat i pktind. Polylinien er afhængig af valget af farve, linietype og liniebredde. Først tegnes en linie fra det første koordinatpar i pktind til det andet koordinatpar. Derfra videre til det tredie, ect.

POLYMARK

kontrol(1)	: 7
kontrol(2)	: antal markeringer
kontrol(3)	: 0
pktind	: tabel over koordinater på markeringer (f.eks. plotter step s)
pktind(1,1)	: første markerings x-koordinat
pktind(1,2)	: første markerings y-koordinat
pktind(n,1)	: n'te markerings x-koordinat
pktind(n,2)	: n'te markerings y-koordinat

Beskrivelse
Kaldet bevirker, at der tegnes markeringer på de koordinater, der er angivet i pktind.

FILLED AREA

kontrol(1)	: 9
kontrol(2)	: antal koordinatpar
kontrol(4)	: 0

Beskrivelse
Kaldet bevirker, at der tegnes en udfyldt polygon på den grafiske enhed. Hjørnerne i polygonen angives i pktind. Polygonen er afhængig af valget af udfyldningsfarve, udfyldningsform, udfyldningsmønster.

SET CHARACTER HEIGHT

kontrol(1)	: 12
kontrol(2)	: 1
kontrol(4)	: 0
pktind(1,1)	: 0
pktind(1,2)	: ønsket tegnhøjde angivet i det aktuelle vindues y-koordinater i forhold til y_nedre.

Beskrivelse
Kaldet sætter størrelsen for tekst, der udskrives med TEXT-sætningen/kommandoen

Returværdier

pktud(1,1)	: Den valgte bredde for et tegn (tegnet W)
pktud(1,2)	: Den valgte højde for et tegn (tegnet W)
pktud(2,1)	: Den valgte bredde for en tegnblok
pktud(1,1)	: Den valgte højde for en tegnblok

SET CHARACTER UP VECTOR

kontrol(1)	: 13
kontrol(2)	: 0
kontrol(4)	: 3
ind(1)	: vinklen for rotationen (i tiendedele grader)
ind(2)	: Punkt for drejning (cos(vinkel) * 100)
ind(3)	: Punkt for drejning (sin(vinkel) * 100)

Beskrivelse
Kaldet sætter rotationer for tegn, der udskrives med TEXT-sætningen/kommandoen.

SET COLOR REPRESENTATION

kontrol(1)	: 14
kontrol(2)	: 0
kontrol(4)	: 4
ind(1)	: Farvenummer
ind(2)	: Rød farveintensitet (i tiendedele procent 0 - 1000)
ind(3)	: Grøn farveintensitet
ind(4)	: Blå farveintensitet

Beskrivelse
Kaldet angiver kombinationen af rød, grøn og blå farve for en valgt farve. Der kan angives følgende intensiteter:

0	: slukket
500	: normal styrke
1000	: kraftigt lysende

Hvis blot een farve angives kraftigt lysende, vil samtlige lysende farver blive kraftigt lysende.

SET POLYLINE LINETYPE

kontrol(1)	: 15
kontrol(2)	: 0
kontrol(4)	: 1
ind(1)	: Ønsket linietype

Beskrivelse
Kaldet bestemmer linietyper for efterfølgende POLYLINE kald. Antallet af forskellige linietyper er afhængigt af den enkelte grafiske enhed, men følgende vil altid findes:

1	: Ubrudt
2	: Stiplet
3	: Punkteret
4	: Stiplet og punkteret
5	: Lange elementer i stiplingen

SET POLYLINE LINE WIDTH

kontrol(1)	: 16
kontrol(2)	: 1
kontrol(4)	: 0
pktind(1,1)	: Ønsket stregbredde angivet i det aktuelle vindues x-koordinater i forhold til x_venstre.
pktind(1,2)	: 0

Beskrivelse
Kaldet bestemmer stregbredden for linier tegnet med efterfølgende POLYLINE kald.

SET POLYLINE COLOR INDEX

kontrol(1) : 17
kontrol(2) : 0
kontrol(4) : 1
ind(1) : Ønsket farve for polylinier

Beskrivelse
Kaldet bestemmer farven for linier tegnet med efterfølgende POLYLINE kald.

SET POLYMARKER TYPE

kontrol(1) : 18
kontrol(2) : 0
kontrol(4) : 1
ind(1) : Ønsket type polymarker

Beskrivelse
Kaldet bestemmer type på markeringen i efterfølgende POLYMARKER kald. Antallet af forskellige markeringer er afhængigt af den enkelte grafiske enhed, men følgende vil altid findes:

1 : . (Punktum)
2 : + (Plus)
3 : * (Stjerne)
4 : O (Cirkel)
5 : X (Diagonalt kryds)

SET POLYMARKER SCALE

kontrol(1) : 19
kontrol(2) : 1
kontrol(4) : 0
pktind(1,1) : 0
pktind(1,2) : ønsket polymarkerhøjde angivet i det aktuelle vindues y-koordinater i forhold til y_nedre.

Beskrivelse
Kaldet angiver højden for hver markering i efterfølgende POLYMARKER kald.

SET POLYMARKER COLOR INDEX

kontrol(1) : 20
kontrol(2) : 0
kontrol(4) : 1
ind(1) : Ønsket farve

Beskrivelse
Kaldet angiver farvenummeret for hver markering i efterfølgende POLYMARK kald.

SET TEXT FONT

kontrol(1) : 21
kontrol(2) : 0
kontrol(4) : 1
ind(1) : Ønsket maskine tekst font

Beskrivelse
Kaldet vælger en tekst font for efterfølgende TEXT-sætninger/komandoer.

SET TEXT COLOR INDEX

kontrol(1) : 22
kontrol(2) : 0
kontrol(4) : 1
ind(1) : Ønsket farve

Beskrivelse
Kaldet angiver farvenummeret, dvs. farven for efterfølgende TEXT-sætninger/komandoer.

SET FILL INTERIOR STYLE

kontrol(1) : 23
kontrol(2) : 0
kontrol(4) : 1
ind(1) : Ønsket udfyldningsform

Beskrivelse
Kaldet angiver udfyldningen ved efterfølgende FILLED AREA kald. Der kan vælges mellem følgende værdier:

0 : Hul
1 : Masiv
2 : Halvtone mønster
3 : Skravering

SET FILL STYLE INDEX

kontrol(1) : 24
kontrol(2) : 0
kontrol(4) : 1
ind(i) : Ønsket halvtone mønster eller skraverings mønster

Beskrivelse
Kaldet angiver det mønster, der skal udfyldes med ved efterfølgende FILLED AREA kald.
For halvtone mønster vil 1 være det lyseste og 6 det mørkeste.
For skraverings mønster kan der vælges imellem:

1 : Vertikale linier
2 : Horisontale linier
3 : +45 grader linier
4 : -45 grader linier
5 : Kryds (vertikale,horisontale linier)
6 : X (diagonale linier)
>6 : Afhængig af den grafiske enhed

SET FILL COLOR INDEX

kontrol(1) : 25
kontrol(2) : 0
kontrol(4) : 1
ind(1) : Ønsket farve

Beskrivelse
Kaldet angiver farvenummeret, dvs. farven for efterfølgende kald af FILLED AREA.

Eksempel 1

```
0010 PROC kasse(x0,x1,y0,y1)
0020   pind(1,1):= x0; pind(1,2):= y0; pind(2,1):= x1; pind(2,2):= y0
0030   pind(3,1):= x1; pind(3,2):= y1; pind(4,1):= x0; pind(4,2):= y1
0040   pind(5,1):= x0; pind(5,2):= y0
0050   kaldgsx(6,5,0,ind,pind,ud,pud)
0060 ENDPROC kasse
0070
0080 PROC kaldgsx(fkt,ind,pkt,REF i(),REF pin(,),REF u(),REF pud(,))
0090   kont(1):= fkt; kont(2):= ind; kont(4):= pkt
0100   GSX kont,i,pin,u,pud
0110 ENDPROC kaldgsx
0120
0130 DIM kont(10),pind(10,2),pud(10,2),ind(10),ud(10)
0140 PRINT "Tegning af søjlediagram"
0150 PRINT
```

```
0160 INPUT "Indtast antal værdier : ": antal
0170 DIM værdi(antal)
0180 max:= 0
0190 FOR i:= 1 TO antal DO
0200   INPUT "Indtast værdi (positiv) : ": værdi(i)
0210   IF værdi(i)>max THEN max:= værdi(i)
0220 NEXT i
0230 OPEN GRAPHICS 1
0240 WINDOW 0,antal+1.5,-max/10,1.1*max
0250 kasse(0,antal+1.5,0,max)
0260 FOR i:= 1 TO antal DO
0270   kasse(i,i+0.8,0,værdi(i))
0280 NEXT i
```

Eksempel 2

```
0010 PROC kasse(x0,x1,y0,y1)
0020   pind(1,1):= x0; pind(1,2):= y0; pind(2,1):= x1; pind(2,2):= y0
0030   pind(3,1):= x1; pind(3,2):= y1; pind(4,1):= x0; pind(4,2):= y1
0040   pind(5,1):= x0; pind(5,2):= y0
0050   kaldgsx(6,5,0,ind,pind,ud,pud)
0060 ENDPROC kasse
0070
0080 PROC fyldt_kasse(x0,x1,y0,y1)
0090   pind(1,1):= x0; pind(1,2):= y0; pind(2,1):= x1; pind(2,2):= y0
0100   pind(3,1):= x1; pind(3,2):= y1; pind(4,1):= x0; pind(4,2):= y1
0110   pind(5,1):= x0; pind(5,2):= y0
0120   kaldgsx(9,5,0,ind,pind,ud,pud)
0130 ENDPROC fyldt_kasse
0140
0150 PROC kaldgsx(fkt,ind,pkt,REF i(),REF pin(,),REF u(),REF pud(,))
0160   kont(1):= fkt; kont(2):= ind; kont(4):= pkt
0170   GSX kont,i,pin,u,pud
0180 ENDPROC kaldgsx
0190
0200 DIM kont(10),pind(10,2),pud(10,2),ind(10),ud(10)
0210 PRINT "Tegning af søjlediagram"
0220 PRINT
0230 INPUT "Antal forskellige søjletyper (max 6) : ": typ
0240 INPUT "Indtast antal værdier pr. søjletype : ": antal
0250 DIM værdi(typ,antal)
0260 max:= 0
0270 FOR søjle:= 1 TO typ DO
0280   PRINT "Værdier for sæt nr ";søjle
0290   FOR i:= 1 TO antal DO
0300     INPUT "Indtast værdi (positiv) : ": værdi(søjle,i)
0310     IF værdi(søjle,i)>max THEN max:= værdi(søjle,i)
0320   NEXT i
0330 NEXT søjle
```

```
0340 OPEN GRAPHICS 1
0350 WINDOW 1,antal+1.5,-max/10,1.1*max
0360 kasse(1,antal+1.5,0,max)
0370 FOR s:= 1 TO typ DO
0380   ind(1):= 3 // halvtone mønster
0390   kaldgsx(23,0,1,ind,pind,ud,pud)
0400   ind(1):= s // halvtone mønster
0410   kaldgsx(24,0,1,ind,pind,ud,pud)
0420   x:= s/(typ+1); bredde:= 0.8/typ
0430   FOR i:= 1 TO antal DO
0440     fyldt_kasse(i+x,i+bredde+x,0,værdi(s,i))
0450   NEXT i
0460 NEXT s
```

Eksempel 3

```
0010 PROC kaldgsx(fkt,ind,pkt,REF i(),REF pin(,),REF u(),REF pud(,))
0020   kont(1):= fkt; kont(2):= ind; kont(4):= pkt
0030   GSX kont,i,pin,u,pud
0040 ENDPROC kaldgsx
0050
0060 DIM kont(10),pind(10,2),pud(10,2),ind(10),ud(10)
0100 OPEN GRAPHICS 1
0110 WINDOW 0,50,0,50
0120 FOR højde:= 1 TO 5 DO
0130   pind(1,1):= 0; pind(1,2):= højde
0140   kaldgsx(12,1,0,ind,pind,ud,pud) // Tegn højden indstilles
0150   MOVETO 0,højde*8
0160   TEXT "Tekst test"
0170 NEXT højde
```

Eksempel 4

```
0010 PROC kaldgsx(fkt,ind,pkt,REF i(),REF pin(,),REF u(),REF pud(,))
0020   kont(1):= fkt; kont(2):= ind; kont(4):= pkt
0030   GSX kont,i,pin,u,pud
0040 ENDPROC kaldgsx
0050
0060 DIM kont(10),pind(10,2),pud(10,2),ind(10),ud(10)
0070 OPEN GRAPHICS 1
0080 WINDOW -10,10,-10,10
0090 FOR vinkel:= 0 TO 360 STEP 45 DO
0100   ind(1):= vinkel*10
0110   ind(2):= COS(vinkel*PI/180)*100
0120   ind(3):= SIN(vinkel*PI/180)*100
0130   kaldgsx(13,0,3,ind,pind,ud,pud)
0140   MOVETO 0,0
0150   TEXT " ->Rotation"
0160 NEXT vinkel
```

13.53 IF - THEN

RcComal80 sætning

Format
IF logisk udtryk THEN simpel sætning

logisk udtryk : et udtryk der, når det udregnes, har værdien sand (<>0) eller falsk (=0).

simpel sætning : en simpel RcComal80 sætning, dvs. CIRCLE, CLEAR, CLOSE, CLOSE GRAPHICS, CONTINUE, CREATE, DELETE, DRAW, DRAWTO, END, EXEC, GOTO, GPARM, GSX, INPUT, INPUT FILE, LOCATE, MOVE, MOVETO, OPEN FILE, OPEN GRAPHICS, PALETTE, PENCOLOR, PRINT, PRINT FILE, PRINT USING, READ, READ FILE, RESTORE, RETRY, RETURN, SELECT OUTPUT, STOP, TEXT, tildeling, WINDOW, WRITE FILE.

Anvendelse
Bruges til at betinge udførelsen af en sætning.

Virkemåde
1. Hvis logisk udtryk er sand (<>0), bliver simpel sætning udført. Hvis simpel sætning ikke bevirker hop i programmet, vil programudførslen fortsætte med første programlinie efter IF-THEN sætningen.

2. Hvis værdien af logisk udtryk er falsk (=0), vil simpel sætning ikke blive udført og programudførelsen fortsætter med første linie efter IF-THEN sætningen.

Eksempel

```
0010 // Simpel forgrening
0020 INPUT i, j
0030 PRINT "De to tal er ";
0040 IF i<>j THEN PRINT "ikke ";
0050 PRINT "ens"
0060 END
```

13.54 IF - THEN - ENDIF

RcComal80 struktur

Format
IF logisk udtryk THEN
  sætningsliste
ENDIF

logisk udtryk : et udtryk der, når det udregnes, har værdien sand (<>0) eller falsk (=0).
sætningsliste : RcComal80 sætninger.

Anvendelse
Konstruktionen bruges til at gøre udførelsen af en blok sætninger betinget af, om et udtryk er sandt eller falsk.

Virkemåde
1. Hvis værdien af logisk udtryk er sand (<>0), vil sætningsliste blive udført een gang.

2. Programudførelsen vil fortsætte med den første sætning efter ENDIF sætningen hvis man ikke foretager hop ud af strukturen.

Bemærkning
1. Sætningen kan ikke udføres som kommando.

Eksempel

```
0010 // IF-THEN-ENDIF
0020 INPUT i,j
0030 IF i<>j THEN
0040    PRINT "De to tal er ikke ens."
0050    PRINT "Forskellen er ";ABS(i-j)
0060 ENDIF
0070 END
```

13.55 IF - THEN - ELSE - ENDIF

RcComal80 struktur

Format
IF logisk udtryk THEN
  sætningsliste-1
ELSE
  sætningsliste-2
ENDIF

logisk udtryk	:	et udtryk der, når det udregnes, har værdien sand (<>0) eller falsk (=0).
sætningsliste-1	:	en række RcComal80 sætninger der vil blive udført, hvis værdien af logisk udtryk er sand (<>0).
sætningsliste-2	:	en række RcComal80 sætninger der vil blive udført, hvis værdien af logisk udtryk er falsk (=0).

Anvendelse
Konstruktionen benyttes til at udføre en af to blokke RcComal80 sætninger, afhængig af værdien af et logisk udtryk.

Virkemåde
1. logisk udtryk udregnes.
2. Hvis værdien af logisk udtryk er sand (<>0) udføres sætningsliste-1
3. Hvis værdien af logisk udtryk er falsk (=0) udføres sætningsliste-2
4. Når sætningsliste-1 eller sætningsliste-2 er blevet udført, og der ikke er foretaget hop ud af listen, fortsættes programudføreisen med første sætning efter ENDIF.

Bemærkning
1. Hvis man hopper ind i sætningsliste-1 eller sætningsliste-2 udenom IF-THEN-ELSE, udskrives fejlmeddelelsen 0116: ULOVLIGT HOP.

Eksempel

```
0010 REPEAT
0020   READ tal1,tal2
0030   PRINT tal1;"+ ";tal2;
0040   INPUT "= ?": svar
0050   IF svar=tal1+tal2 THEN
0060     PRINT "Det er korrekt"
0070   ELSE
```

```
0080       PRINT "Det er forkert"
0090       PRINT "Det rigtige svar er: ";tal1+tal2
0100     ENDIF
0110 UNTIL EOD
0120 DATA 2,0,3,1,5,3

RUN
2 + 0 = ? 2
Det er korrekt
3 + 1 = ? 5
Det er forkert
Det rigtige svar er: 4
5 + 3 = ? 8
Det er korrekt
END
AT 0120
```

13.56 IMPORT

RcComal80 sætning

Format
IMPORT navn [, navn]...

navn : navn på en variabel, array, streng, strengtabel, etikette, procedure eller funktion

Anvendelse
Sætningen anvendes kun i FUNC-ENDFUNC og PROC-ENDPROC, der er CLOSED. Sætningen benyttes til at angive hvilke variable, der optræder på det niveau hvor kaldet foretages, der skal kunne refereres til i en lukket procedure eller funktion (se endvidere afsnit 8.4).

13.57 IN

RcComal80 operator

Format
sudtr1 IN sudtr2

sudtr1 : strengudtryk
sudtr2 : strengudtryk

Operatorprioritet = 5 (se afsnit 4.6.1)

Anvendelse
Operatoren finder positionen for sudtr1 i sudtr2. Resultatet er et ikke-negativt heltal.

Bemærkninger
1. Hvis sudtr1 findes i sudtr2 returnerer operatoren positionen for sudtr1's første tegn i sudtr2.

2. Hvis sudtr1 ikke findes i sudtr2 returneres værdien 0.

3. Hvis sudtr1 er tom, returneres med længden af sudtr2 plus 1.

Eksempel 1

```
PRINT "ka" IN "abekat"
4
```

Eksempel 2

```
PRINT "" IN "abekat"
7
```

Eksempel 3

```
PRINT "hund" IN "abekat"
0
```

Eksempel 4

```
0010 DIM ifil$ OF 20,ufil$ OF 20
0020 DIM fra$ OF 132,til$ OF 132,l$ OF 132
0030 PRINT "Programmet retter den samme tekst"
0040 PRINT "alle steder i en tekstfil."
0050 PRINT "Dog kun en rettelse pr. linie"
0060 INPUT "Inputfil :": ifil$
0070 INPUT "Outputfil :": ufil$
0080 INPUT "Hvilken tekst skal ændres ?": fra$
0090 INPUT "Ændres til hvilken tekst ?": til$
0100 OPEN 1,ifil$, READ
0110 OPEN 2,ufil$, WRITE
0120 forsk:=LEN(til$)-LEN(fra$)
0130 WHILE NOT EOF(1) DO
0140   INPUT FILE 1:l$
0150   IF NOT EOF(1) THEN
0160     p:=(fra$ IN l$)
0170     IF p<>0 THEN
0180       l$(p:len(l$)+forsk):=til$+l$(p+LEN(fra$):LEN(l$))
0190     ENDIF
0200   ENDIF
0210   PRINT FILE 2: l$
0220 ENDWHILE
0230 CLOSE
0240 END
```

13.58 INPUT

RcComal80 sætning

Format

INPUT [skonst :] { nvar / svar } [, { nvar / svar }] ... [;]

INPUT AT (nudtr1,nudtr2) [, skonst:] { nvar / svar } [, { nvar / svar }] ... [;]

skonst : en strengkonstant
nvar : en numerisk variabel
svar : en strengvariabel
nudtr1 : et numerisk udtryk med værdi i intervallet 1 <= nudtr1 <= 80
nudtr2 : et numerisk udtryk med værdi i intervallet 1 <= nudtr2 <= 25

Anvendelse

Sætningen bruges til at tildele værdier til variable fra tastaturet, når et program kører.

Virkemåde

1. Hvis AT(nudtr1,nudtr2) er angivet, flyttes udskriften til den tilhørende position på skærmen (se 13.3 AT).

2. Hvis skonst er angivet udskrives denne.

3. Cursoren fremkommer på skærmen, som tegn på at systemet forventer input.

4. Herefter kan brugeren indtaste værdien for første INPUT-element.

5. Skal en numerisk variabel indtastes, kan indtastningen af den afsluttes med vognretur, eller, hvis der er yderligere INPUT-elementer, med et komma eller et mellemrum.

6. Skal en strengvariabel indtastes skal indtastningen afsluttes med vognretur.

7. Hvis der er flere INPUT-elementer, fortsættes med det næste element (hop til trin 5).

8. Hvis INPUT-sætningen afsluttes med et semikolon (;) vil efterfølgende udskrift fortsættes umiddelbart efter det indtastede, ellers udføres en vognretur.

13.59 INPUT FILE

RcComal80 sætning/kommando

Format

INPUT FILE strømnr: { nvar / svar } [, { nvar / svar }] ...

strømnr : et taludtryk, hvis værdi angiver nummeret på en åben datastrøm.
nvar : en numerisk variabel
svar : en strengvariabel

Anvendelse

Sætningen indlæser data i ASCII-format fra en datastrøm, der er åbnet (OPENed) i READ-mode.

Bemærkninger

1. Hvert argument i INPUT FILE-sætningen skal have samme type (numeriske eller strenge) som dataene i datastrømmen, ellers fås fejl 0109: TYPEKONFLIKT.

2. Hvis datastrømmen er en diskette-fil, skal den være skrevet ved hjælp af PRINT FILE, SELECT OUTPUT (og DIR, LIST, PRINT etc.).

3. Input datastrømmen skal være opbygget således at tegnet CR (se appendix D) adskiller de enkelte elementer.

4. Hvis længden af en streng i datastrømmen er større end den dimensionerede længde af den tilsvarende streng-variabel, vil de overskydende tegn blive ignoreret.

5. I øvrigt henvises til kapitel 6: Indlæsning og udskrivning.

Eksempel

```
0010 DIM ifil$ OF 17,ufil$ OF 17,l$ OF 132
0020 INPUT "Listning af hvilket program ? ": ifil$
0030 INPUT "Listning hvorhen ? ":ufil$
0040 INPUT "Sidehøjde ? ": sidehøjde
0050 OPEN FILE 1,ifil$,READ
0060 SELECT OUTPUT ufil$
0070 lnr:=0; side:=0
0080 EXEC sideskift
```

```
0090 WHILE NOT EOF(1) DO
0100   INPUT FILE 1: l$
0110   IF NOT EOF(1) THEN
0120     PRINT l$
0130     lnr:=lnr+1
0140     if lnr=sidehøjde THEN EXEC sideskift
0150   ENDIF
0160 ENDWHILE
0170 CLOSE
0180 SELECT OUTPUT "console"
0190 END
0200 PROC sideskift
0210   side:=side+1; lnr:=1
0220   PRINT CHR$(12);TAB(70);"SIDE ";side
0230 ENDPROC sideskift
```

13.60 INT

RcComal80 funktion

Format
INT(nudtr)

nudtr : et vilkårligt numerisk udtryk

Anvendelse
Funktionen udregner det nærmeste heltal, der ikke er større end nudtr.

Eksempel

```
0010 FUNC afrund(x)// afrunder efter normale regler
0020   return INT(x+0.5)
0030 ENDFUNC afrund
0040
0050 MARGIN 80
0060 ZONE 20
0070 PRINT "Tal", "Heltalsværdi", "Afrundet"
0080 RANDOMIZE
0090 FOR i:= 1 TO 10 DO
0100   tal:= RND*100-50
0110   PRINT tal, INT(tal), afrund(tal)
0120 NEXT i
0130 END
```

```
RUN
Tal                 Heltalsværdi     Afrundet
38.8569663724         38               39
 2.98225784899         2                3
-1.55790098222        -2               -2
-21.79328204523      -22              -22
 13.90425481596        13               14
-34.64199176265      -35              -35
-27.57420626506      -28              -28
-12.49762589527      -13              -12
 43.28272182272        43               43
-25.63532655509      -26              -26
END
AT 0230
```

13.61 KEY$

RcComal80 funktion

Format
KEY$

Anvendelse
Funktionen returnerer med værdien af den sidste tast, der er trykket ned på tastaturet.

Bemærkninger
1. Hvis ingen tast har været rørt siden sidste INPUT, GET$ eller KEY$ returneres med værdien CHR$(0).

2. Proceduren venter således ikke på, at der trykkes på en tast. Hvis brugeren ønsker dette, skal han åbne en datastrøm til tastaturet (keyboard) og foretage kald af GET$.

3. Da mellemrumstasten kan bruges til midlertidig afbrydelse af udførelsen af et program, kan der opstå uhensigtsmæssigheder, hvis man venter på at KEY$ skal antage værdien " " (se kommandoen RUN, 13.105).

Eksempel
Denne funktion venter en bestemt tid på et tegn fra tastaturet og returnerer, hvis tiden er gået, eller der er modtaget et tegn.

```
0010 FUNC tchar(tid) CLOSED
0020   nu:=SYS(3)
0030   REPEAT
0040     i:=ORD(KEY$)
0050   UNTIL SYS(3)>nu+tid OR i<>0
0060   RETURN i
0070 ENDFUNC tchar
```

13.62 LEN

RcComal80 funktion

Format
LEN(sudtr)

sudtr : et vilkårligt strengudtryk

Anvendelse
Funktionen returnerer den aktuelle længde (antal tegn) af strengen sudtr.

Bemærkninger
1. LEN-funktionen må optræde i et vilkårligt numerisk udtryk.

2. Hvis sudtr er den tomme streng, returneres med værdien 0.

3. Bemærk, at funktionen returnerer med det aktuelle antal tegn i strengen og ikke med det antal, en strengvariabel er dimensioneret til.

Eksempel

```
0010 DIM tekst$ OF 80, konsonant$ OF 20, vokal$ OF 9
0020 konsonant$:= "bcdfghjklmnpqrstvwxz"
0030 vokal$:= "aeiouyæøå"
0040 INPUT "Indtast tekst (små bogstaver):": tekst$
0050 antkons:= 0; antvokal:= 0
0060 FOR i:= 1 TO LEN(tekst$) DO
0070   IF tekst$(i:i) IN konsonant$ THEN antkons:= antkons+1
0080   IF tekst$(i:i) IN vokal$ THEN antvokal:= antvokal+1
0090 NEXT i
0100 PRINT tekst$
0110 PRINT "Indeholder ";antkons;"konsonanter"
0120 PRINT "og ";antvokal;"vokaler."
0130 END
```

13.63 LIST

RcComal80 kommando

Format

```
                navn
                lnr1 [,]
LIST  [ [ {     ,  lnr2  } ]  ]   [filnavn]
                lnr1,lnr2
```

navn : navn på en procedure, funktion eller pakke, der skal listes
lnr1 : linienummeret på den første sætning der skal listes
lnr2 : linienummeret på den sidste sætning der skal listes
filnavn : navnet på en datastrøm angivet i anførselstegn

Anvendelse
Kommandoen anvendes til at udskrive hele eller dele af det program, der ligger i programlageret.

Bemærkninger
1. Angives filnavn, udskrives programmet på filnavn istedet for på skærmen. Programmet gemmes i ASCII-format og skal hentes ind i lageret igen ved ENTER-kommandoen.

2. Kombinationerne af parametrene til LIST-komamndoen har følgende betydning:

LIST	Udskriver hele programmet i programlageret.
LIST navn	Udskriver den struktur, der er defineret til at hedde navn
LIST lnr1	Udskriver linien med linienummeret lnr1
LIST lnr1,	Udskriver alle de linier, hvor linienummeret >= lnr1
LIST , lnr2	Udskriver alle de linier, hvor linienummeret <= lnr2
LIST lnr1,lnr2	Udskriver alle de linier, hvor lnr1 <=linienummeret <= lnr2

3. Ønskes udskriften standset midlertidigt, trykkes på mellemrumstangenten. Ved næste tryk genstartes udskriften.

4. Udskriften afbrydes ved tryk på ESC-tasten.

Eksempel 1
Hvis vi i det følgende forudsætter, at programlageret indeholder følgende program:

```
0010 a := 7
0020 PROC p(tal)
0030     PRINT "tal er lig : "
0040     PRINT tal
0050 ENDPROC p
0060 p(a)
```

Da vil:

1.	LIST	udskrive linierne 0010, 0020, 0030, 0040, 0050, 0060
2.	LIST p	udskrive linierne 0020, 0030, 0040, 0050
3.	LIST 20	udskrive linien 0020
4.	LIST 40,	udskrive linierne 0040, 0050, 0060
5.	LIST ,20	udskrive linierne 0010, 0020
6.	LIST 30,40	udskrive linierne 0030, 0040

Eksempel 2
Ønskes programmet i programlageret udskrevet på printeren udføres en af kommandoerne

```
LIST "printer"
LIST "printer0"
LIST "printer1"
LIST "printer2"
```

13.64 LISTPACK

RcComal80 kommando

Format
LISTPACK pakkenavn

pakkenavn : Navn på en bestemt pakke

Anvendelse
Kommandoen udskriver en liste over, hvilke procedurer og funktioner, pakken stiller til rådighed, samt hvilke parametre disse procedurer og funktioner kræver.

Bemærkning
1. Kommandoen kan kun anvendes på de pakkenavne, der vises ved kommandoen SHOWPACK.

Eksempel

```
SHOWPACK
  PACKAGE bitop

LISTPACK bitop
  PACKAGE bitop
    FUNC bitand(arg1,arg2)
    FUNC bitor(arg1,arg2)
    FUNC bitxor(arg1,arg2)
    FUNC bitnot(arg1)
    FUNC bitshl(arg1,noofbits)
    FUNC bitshr(arg1,noofbits)
  ENDPACKAGE bitop
```

13.65 LOAD

RcComal80 kommando

Format
LOAD [filnavn]

filnavn : navnet på en diskettefil angivet i anførselstegn

Anvendelse
Kommandoen bruges til at hente et program fra en disk- eller kassettebåndsfil. Programmet skal være gemt med SAVE-kommandoen.

Bemærkning
1. LOAD udfører automatisk en NEW-kommando, som bl.a. lukker evt. åbne filer og fjerner evt. pakker.

2. Udelades filnavn vil kommandoen hente indholdet af den fil, der er angivet i statuslinien.

3. Efter LOAD vil statuslinien indeholde navnet på den fil, der er LOADed.

4. Hvis der ikke angives nogen type i filnavnet, tilføjes automatisk .CSV til navnet.

Eksempel 1
```
LOAD "/2/MITPROG"
```

Eksempel 2
```
LOAD "TIPS"
```

13.66 LOADPACK

RcComal80 kommando

Format
LOADPACK navn [FROM filnavn]

navn : navn på pakken
filnavn : navn på en diskettefil angivet i anførselstegn

Anvendelse
Kommandoen bruges til at hente en RcComal80-pakke fra en disk- eller kassettebåndsfil ind i programlageret. Derefter er det muligt at rette i pakken. Pakken skal være gemt med SAVEPACK-kommandoen.
Hvis pakken er beskyttet med en PASSWORD-sætning, skal det korrekte løsen angives, før pakken kan hentes ind.

Bemærkninger
1. LOADPACK udfører automatisk en NEW-kommando, som lukker evt. åbne filer.

2. Udelades FROM filnavn bruges navn til navnet på diskettefilen.

3. Hvis der ikke angives nogen type i filnavnet, tilføjes automatisk .PCK til navnet.

4. Efter LOADPACK vil statuslinien indeholde navnet på den fil, der er LOADPACKed.

Eksempel
Følgende kommando henter en RcComal80-pakke fra filen VINDUE.PCK

```
LOADPACK vindue
```

13.67 LOCATE

RcComal80 grafik sætning/kommando

Format
LOCATE tast , xkoor , ykoor

tast : en stregnvarlabel
xkoor,ykoor : numeriske variable

Anvendelse
Sætningen/kommandoen benyttes til at læse en positionskode ved grafisk input (dvs. ved benyttelse af digitizer eller mus).

Bemærkninger
1. Ved kald af LOCATE vises "trådkorset" i det punkt der angives med xkoor,ykoor.

2. Man skal huske at benytte en GSX-skærmdriver, der indeholder den ønskede form for grafisk input. Se i manualen PICCOLINE Installation og Vedligeholdelse eller Partner Installation og Vedligeholdelse.

3. LOCATE skal altid afsluttes med tryk på en af tasterne på digitizeren eller musen. Hvilken tast, der er nedtrykket er angivet i strengvariablen tast. Hvis tast = CHR$(32) er det tast nr. 1, hvis tast = CHR$(33) er det tast nr. 2, etc.

4. Benyttes tastaturet som grafisk input enhed skal man være opmærksom på følgende:
   - Piltasterne anvendes til at flytte "trådkorset", A1-tasten ændrer længden af et trin.
   - Funktionstasterne mister deres "normale" værdier ved kald af LOCATE. Funktionstasterne reinitialiseres ved programafslutning eller med tegnsekvensen CHR$(27)+"7"
   - tast angiver den tast på tastaturet, der nedtrykkes for at markere et punkt

Eksempel

```
0010 OPEN GRAPHICS 1
0020 PRINT "Tegn figurer på skærmen"
0030 PRINT
0040 PRINT "Angiv punkter med mellemrum, afslut med I"
0050 DIM t$ OF 1
0060 x:= 0; y:= 0
0070 WINDOW -1,1,-1,1
```

```
0080 LOCATE t$,x,y
0090 MOVETO x,y
0100 REPEAT
0110    LOCATE t$,x,y
0120    DRAWTO x,y
0130 UNTIL t$="l"
```

13.68 LOG

RcComal80 funktion

Format
LOG (nudtr)

nudtr : et positivt numerisk udtryk

Anvendelse
Funktionen udregner den naturlige logaritme af nudtr.

Eksempel

```
0010 FUNC log10(x)// fkt. udregner ti-tals logaritmen til x
0020   RETURN LOG(x)/LOG(10)
0030 ENDFUNC log10
0040
0050 ZONE 20
0060 PRINT "x","nat.log","titalslog."
0070 FOR i:= 1 TO 10 DO
0080   tal:= RND*200
0090   PRINT tal,LOG(tal),log10(tal)
0100 NEXT I
0110 END

RUN
          x                nat.log                titalslog.
 22.32006388358   3.105485999676   1.348695433287
167.8980906691    5.123357192184   2.225045757385
164.0659968311    5.100268766578   2.215018581549
  7.41540988152   2.003560251407   0.8701351613471
129.5512502923    4.864076558003   2.112441608696
177.4245282954    5.178545325793   2.249013659278
  8.59768468306   2.15149294411    0.9343815134811
161.549995607     5.084814665078   2.208306950545
174.0521613794    5.159355032333   2.240679420723
134.8887992722    4.904450729909   2.129975888766
END
AT 0110
```

13.69 MARGIN

RcComal80 sætning/kommando

Format
MARGIN nudtr

nudtr : Vilkårligt ikke negativt numerisk udtryk

Anvendelse
Sætningen/kommandoen sætter højre-margen på udskriftslinien.

Bemærkninger
1. Standardværdien ved opstart er MARGIN 0.

2. Benyttes MARGIN 0, vil systemet på intet tidspunkt tilføje et linieskift i udskriften. Dette bør angives i forbindelse med skærmadressering (AT-funktionen).

3. Efter NEW udføres automatisk MARGIN 0.

4. Hvis nudtr er negativ, bevares den forrige højremargen.

Eksempel

```
0010 MARGIN 40
0020 ZONE 5
0030 FOR i:= 1 TO 10 DO PRINT i,
0040 PRINT
0050 MARGIN 30
0060 FOR i:= 1 TO 10 DO PRINT i,
0070 PRINT
0080 END

RUN
1    2    3    4    5    6    7    8
9    10
1    2    3    4    5    6
7    8    9    10
END
AT 0080
```

13.70 MOD

RcComal80 operator

Format
nudtr1 MOD nudtr2

nudtr1 : et vilkårligt numerisk udtryk opfattet som heltalsvariabel
nudtr2 : et vilkårligt numerisk udtryk forskellig fra nul opfattet som heltalsvariabel

Operatorprioritet = 3 (se afsnit 4.6.1)

Anvendelse
Operatoren udregner resten ved en heltalsdivision.

Bemærkninger
1. Definitionen på a MOD b er lig a-INT(a/ABS(b))*ABS(b).

2. RcComal80 checker ikke om nudtr1 og nudtr2 er heltalsudtryk. Hvis de ikke er det, kan man få en ikke-heltallig rest.

Eksempel

```
0010 FUNC sfd(m,n)
0020   IF n=0 THEN
0030     RETURN m
0040   ELSE
0050     RETURN sfd(n,m MOD n)
0060   ENDIF
0070 ENDFUNC sfd
0080
0090 ZONE 20
0100 WHILE NOT EOD DO
0110   READ tal1,tal2
0120   PRINT tal1,tal2,sfd(tal1,tal2)
0130 ENDWHILE
0140 DATA 7,14,60,24,13,19

RUN
7                   14                   7
60                  24                  12
13                  19                   1
END
AT 00140
```

13.71 MOVE

RcComal80 grafik sætning/kommando

Format
MOVE xkoor , ykoor

xkoor,ykoor : numeriske udtryk

Anvendelse
Sætningen/kommandoen flytter det løbende punkt (x,y) til punktet (x+xkoor,y+ykoor). MOVE flytter altså relativt fra det løbende punkt.

Eksempel

```
0010 PROC kasse(side1,side2)
0020   DRAW side1,0
0030   DRAW 0,side2
0040   DRAW -side1,0
0050   DRAW 0,-side2
0060 ENDPROC kasse
0070
0080 OPEN GRAPHICS 1
0090 WINDOW -10,10,-10,10
0105 MOVETO 0,0
0110 FOR i:=1 TO 10 DO
0120   MOVE -1,-1 // nederste venstre hjørne af kasse
0130   EXEC kasse(2*i,2*i)
0140 NEXT i
```

13.72 MOVETO

RcComal80 grafik sætning/kommando

Format
MOVETO xkoor , ykoor

xkoor,ykoor : numeriske udtryk

Anvendelse
Sætningen/kommandoen flytter det løbende punkt (x,y) til punktet (xkoor,ykoor). MOVETO flytter absolut til det specificerede punkt.

Eksempel

```
0010 PROC stjerne CLOSED
0020   op:=1
0030   REPEAT
0040     DRAW 4,0
0050     DRAW -2,3*op
0060     DRAW -2,-3*op
0070     MOVE 0,2*op
0080     op:=-op
0090   UNTIL op=1
0100 ENDPROC stjerne
0110
0120 OPEN GRAPHICS 1
0130 WINDOW 0,100,0,100
0140 FOR x:=10 TO 90 STEP 10 DO
0150   FOR y:=10 TO 90 STEP 10 DO
0160     MOVETO x,y
0170     EXEC stjerne
0180   NEXT y
0190 NEXT x
```

13.73 NEW

RcComal80 sætning/kommando

Format
NEW [filnavn]

filnavn : navnet på en fil angivet i anførselstegn.

Anvendelse
Sætningen/kommandoen bruges til at slette program, pakker og data i maskinens lager, samt lukke eventuelle åbne filer.

Bemærkning
1. NEW anvendes, når brugeren ønsker at indtaste et nyt program, og derfor vil have slettet det nuværende programindhold, de nuværende pakker og de nuværende data.

2. MARGIN- og ZONE-værdierne sættes til 0 ved NEW.

3. Alle datastrømme, der er blevet åbnet af programmet, lukkes ved NEW. Filer, der er blevet åbnet af pakker, bliver ikke lukket ved NEW.

4. Kommandoen DISCARD udføres, dvs. alle pakker fjernes.

5. Angives filnavn vil filnavnet blive udskrevet i statuslinien, ellers vil den indeholde navnet 01/SAVE.CSV.

13.74 NOT

RcComal80 operator

Format
NOT nudtr

nudtr : vilkårligt numerisk udtryk opfattet som logisk udtryk (0:falsk, <>0: sand)

Operatorprioritet = 6 (se afsnit 4.6.1)

Anvendelse
Operatoren anvendes til at negere det logiske udtryk nudtr. Hvis nudtr er falsk (=0) er NOT nudtr sand (=1), hvis nudtr er sand (<>0) er NOT nudtr falsk (=0).

Eksempel

```
0010 OPEN 1,"klassedata",READ
0020 sumalder:=0; sumhøjde:=0; antal:=0
0030 READ FILE 1: alder,højde
0040 WHILE NOT EOF(1) DO
0050   sumalder:=sumalder+alder; sumhøjde:=sumhøjde+højde
0060   antal:=antal+1
0070   READ FILE 1: alder,højde
0080 ENDWHILE
0090 PRINT "Antal indlæste data      : ";antal
0100 PRINT "Gennemsnitsalder         : ";sumalder/antal
0110 PRINT "Gennemsnitshøjde         : ";sumhøjde/antal
0120 END
```

13.75 OPEN

RcComal80 sætning/kommando

Format

OPEN FILE strømnr, filnavn { READ / WRITE / APPEND / RANDOM recl }

strømnr : numerisk heltal, 1 <= strømnr <= 5
filnavn : navnet på en ydre enhed eller en fil angivet i anførselstegn
recl : numerisk heltal, der angiver det maximale antal tegn i hver post

Anvendelse
Sætningen/kommandoen benyttes til at knytte et logisk nummer (strømnr) til en datastrøm.

Bemærkninger

1. OPEN-sætningen har en parameter, der angiver, hvorledes strømmen skal benyttes:

   READ : læsning af fil (READ FILE- og INPUT FILE-sætninger)
   WRITE : skrivning af fil (WRITE FILE- og PRINT FILE-sætninger).
   APPEND : skrivning af fil, når uddata skal hægtes bag på allerede udskrevet data, (WRITE FILE- og PRINT FILE-sætninger). APPEND kan ikke bruges i forbindelse med kassettebåndsfiler.
   RANDOM : binær udskrivning og indlæsning af direkte tilgang (randomaccess) fil (READ FILE-, WRITE FILE-sætninger). RANDOM kan ikke bruges i forbindelse med kassettebåndsfiler.

2. En diskettefil oprettes automatisk, hvis WRITE angives.

3. Hvis READ, WRITE angives, vil systemet positionere til begyndelsen af filen. Hvis APPEND angives, vil systemet positionere til lige efter de sidst udskrevne data.

4. Når OPEN er udført tilknyttes strømnr til strømmen. Yderligere referencer (READ, WRITE, CLOSE etc.) til strømmen skal herefter foregå via strømnr.

5. recl angiver poststørrelsen i tegn (bytes). Følgende udregningsregler skal anvendes:

   - en numerisk variabel fylder 8 tegn
   - en streng fylder "længden af strengen + 2" tegn

6. Iøvrigt henvises til kapitel 6 : Indlæsning og udskrivning.

Eksempel

```
0010 OPEN FILE 1,"datafil",READ
0020 sum:=0; antal:=0
0030 READ FILE 1: tal
0040 WHILE NOT EOF(1) DO
0050   sum:=sum+tal; antal:=antal+1
0060   READ FILE 1:tal
0070 ENDWHILE
0080 CLOSE FILE 1
0090 PRINT antal;" tal indlæst. Summen af tallene er :";sum
0100 END
```

13.76 OPEN GRAPHICS

RcComal80 grafik sætning/kommando

Format
OPEN GRAPHICS [nudtr].

nudtr : et positivt numerisk udtryk

Anvendelse
Sætningen/kommandoen vælger en ydre grafik-enhed. Udelades nudtr vælges den første grafik-enhed i GSX-systemet.

Bemærkninger

1. nudtr = 1 vælger normalt skærmen

2. Sammenhængen mellem nudtr og det faktiske grafik-enheder fremgår af GSX-modulet ASSIGN.SYS.

3. Der gælder normalt følgende regler for nummerering af enheder:
   1-10 : Skærm
   11-20 : Plottere
   21-30 : Skrivere
   31-40 : Andre grafikenheder

13.77 OR

RcComal80 operator

Format
nudtr1 OR nudtr2

nudtr1 : vilkårligt numerisk udtryk opfattet som logisk udtryk
nudtr2 : vilkårligt numerisk udtryk opfattet som logisk udtryk

Et logisk udtryk opfattes som falsk, hvis det er lig nul; ellers opfattes det som sandt.

Operatorprioritet = 8 (se afsnit 4.6.1)

Anvendelse
Den logiske operator OR er sand (sættes lig 1), hvis enten nudtr1 eller nudtr2 er sande (forskellige fra nul). Hvis både nudtr1 og nudtr2 er falsk (lig nul), bliver resultatet også falsk (nul).

Eksempel

```
0010 DIM logisk$(0:1) OF 5
0020 logisk$(TRUE):="SAND"; logisk$(FALSE):="FALSK"
0030 ZONE 10
0040 PRINT "a","b","a OR b"
0050 FOR i:=1 TO 30 DO PRINT "-";
0060 PRINT
0070 FOR a:=FALSE TO TRUE DO
0080   FOR b:=FALSE TO TRUE DO
0090     PRINT logisk$(a),logisk$(b),logisk$(a OR b)
0100   NEXT b
0110 NEXT a
0120 END
```

```
RUN
a         b         a OR b
------------------------------

SAND      SAND      SAND
SAND      FALSK     SAND
FALSK     SAND      SAND
FALSK     FALSK     FALSK
```

13.78 ORD

RcComal80 funktion

Format
ORD(sudtr)

sudtr : et vilkårligt strengudtryk

Anvendelse
Funktionen returnerer ASCII-værdien for det første tegn i sudtr.

Bemærkninger
1. Værdien, der returneres, er lig den interne værdi. Sammenhængen mellem et tegn og dets interne værdi fremgår af appendix D.

2. ORD-funktionen må indgå i et vilkårligt numerisk udtryk.

13.79 OUT

RcComal80 sætning/kommando

Format
OUT filnavn

filnavn : navnet på en datastrøm i anførselstegn

Anvendelse
Sætningen/kommandoen anvendes til at vælge udskriftsdatastrøm.

Bemærkning
1. Sætningen/kommandoen er identisk med sætningen/kommandoen SELECT OUTPUT (se afsnit 13.109).

13.80 PACKAGE - ENDPACKAGE

RcComal80 struktur

Format
PACKAGE navn
  sætningsliste
ENDPACKAGE navn

navn : navn på pakken
sætningsliste : RcComal80 sætninger

Anvendelse
Konstruktionen benyttes til at definere en RcComal80-pakke.

Bemærkninger
1. navn skal være forskellig fra alle andre navne på pakker, procedurer, funktioner, etiketter, reelle- og strengvariable i programmet.

2. sætningsliste må ikke indeholde PACKAGE-ENDPACKAGE strukturer, dvs. der må ikke defineres pakker indeni en pakke. Men en pakke må gerne bruge andre pakker (ved en USE-sætning).

3. En pakke er lukket, så variable der optræder i pakken er lokale og berører ikke variablerne i resten af programmet (de globale variable). Procedurer og funktioner i pakken kan gøres globale v.hj.a. PUBLIC-sætninger. De lokale variable, der optræder i pakken, bevares, indtil pakken fjernes. Dog vil variable, der opstår ved kald af pakkeprocedurer og pakkefunktioner fra en lukket procedure eller funktion, kun overleve, indtil den lukkede procedure eller funktion afsluttes.

4. Når en pakke gøres brugbar (ved udførelse af en USE-sætning eller USE-kommando) udføres alle sætninger i pakken (pakken og dermed dens variable initialiseres).

5. Når en pakke fjernes (f.eks. ved en NEW- eller DISCARD-kommando) udføres en eventuel EXITPROC.

6. En pakke kan beskyttes v.hj.a. PASSWORD-sætningen. Det betyder, at når pakken gemmes v.hj.a. kommandoen SAVEPACK, er det muligt at bruge pakken, men ikke at LIST'e og ændre pakken (se kommandoerne LOADPACK og PASSWORD).

Eksempel

```
0010 PACKAGE lillegsx
0020   // se beskrivelse af GSX sætning i RcComal80-manual
0030   PUBLIC polyline, polylinetype
0040   DIM kontrol(6),ind(10),pktind(1,2),ud(10),pktud(100,2)
0050
0060   PROC polyline(ant,REF pkti(,))
0070     kontrol(1):= 6
0080     kontrol(2):= ant
0090     kontrol(4):= 0
0100     GSX kontrol,ind,pkti,ud,pktud
0110   ENDPROC polyline
0120
0130   PROC polylinetype(type)
0140     kontrol(1):= 15
0150     kontrol(2):= 0
0160     kontrol(4):= 1
0170     ind(1):= type
0180     GSX kontrol,ind,pktind,ud,pktud
0190   ENDPROC polylinetype
0200 ENDPACKAGE lillegsx
```

13.81 PALETTE

RcComal80 grafik sætning/kommando

Format
PALETTE farvenr ,farveangivelse

farvenr,farveangivelse : positive numeriske udtryk

Anvendelse
Sætningen/kommandoen benyttes til at tilknytte en aktuel farve til de enkelte farvenumre.

Bemærkninger
1. Farveangivelserne er nummereret på følgende måde:
   0 : Sort
   1 : Blå
   2 : Grøn
   3 : Cyan
   4 : Rød
   5 : Magenta
   6 : Gul
   7 : Hvid

2. farvenr kan være 0, 1, 2 eller 3.

3. Sætningen/kommandoen benyttes kun i forbindelse med dataskærmen.

Eksempel

```
0010 // Følgende program vælger sort,cyan,grøn og gul som
0020 // henholdsvis farve 0, 1, 2 og 3
0030 sort:= 0; blå:= 1; grøn:= 2; rød:= 4; hvid:=blå+grøn+rød
0040 cyan:= blå+grøn; magenta:= blå+rød; gul:= grøn+rød
0050 OPEN GRAPHICS 1
0060 PALETTE 0,sort
0070 PALETTE 1,cyan
0080 PALETTE 2,grøn
0090 PALETTE 3,gul
```

13.82 PASSWORD

RcComal80 sætning

Format
PASSWORD løsen

løsen : hemmelig kode i form af en bestemt kombination af tegn

Anvendelse
Sætningen anvendes kun indeni strukturen PACKAGE-ENDPACKAGE. Sætningen benyttes til at beskytte en RcComal80 pakke mod uønsket LIST'ning og editering. En pakke, der er beskyttet med et password kan kun hentes ind med LOADPACK kommandoen, hvis man er i stand til at angive det korrekte løsen.

Bemærkninger
1. Der må kun være een PASSWORD sætning i hver PACKAGE-ENDPACKAGE struktur.

2. Hvis man har beskyttet en pakke med en PASSWORD sætning, skal man kunne huske det korrekte løsen for senere at få mulighed for at liste eller editere pakken.

Eksempel 1

```
0010 PACKAGE min
0020   PASSWORD kukbøh%%%
0030   ...
0200 ENDPACKAGE min
```

13.83 PENCOLOR

RcComal80 grafik sætning/kommando

Format
PENCOLOR nudtr

nudtr : et positivt numerisk udtryk

Anvendelse
Sætningen/kommandoen vælger hvilken "farve", der skal tegnes med (i DRAW, DRAWTO, CIRCLE og TEXT sætningerne). nudtr angiver, hvilket nummer i paletten, der skal anvendes (se PALETTE).

Bemærkning
1. Hvis nudtr er 0 vælges baggrundsfarven.

13.84 PI

RcComal80 konstant

Format
PI

Anvendelse
PI er en konstant med værdien 3.141592653590

13.85 PREFIX

RcComal80 sætning/kommando

Format
PREFIX sudtr

sudtr : et vilkårligt strengudtryk

Anvendelse
Sætningen/kommandoen bruges til at definere en tegnfølge, som automatisk sættes foran de filnavne, der angives i sætninger og kommandoer.

Bemærkninger
1. sudtr kan være en vilkårlig tegnfølge, men normalt angives "1/","2/",... da man derved ikke behøver at angive unitnummeret i filnavne.

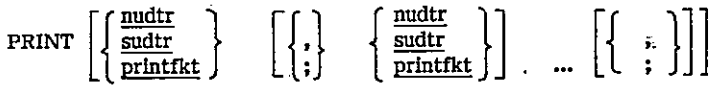
2. Ved opstart udføres en PREFIX-kommando svarende til den unit, man startede RcComal80 op fra. Hvis man f.eks. startede RcComal80 op fra CCP/M unit A vil systemet sætte præfixet til "1/".

3. Ønsker man ikke at benytte præfixet i et filnavn, skal filnavnet blot indledes med "/".

13.86 PRINT

RcComal80 sætning/kommando

Format



printfkt : TAB- eller AT-funktionen
nudtr : et numerisk eller logisk udtryk
sudtr : en strengkonstant eller strengvariabel

Anvendelse
Sætningen/kommandoen anvendes til udskrivning af resultater, tekster m.v.

Bemærkninger

1. Udskrift af nudtr
   Numerisk udtryk (heltal, decimale eller E-notation) udskrives med følgende format: fortegn tal
   Fortegnet er enten minus (-) eller tomt (altså intet mellemrum).

2. Udskrift af sudtr
   Strengudtryk udskrives uden indledende og efterfølgende mellemrum.

3. Udskrift af printfkt
   printfkt udføres. Hvis TAB-argumentet er mindre end nuværende print position ignoreres funktionen (se desuden AT (13.3), TAB (13.119)).

4. Anvendelse af ,
   Udskriftslinien er inddelt i udskriftssøjler. Den første udskriftssøjle starter i position 1, den næste i positionen, der er angivet i en ZONE-sætning. Før hvert PRINT-argument udskrives sammenlignes dets længde med den resterende plads på linien. Er der ikke nok plads, udføres en vognretur, og udskriften fortsætter i første udskriftssøjle.

5. Anvendelse af ;
   Angives semikolon (;) efter nudtr udskrives et mellemrum (et blankt tegn). Angives semikolon efter sudtr eller printfkt, udskrives intet.

13.87 PRINT FILE

RcComal80 sætning/kommando

Format

PRINT FILE strømnr: [{nudtr / sudtr / printfkt} [{, / ;} {nudtr / sudtr / printfkt}] ... [{, / ;}]]

strømnr : et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm, 1 <= strømnr <= 5
printfkt : TAB-funktion
nudtr : et numerisk eller logisk udtryk
sudtr : en strengkonstant eller strengvariabel

Anvendelse
Sætningen/kommandoen benyttes til at skrive data i ASCII format til en datastrøm.

Bemærkninger
1. PRINT FILE-sætningen anvendes til at udskrive data til en ASCII enhed, (console, printer) eller til en diskettefil.

2. Datastrømmen skal være åbnet (OPEN) i WRITE-mode.

3. Udskriftsreglerne er i øvrigt de samme som ved PRINT (se 13.86).

13.88 PRINT FILE USING

RcComal80 sætning/kommando

Format
PRINT FILE strømnr: USING sudtr: nudtr [,nudtr] ... [;]

strømnr : et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm, 1 <= strømnr <= 5
sudtr : streng udtryk, formatstreng
nudtr : numerisk udtryk

Anvendelse
Sætningen/kommandoen anvendes til at udskrive værdier til en datastrøm i et bestemt format.

Bemærkninger
1. Se bemærkningerne til PRINT (se 13.86), PRINT FILE (se 13.87) og PRINT USING (se 13.89).

13.89 PRINT USING

RcComal80 sætning/kommando

Format
PRINT USING sudtr: nudtr [,nudtr] ... [;]

sudtr : streng udtryk, formatstreng
nudtr : numerisk udtryk

Anvendelse
Sætningen/kommandoen anvendes til at udskrive værdier i et bestemt format. Formattegnet er § (paragraftegn).

Bemærkninger
1. sudtr må være et vilkårligt strengudtryk, der udskrives direkte, idet § og . ændres efter følgende regler:
   NB: I det følgende er blanktegn skrevet som : for at tydeliggøre, hvor mange blanke, der er.

   a) Heltalsudskrift
   Hvert § i sudtr afsætter plads til et ciffer (0...9) eller et negativt fortegn (-):

sudtr	nudtr	Udskrift	Kommentarer
§§§§	50	::50	Tal bliver højrestillet indenfor formatet, med indledende blanke tegn.
§§§§	-37	:-37	Fortegnet udskrives umiddelbart før cifrene.
§§§§	1.52	:::2	Decimaltal afrundes.
§§§§	2750	2750	Positivt fortegn optager ikke plads.
§§§§	-4096	****	Hvis nudtr kræver flere positioner end angivet udskrives stjerner.

   b) Decimaltalsudskrift
   Decimalpunktummet (.) udskriver et punktum på en fast plads i udskriftsformatet. Der skal altid angives § på begge sider af punktummet. Hvis nudtr indeholder flere decimaler end angivet, afrundes. Hvis nudtr indeholder færre decimaler end angivet, fyldes op med nuller. For heltalsdelen følges reglerne i pkt. a).

sudtr	nudtr	Udskrift	Kommentarer
§§§.§§	50	:50.00	Decimalerne udskrives altid.
§§§.§§	3.985	::3.99	Decimaler afrundes.
§§§.§§	4096	******	Hvis nudtr har for mange cifre til venstre for decimal-punktummet, erstattes hele formatstrengen af stjerner.

2. Antallet af felter i formatstrengen skal svare til antallet af argumenter i argumentlisten.

Eksempel

```
0010 DIM format$ OF 20
0020 format$:="§§    §."
0040 PRINT "DEC. PI"
0050 FOR decimaler:= 1 TO 11 DO
0060   format$:= format$+"§"
0070   PRINT USING format$:decimaler,PI
0080 NEXT decimaler
0090 END

RUN
DEC. PI
 1   3.1
 2   3.14
 3   3.142
 4   3.1416
 5   3.14159
 6   3.141593
 7   3.1415927
 8   3.14159265
 9   3.141592654
10   3.1415926536
11   3.14159265359
END
AT 0090
```

13.90 PRINTER

RcComal80 kommando

Format
PRINTER [skrivernr]

skrivernr : nummer på en skriverudgang

Anvendelse
Kommandoen benyttes til at aflæse eller ændre skrivernummer i styresystemet fra RcComal80.

Bemærkninger:
1. Uden parameter aflæses det nuværende skrivernummer i styresystemet.
2. Med parameter, ændres skrivernummeret i styresystemet til parameterens værdi.

13.91 PROC - ENDPROC

RcComal80 struktur

Format
PROC navn [(par [, par] ...)] [CLOSED]
  sætningsliste
ENDPROC navn

par:
{ [REF] nvar
  [REF] svar
  REF ntabl ([,] ...)
  REF stabl ([,] ...) }

navn	:	Navn på proceduren (max. 16 tegn)
nvar	:	Vilkårligt numerisk variabel navn
svar	:	Vilkårligt strengvariabelnavn
ntabl	:	Vilkårligt array-navn
stabl	:	Vilkårligt tekstabelnavn
sætningsliste	:	RcComal80 sætninger

Anvendelse
Konstruktionen benyttes til at definere en procedure.

Bemærkninger
1. navn skal være forskellig fra alle andre navne på procedurer, funktioner, etiketter, numeriske variable og strengvariable.

2. En procedure kan gøres lukket (CLOSED) så variable, der optræder i proceduren, er lokale og ikke berører variablerne i resten af programmet (de globale variable). Ønsker man at benytte en variabel, der optræder udenfor proceduren, benyttes GLOBAL-sætningen eller IMPORT-sætningen. De lokale variable, der optræder i proceduren, slettes, når man forlader proceduren.

3. Anføres REF foran variabelnavnet i parameterlisten (de formelle parametre), vil den tilsvarende parameter i EXEC-sætningen (den aktuelle parameter) blive ændret, samtidig med at den formelle parameter ændres nede i proceduren. På denne måde kan man returnere resultater via parametrene.

4. Hvis en pakke, der også bruges udenfor, skal bruges i en CLOSED procedure, er det nødvendigt at importere (v.hj.a. IMPORT eller GLOBAL) pakken og/eller de procedurer og funktioner, pakken stiller til rådighed. Hvis pakken importeres, kan pakkeprocedurerne og pakkefunktionerne kaldes ved punktum-notationen (se EXEC 13.40). Se desuden USE-FROM 13.125.

Eksempel

```
0010 PROC flyt(n,tårn1,tårn2,tårn3)
0020   IF n>1 THEN EXEC flyt(n-1,tårn1,tårn3,tårn2)
0030   EXEC vis(n,tårn1,tårn2)
0040   IF n>1 THEN EXEC flyt(n-1,tårn3,tårn2,tårn1)
0050 ENDPROC flyt
0060 PROC init
0070   MARGIN 0
0080   PRINT CHR$(12)
0090   DIM højde(3),pos(3)
0100   højde(1):=antal; højde(2):=0; højde(3):=0
0110   pos(1):=5; pos(2):=30; pos(3):=55
0120   DIM skive$(antal) OF 22,tom$ OF 22,blok$ of 1
0130   blok$:=CHR$(223)
0140   tom$:="                              "
0150   skive$(1):="             "+blok$+blok$+"             "
0160   FOR i:=2 TO antal DO
0170     skive$(i):=skive$(i-1)(2:11)+blok$
0180     skive$(i)(12:22):=blok$+skive$(i-1)(12:21)
0190   NEXT i
0200   FOR y:=20 TO 21-antal STEP -1 DO
0210     PRINT AT(pos(1),y);skive$(antal+y-20);
0220   NEXT y
0230 ENDPROC init
0240
0250 PROC vis(n,fra,til)
0260   x:=pos(fra)
0270   FOR y:=20-højde(fra) TO 20-antal STEP -1 DO
0280     PRINT AT(x,y);skive$(n);AT(x,y+1);tom$;
0290   NEXT y
0300   retning:=SGN(til-fra)
0310   WHILE x<>pos(til) DO
0320     PRINT AT(x,20-antal);skive$(n);
0330     x:=x+retning
0340   ENDWHILE
0350   FOR y:=20-antal TO 20-højde(til) DO
0360     PRINT AT(x,y);skive$(n);AT(x,y-1);tom$;
0370   NEXT y
0380   højde(fra):=højde(fra)-1; højde(til):=højde(til)+1
0390 ENDPROC vis
0400
0410 PRINT "Tårnene i HANOI"
0420 PRINT
0430 INPUT "Antal skiver (ml. 1 og 10) ":antal
0440 EXEC init
0450 EXEC flyt(antal,1,2,3)
0460 PRINT AT(1,21);
```

13.92 PROC - EXTERNAL

RcComal80 sætning

Format
PROC navn [(par [,par] ...)] EXTERNAL filnavn

par:
```
{  [REF]  nvar          }
{  [REF]  svar          }
{   REF   ntabl ( [,] ... ) }
{   REF   stabl ( [,] ... ) }
```

navn : navn på proceduren
nvar : vilkårligt numerisk variabelnavn
svar : vilkårligt strengvariabel navn
ntabl : vilkårligt array-navn
stabl : vilkårligt teksttabelnavn
filnavn : navn på diskettefil, hvori den externe procedure er gemt med kommandoen SAVE

Anvendelse
Sætningen benyttes til at erklære en extern procedure.

Bemærkninger
1. Den externe procedure skal være SAVE't i filnavn, og første linie i dette program skal være et procedurehoved for en lukket (CLOSED) procedure. Den lukkede procedure må ikke indeholde GLOBAL sætninger eller IMPORT sætninger.

2. Parameterbeskrivelsen i PROC-EXTERNAL skal svare nøjagtig til parameter-beskrivelsen i procedurehovedet i den SAVE'ede procedure.

3. Senere i programmet kan man referere til den externe procedure, som man refererer til en almindelig PROC-ENDPROC. Ved hver EXEC af den externe procedure bliver den indlæst fra disketten.

4. Den externe procedure kan i sig selv indeholde andre procedurer og funktioner, både åbne, lukkede og externe.

5. Hvis en extern procedure kalder sig selv, indlæses den ikke igen.

6. Stopper programudførslen under udførelsen af en extern procedure, er det kun denne man kan se med LIST-kommandoen. SAVE-kommandoen gemmer også kun den externe procedure, hvorimod RUN genstarter hovedprogrammet. NEW sletter både hovedprogram og den externe procedure.

7. Externe procedurer gør det muligt at oprette biblioteker af procedurer, der kan benyttes af flere forskellige programmer.

8. Se endvidere afsnit 8.5 om externe procedurer.

13.93 PROC - HANDLER

RcComal80 struktur

Format
PROC navn HANDLER
  sætninger
ENDPROC navn

navn : navnet på handleren (max. 16 tegn)
sætninger : RcComal80 sætninger

Anvendelse
En PROC-HANDLER er en fejlhåndteringsprocedure, der kaldes automatisk i tilfælde af fejl.

Bemærkninger
1. En PROC-HANDLER kan ikke kaldes med EXEC eller som funktion.

2. En PROC-HANDLER aktiveres med sætningen ENABLE (se afsnit 13.32 ENABLE).

3. Man kan returnere på 4 forskellige måder fra en PROC-HANDLER:

   a. ENDPROC sætningen nås, hvilket bevirker, at programudførelsen standses med en sædvanlig fejludskrift.

   b. Hvis en CONTINUE-sætning udføres, vil programudførelsen fortsætte med sætningen efter den sætning, der forårsagede fejlen. (Se afsnit 13.15 CONTINUE).

   c. Hvis en RETRY-sætning udføres, vil programudførslen fortsætte med den sætning, der forårsagede fejlen (se afsnit 13.102 RETRY).

   d. Hvis en RETURN-sætning udføres, vil den blive udført, som om den optrådte på det niveau, hvor fejlen opstod (bør kun anvendes, hvis fejlen opstod i en FUNC-ENDFUNC eller PROC-ENDPROC). Opstod fejlen i hovedprogrammet, udskrives normal fejlmeddelelse.

4. Nummeret på den fejl, der bevirkede, at PROC-HANDLER blev kaldt, er værdien af SYS(0) eller ERR.

5. Et tryk på ESC-tasten bevirker kald af PROC-HANDLER og stopper således ikke programudførslen.

6. Det er muligt at erklære en PROC-HANDLER inden i en PACKAGE-ENDPACKAGE struktur (RcComal80-pakke). Det er ikke muligt at bruge en handler, der er erklæret udenfor PACKAGE-ENDPACKAGE, indeni pakken.

13.94 PUBLIC

RcComal80 sætning

Format
PUBLIC navn [,navn]...

navn : navn på en procedure eller funktion

Anvendelse
Sætningen anvendes kun i strukturen PACKAGE-ENDPACKAGE. Sætningen benyttes til at angive, hvilke af pakkens procedurer og funktioner, der skal være kendte og kunne benyttes udefra. Efter en USE af pakken, vil det være muligt at benytte de PUBLIC'erede procedurer og funktioner.

13.95 RANDOMIZE

RcComal80 sætning

Format
RANDOMIZE

Anvendelse
Sætningen får "tilfældigtal-generatoren" til at starte på et nyt sted i følgen af tilfældige tal.

Bemærkninger
1. RND funktionen (se afsnit 13.104) genererer normalt den samme følge af tilfældige tal, startende med den samme værdi. Dette er praktisk under indkøringen af programmer. Efter programmet er testet er det derimod praktisk at starte et nyt sted i følgen. For at opnå dette, skal brugeren angive sætningen RANDOMIZE før første kald af RND.

2. RANDOMIZE udregner det nye RND-tal på grundlag af en intern tæller i forbindelse med skærmstyringen.

Eksempel
Både RUN og CON starter et nyt sted i følgen

```
0010 RANDOMIZE
0020 WHILE TRUE DO
0030    FOR i:= 1 TO 3 DO PRINT RND
0040    STOP
0050 ENDWHILE

RUN
0.9726401133793
0.7816163391192
0.3161324487011

STOP
AT 0040

RUN
0.9416661551832
0.6788533738851
0.377864580389
STOP
AT 0040
```

```
CON
0.814314058549
0.7384990342428
0.1156652263127

STOP
AT 0040
```

13.96 READ

RcComal80 sætning

Format

READ { nvar / svar } [, { nvar / svar }] ...

nvar : en numerisk variabel
svar : en streng variabel

Anvendelse
Sætningen læser værdier fra DATA-sætninger og tildeler værdierne til variablene angivet i READ sætningerne.

Bemærkninger
1. READ bruges altid i forbindelse med DATA-sætninger

2. Rækkefølgen af de forskellige typer af parametre i READ-sætningen skal svare til rækkefølgen af værdierne i DATA-sætningerne

3. For hver gang READ læser et dataelement, flyttes datapegepinden til det næste element i listen af DATA-sætninger.

4. Er typen af READ-variablen (numerisk eller streng) ikke den samme som typen af det tilhørende DATA-element udskrives fejlmeddelelsen 0109: TYPE KONFLIKT.

5. Hvis man med READ sætningen forsøger at læse flere data end antallet af elementer i DATA sætningerne udskrives fejlmeddelelsen 0117: IKKE FLERE DATA.

6. Man kan flytte "rundt på" datapegepinden ved hjælp af RESTORE (se afsnit 13.101).

7. EOD-funktionen bliver sand samtidig med, at det sidste element i DATA-sætningerne bliver læst.

13.97 READ FILE

RcComal80 sætning/kommando

Format

READ FILE strømnr [,recnr]: { nvar / svar } [, { nvar / svar }] ...

strømnr : et taludtryk, hvis værdi angiver nummeret på en åben datastrøm, 1 <= strømnr <= 5
recnr : et taludtryk, der angiver et postnummer (kun strømme åbnet i RANDOM mode)
nvar : en numerisk variabel
svar : en strengvariabel

Anvendelse

Sætningen/kommandoen indlæser data i binært format fra en datastrøm, der er åbnet (OPENed) i READ- eller RANDOM-mode.

Bemærkninger

1. Hver variabel i argumentlisten skal have samme type (numeriske eller strenge) som dataene i datastrømmene.

2. Da sætningen indlæser binært format, anvendes sætningen til at læse data, der er skrevet med WRITE FILE-sætningen.

3. recnr kan angives hvis filen er åbnet i RANDOM-mode.

4. EOF-funktionen kan anvendes til at bestemme, hvornår det sidste dataelement er læst.

Eksempel

```
0010 PROC matreadfile(strøm,REF mat(,),dim1,dim2) CLOSED
0020   FOR i:=1 TO dim1 DO
0030     FOR j:=1 TO dim2 DO READ FILE strøm: mat(i,j)
0040   NEXT i
0050 ENDPROC matreadfile
0060
0070 PROC matreadrndfile(strøm,post,REF mat(,),dim1,dim2) CLOSED
0080   READ FILE strøm,post: mat(1,1)
0090   FOR j:=2 TO dim2 DO READ FILE strøm:mat(1,j)
0100   FOR i:=2 TO dim1 DO
0110     FOR j:=1 TO dim2 DO READ FILE strøm: mat(i,j)
0120   NEXT i
0130 ENDPROC matreadrndfile
```

13.98 RENAME

RcComal80 sætning/kommando

Format
RENAME filnavn1, filnavn2

filnavn1 : Navnet på en diskettefil, der skal omdøbes
filnavn2 : Det nye navn til filnavn1

Anvendelse
Sætningen/kommandoen anvendes til at give en diskettefil et nyt navn.

Bemærkning
1. Både filnavn1 og filnavn2 er strengudtryk.

2. filnavn2 må ikke indeholde unitnr.

3. RENAME kan ikke bruges til at omdøbe filer på kassettebånd.

Eksempel
Filen prog.csv omdøbes til glprog.csv

```
RENAME "prog.csv", "glprog.csv"
```

13.99 RENUMBER

RcComal80 kommando

Format

RENUMBER [{ lnr [,] / , lnrspring / lnr, lnrspring }]

lnr : Første linienummer efter omnummereringen
lnrspring : Forskellen mellem linienumrene efter omnummereringen

Anvendelse
Kommandoen anvendes til at omnummerere linienumrene i et program.

Bemærkning

1. Kombinationerne af parametrene til RENUMBER har følgende betydning:

RENUMBER	Programlinierne bliver nummereret så de starter med 0010 og har spring på 0010 mellem linierne.
RENUMBER lnr	Programlinierne bliver nummereret så de starter med lnr og har spring på lnr mellem linierne.
RENUMBER lnr,	Programlinierne bliver nummereret så de starter med lnr og har spring på 0010 mellem linierne.
RENUMBER ,lnrspring	Programlinierne bliver nummereret så de starter med 0010 og har spring på lnrspring mellem linierne.
RENUMBER lnr,lnrspring	Programlinierne bliver nummereret så de starter med lnr og har spring på lnrspring mellem linierne.

Eksempel
Vi forudsætter i det følgende, at programlageret indeholder følgende program:

```
0001 PRINT
0007 PRINT
0012 PRINT
0100 PRINT
```

Da vil

RENUMBER	omnummerere linierne til 0010, 0020, 0030, 0040
RENUMBER 20	omnummerere linierne til 0020, 0040, 0060, 0080
RENUMBER 20,	omnummerere linierne til 0020, 0030, 0040, 0050
RENUMBER ,50	omnummerere linierne til 0010, 0060, 0110, 0160
RENUMBER 100,20	omnummerere linierne til 0100, 0120, 0140, 0160

13.100 REPEAT - UNTIL

RcComal80 struktur

Format
REPEAT
  sætningsliste
UNTIL logisk udtryk

sætningsliste : RcComal80 sætninger
logisk udtryk : et udtryk, der når det udregnes har værdien sand (<>0) eller falsk (=0)

Anvendelse
Konstruktionen benyttes til at gentage udførelsen af en blok sætninger indtil et udsagn er sandt.

Virkemåde
1. sætningsliste udføres
2. logisk udtryk udregnes
3. Hvis værdien af logisk udtryk er falsk, hoppes tilbage til trin 1.
4. Hvis værdien af logisk udtryk er sand, fortsætter programmet med den sætning, der følger efter UNTIL-sætningen.

Bemærkninger
1. Strukturen kan ikke udføres som kommando.
2. En REPEAT-UNTIL løkke må gerne indeholde andre REPEAT-UNTIL løkker.
3. Hvis man hopper ind i en REPEAT-UNTIL løkke udenom REPEAT sætningen fås fejludskriften 0116: ULOVLIGT HOP.
4. NB! sætningsliste udføres altid mindst en gang.

Eksempel 1

```
0010 MARGIN 80
0020 ZONE 4
0030 i:= 1
0040 REPEAT
0050   PRINT i,
0060   i:= i+1
0070 UNTIL i>10
0080 PRINT
0090 PRINT "Efter UNTIL er i=";i
0100 END

RUN
1   2   3   4   5   6   7   8   9   10
Efter UNTIL er i=11
END
AT 0100
```

Eksempel 2
Sætningerne mellem REPEAT og UNTIL udføres altid mindst en gang

```
0010 MARGIN 80
0020 ZONE 4
0030 i:= 20
0040 REPEAT
0050   PRINT "Bliver ALTID udført mindst 1 gang" gang
0060   i:= i-1
0070   PRINT i,
0080 UNTIL i>10
0090 PRINT
0100 PRINT "Efter UNTIL er i=";i
0110 END

RUN
Bliver ALTID udført mindst 1 gang
19
Efter UNTIL er i=19
END
AT 0110
```

Eksempel 3

```
0010 PRINT"Dette program udskriver et blokdiagram på skærmen."
0030 INPUT "Antal søjler (max 12):": antal
0040 DIM blok$ OF 10,grafik$ OF 2, normal$ OF 2,box$ OF 1
0050 grafik$:=" ";normal$:=" ";box$:=CHR$(223)
0060 blok$:= grafik$+box$+box$+box$+box$+normal$
0070 FOR i:=1 TO antal DO INPUT "Værdi (ml. 0 og 20):": søjler(i)
0080
0090 PRINT CHR$(12);AT(1,22);normal$
0100 FOR i:=1 TO 79 DO PRINT "-";// vandret streg
0110 FOR i:=1 TO antal DO PRINT AT(i*6+2,23);søjler(i);
0120 værdi:=1
0130 REPEAT
0140   skrevetsøjle:=FALSE
0150   FOR i:=1 TO antal DO
0160     IF søjler(i)>=værdi THEN
0170       skrevetsøjle:=TRUE
0180       PRINT AT(i*6,22-værdi);blok$
0190     ENDIF
0200   NEXT i
0210   værdi:=værdi+1
0220 UNTIL NOT skrevetsøjle OR værdi>20
0230 PRINT AT(1,22);
0240 END
```

Eksempel 4

```
0010 antbyer:= 8
0020 DIM afstand(antbyer,antbyer)
0030 DIM by$ OF 10,bynavn$ OF 10,svar$ OF 3
0040 PRINT "Programmet finder afstande mellem ";
0050 PRINT "de 8 største byer på Fyn"
0060 PRINT
0070 RESTORE afstandstabel
0080 FOR i:= 1 TO antbyer DO
0090   FOR j:= 1 TO i DO
0100     READ afstand(i,j)
0110     afstand(j,i):= afstand(i,j)
0120   NEXT j
0130 NEXT i
0140
0150 REPEAT
0160   REPEAT
0170     INPUT "Hvorfra ": bynavn$
0180     EXEC findby
0190   UNTIL byfundet
```

```
0200    fraby:= bynr
0210
0220    REPEAT
0230      INPUT "Hvortil ": bynavn$
0240      EXEC findby
0250    UNTIL byfundet
0260    tilby:= bynr
0270
0280    PRINT "Afstanden er ";afstand(fraby,tilby);" km."
0290    PRINT
0300    REPEAT
0310      INPUT "Vil du prøve igen ?": svar$
0320      IF svar$<>"ja" and svar$<>"nej" THEN
0330        PRINT "Svar venligst ja eller nej"
0340      ENDIF
0350    UNTIL svar$="ja" OR svar$="nej"
0360
0370 UNTIL svar$="nej"// hovedløkken gennemløbes til svar er nej
0380 END
0390
0400 PROC findby
0410    RESTORE byer
0420    bynr:= 0
0430    REPEAT
0440      bynr:= bynr+1
0450      READ by$
0460    UNTIL bynr=antbyer OR by$=bynavn$
0470    byfundet:= (bynavn$=by$)// logisk variabel
0480    IF NOT byfundet THEN
0490      PRINT "Den by kender jeg ikke, jeg kender kun:"
0500      RESTORE byer
0510      FOR bynr:= 1 TO antbyer DO
0520        READ by$
0530        PRINT by$
0540      NEXT bynr
0550    ENDIF
0560 ENDPROC findby
0570
0580 byer:
0590 DATA "assens", "bogense","fåborg","kerteminde"
0600 DATA "middelfart","nyborg","odense","svendborg"
0610
0620 afstandstabel:
0630 DATA 0
0640 DATA 42,0
0650 DATA 36,63,0
0660 DATA 59,51,63,0
```

```
0670 DATA 34,30,68,67,0
0680 DATA 67,59,47,19,75,0
0690 DATA 39,30,38,22,45,29,0
0700 DATA 62,72,26,51,86,36,42,0

RUN
Programmet finder afstande mellem de 8 største byer på Fyn

Hvorfra odense
Hvortil assens
Afstanden er 39 km.

Vil du prøve igen ?ja
Hvorfra rynkeby
Den by kender jeg ikke, jeg kender kun:
assens
bogense
fåborg
kerteminde
middelfart
nyborg
odense
svendborg
Hvorfra kerteminde
Hvortil middelfart
Afstanden er 67 km.

Vil du prøve igen ?nej
END
AT 0380
```

13.101 RESTORE

RcComal80 sætning

Format
RESTORE [navn]

navn : navnet på en etikette i programmet

Anvendelse
Sætningen bruges til at flytte data pegepinden til enten den første DATA sætning eller til den første DATA sætning, der står efter den angivne etikette.

Bemærkninger
1. Hvis RESTORE sætningen bruges uden angivelse af navn, flyttes data pegepinden til det første dataelement i den første DATA-sætning.

2. Hvis RESTORE sætningen indeholder angivelse af navn, vil datapegepinden blive flyttet til det første dataelement i den DATA-sætning, der står lige efter etiketten navn.

Eksempel
```
0010 READ held
0020
0030 RESTORE navn
0040 READ antalnavne
0050 DIM navne$(antalnavne) OF 30
0060 FOR i:= 1 TO antalnavne DO READ navne$(i)
0070
0080 RESTORE postnumre
0090 READ antalpostnr
0100 DIM postnr(antalpostnr)
0110 FOR i:= 1 TO antalpostnr DO READ postnr(i)
0120
0130 DATA 7,9,13
0140
0150 postnumre:
0160 DATA 4,2750,2770,2830,2000// 4 er antallet der læses i 0090
0170
0180 navn:
0190 DATA 3, "peter","jens","søren"//3 er antallet der læses i 0040
```

```
0200
0210 FOR i:= 1 TO antalnavne DO PRINT navne$(i)
0220 PRINT
0230 FOR i:= 1 TO antalpostnr DO PRINT postnr(i)
0240 END

RUN
peter
jens
søren

2750
2770
2830
2000
```

13.102 RETRY

RcComal80 sætning

Format
RETRY

Anvendelse
Sætningen anvendes kun i en HANDLER. Sætningen bevirker, at programudførelsen fortsætter med den sætning, der forårsagede kaldet af PROC-HANDLER.

13.103 RETURN

RcComal80 sætning

Format
RETURN [nudtr]

nudtr : et vilkårligt numerisk udtryk

Anvendelse
Sætningen anvendes til at returnere fra en procedure eller en funktion. Hvis RETURN optræder i en FUNC-ENDFUNC vil nudtr blive funktionens værdi.

Bemærkninger
1. PROC-ENDPROC konstruktionen behøver ikke at indeholde en RETURN-sætning, da proceduren altid vil returnere ved ENDPROC.

2. Hvis systemet under programudførelsen af FUNC-ENDFUNC når frem til ENDFUNC udenom en RETURN-sætning fås fejl 0113: FUNKTIONSVÆRDI UDEFINERET.

3. Optræder RETURN i en PROC-HANDLER vil den blive udført, som om den optrådte i den FUNC-ENDFUNC eller PROC-ENDPROC, hvor fejlen opstod.

Eksempel

```
0010 FUNC lige(n)
0020   RETURN (n+1) MOD 2 // 1:sand  0:falsk
0030 ENDFUNC lige
0040
0050 REPEAT
0060   INPUT "Indtast et tal mellem -32768 og 32767 (0 stopper):":tal
0070   PRINT tal;" er ";
0080   IF NOT lige(tal) then print "u";
0090   PRINT "lige"
0100 UNTIL tal=0
0110 END
```

```
RUN
Indtast et tal (0 stopper): 7
7 er ulige
Indtast et tal (0 stopper): -4
-4 er lige
Indtast et tal (0 stopper): 0
0 er lige
END
AT 0110
```

13.104 RND

RcComal80 funktion

Format
RND [(nudtr1,nudtr2)]

nudtr1 : et vilkårligt numerisk udtryk
nudtr2 : et vilkårligt numerisk udtryk

Anvendelse
RND er en numerisk funktion, der genererer et tilfældigt tal.

Bemærkninger
1. De genererede tal er pseudo-tilfældige, idet det næste tal beregnes ud fra det forrige.

2. Angives nudtr1 og nudtr2 vil RND generere et heltal i intervallet fra nudtr1 til nudtr2. Udelades nudtr1 og nudtr2 genereres et decimaltal mellem 0 og 1.

3. RND genererer altid den samme følge af tilfældige tal. Ved NEW, RUN og systemopstart gentages følgen med de samme værdier.

4. RANDOMIZE sætningen (se afsnit 13.95) bevirker, at der startes på et vilkårligt sted i følgen af tilfældige tal.

Eksempel 1

```
0010 WHILE TRUE DO
0020   FOR i:= 1 TO 3 DO PRINT RND
0030   STOP
0040 ENDWHILE

RUN
0.1116003194179          RUN kommandoen repeterer de samme tal
0.8394904533458          påny, CON fortsætter følgen.
0.8203299841557
STOP
AT 0030

RUN
0.1116003194179
0.8394904533458
0.8203299841557
STOP
AT 0030
```

```
CON
0.0370770494076
0.6477562514615
0.8871226414774
STOP
AT 0030
```

Eksempel 2

```
0010 RANDOMIZE
0020 tal:= RND(0,99)  // tilfældigt tal mellem 0 og 99 (incl)
0030 forsøg:= 0// antal forsøg
0040 PRINT "Gæt et tal mellem 0 og 99 (incl)"
0050 REPEAT
0060    INPUT "Indtast dit gæt >": gæt;
0070    forsøg:= forsøg+1
0080    CASE TRUE OF
0090       WHEN gæt<tal
0100       PRINT " For lavt !"
0110       WHEN gæt>tal
0120       PRINT " For højt !"
0130       WHEN gæt=tal
0140       PRINT " TILLYKKE! du fandt tallet på ";forsøg"
0150    ENDCASE
0160 UNTIL gæt=tal
0170 END
```

13.105 RUN

RcComal80 kommando

Format
RUN [filnavn]

filnavn : Navn på en diskettefil, angivet i anførselstegn

Anvendelse
Angives kommandoen uden filnavn startes udførelsen af det program, der ligger i programlageret. Angives et filnavn LOAD'es programmet på denne fil ind, og dette program udføres.
Inden selve udførelsen af programmet påbegyndes, hentes de pakker, der skal bruges. Pakkerne hentes fra disketten (dog kun hvis de ikke allerede er i programlageret). Alle pakker, der bruges fra programmet (se 13.125 USE-FROM), bliver initialiseret.

Bemærkninger

1. Hvis der kun er pakker og ikke noget program i programlageret, sker det intet ved RUN-kommandoen.

2. Datalageret slettes, når RUN-kommandoen bruges.

3. Programudførelsen starter i programlinien med det laveste linienummer.

4. Programnavnet angivet ved filnavn skal være gemt med kommandoen SAVE.

5. Ønskes udskriften standset midlertidigt, trykkes på mellemrumstasten. Ved næste tryk startes der atter.

13.106 SAVE

RcComal80 kommando

Format
SAVE [filnavn]

filnavn : navnet på en diskettefil angivet i anførselstegn

Anvendelse
Kommandoen bruges til at gemme det program, der ligger i programlageret på en diskette eller et kassettebånd. Programmet kan senere indlæses med kommandoen LOAD.

Bemærkninger
1. Programmet gemmes i binært format.

2. Hvis filnavn eksisterer på disketten udskrives fejl 0213: FIL EKSISTERER ALLEREDE.

3. Udelades filnavn vil programmet blive gemt i filen, hvis navn er angivet i statuslinie. Hvis filen allerede eksisterer, slettes den først, hvorefter det nye program SAVEs.

4. Efter SAVE vil statuslinien indeholde navnet på den fil, der er SAVEd.

Eksempel
Gemmer programmet i programlageret på filen MITPROG.CSV på disketten i diskettestation A

```
SAVE "/1/MITPROG"
```

13.107 SAVEPACK

RcComal80 kommando

Format
SAVEPACK navn [ON filnavn]

navn : navn på pakken
filnavn : navn på en fil angivet i anførselstegn

Anvendelse
Kommandoen bruges til at gemme en RcComal80 pakke fra programlageret ud på en diskettefil. Pakken kan senere indlæses med kommandoen LOADPACK.

Bemærkninger
1. Programmet gemmes i binært format.

2. Hvis filnavn eksisterer på disketten, udskrives fejl 0213: FIL EKSISTERER ALLEREDE.

3. Udelades ON filnavn, vil programmet blive gemt i filen, som hedder det samme som navn.

4. Hvis der ikke angives nogen type i filnavnet, tilføjes automatisk .PCK til navnet.

5. Der kan kun være een pakke på hver pakkefil.

Eksempel
Gemmer pakken pakke på filen PAKKE.PCK :

```
SAVEPACK pakke
```

Eksempel
Gemmer pakken test på filen TEST2.PCK på disketten i diskettestation A:

```
SAVEPACK test ON "/1/TEST2"
```

13.108 SCREEN$

RcComal80 funktion

Format
SCREEN$

Anvendelse
SCREEN$ er en system-strengvariabel, der indeholder det aktuelle skærmbillede.

Bemærkning
1. På Partner og PICCOLINE er SCREEN$ en "ægte" funktion, dvs. SCREEN$ ikke kan tildeles en værdi.

13.109 SELECT OUTPUT

RcComal80 sætning/kommando

Format
SELECT OUTPUT filnavn

filnavn : navnet på en datastrøm angivet i anførselstegn

Anvendelse
Sætningen/kommandoen anvendes til at vælge udskriftsdatastrøm.

Bemærkninger
1. Al udskrift, der normalt fremkommer på skærmen (fra PRINT, DIR etc.) vil blive udskrevet på filnavn. Dog udskrives INPUT-tekst og fejlmeddelser stadigvæk på skærmen.

2. Hvis filnavn er navnet på en diskettefil, vil den automatisk blive oprettet.

3. Når RUN afbrydes eller afsluttes, sættes udskriftsdatastrømmen tilbage til datastrømmen "/1/console".

Eksempel

```
0010 DIM svar$ OF 1
0020 REPEAT
0030   INPUT "Ønskes udskrift på printer eller skærm (p/s) ":svar$
0040 UNTIL svar$ IN "pPsS"
0050 IF svar$ IN "pP" THEN SELECT OUTPUT "printer"
0060 ...
1000 SELECT OUTPUT "console" // Kan ikke gøres for mange gange
1010 END
```

13.110 SGN

RcComal80 funktion

Format
SGN (nudtr)

nudtr : et numerisk udtryk

Anvendelse
Funktionen har værdien +1 hvis nudtr er større end nul, værdien 0 hvis nudtr er lig nul og værdien -1 hvis nudtr er mindre end nul.

Eksempel

```
0010 WHILE NOT EOD DO
0020   READ tal
0030   PRINT TAL;" er ";
0040   CASE SGN(tal) OF
0050   WHEN -1
0060     PRINT "negativ"
0070   WHEN 0
0080     PRINT "nul"
0090   WHEN 1
0100     PRINT "positiv"
0110   ENDCASE
0120 ENDWHILE
0130
0140 DATA 6,-6,0

RUN
6 er positiv
-6 er negativ
0 er nul
END
AT 0140
```

13.111 SHOWPACK

RcComal80 kommando

Format
SHOWPACK

Anvendelse
Kommandoen angiver hvilke pakker, der befinder sig i programlageret.

Eksempel
Det følgende viser, at pakkerne copy603 og stack befinder sig i lageret:

```
SHOWPACK
  PACKAGE copy603
  PACKAGE stack
```

13.112 SHOWPROC

RcComal80 kommando

Format
SHOWPROC

Anvendelse
Kommandoen angiver hvilke procedurer og funktioner, der er i programmet, som befinder sig i programlageret.

Eksempel

```
SHOWPROC
0010 PROC initsound
0090 PROC hz(c,f) // kanal(0,1,2), frekvens
0150 PROC atten(c,s) // kanal (0,1,2), styrke (0:høj,...,15:lav).
0190 PROC silence
0260 PROC tone(c,o,t) // kanal, oktav, tone
```

13.113 SIN

RcComal80 funktion

Format
SIN (nudtr)

nudtr : et numerisk udtryk, der angiver et radianantal

Anvendelse
Funktionen udregner sinus til en vinkel udtrykt i radianer.

Eksempel

```
0010 FUNC rad(vinkel)// procedure omformer vinkel -> radianer
0020   return vinkel*PI/180
0030 ENDFUNC rad
0040
0050 // lav en tabel over sinus
0060
0070 MARGIN 80
0080 ZONE 20
0090 PRINT "vinkel","radianer","sinus"
0100 FOR v:= 0 TO 90 STEP 5 DO PRINT v,rad(v),SIN(rad(v))
0110 END
```

```
RUN
vinkel              radianer                sinus
0                   0                       0
5                   8.726646259971E-002     8.715574274761E-002
10                  0.1745329251994         0.1736481776669
15                  0.2617993877991         0.2588190451025
20                  0.3490658503986         0.3420201433254
25                  0.4363323129984         0.4226182617404
30                  0.5235987755982         0.5
35                  0.6108652381977         0.5735764363506
40                  0.6981317007975         0.6427876096864
45                  0.7853981633973         0.7071067811865
50                  0.8726646259971         0.7660444431189
55                  0.9599310885966         0.8191520442895
60                  1.047197551196          0.8660254037844
65                  1.134464013796          0.9063077870373
```

70	1.221730476395	0.9396926207864
75	1.308996938995	0.9659258262897
80	1.396263401595	0.9848077530132
85	1.483529864194	0.9961946980927
90	1.570796326794	1

```
END
AT 0110
```

13.114 SIZE

RcComal80 kommando

Format
SIZE

Anvendelse
Angiver fordelingen af frit lager og lager benyttet til henholdsvis program og data.

Bemærkning
1. Dataarealet vil være tomt, hvis programmet i maskinens lager ikke har været udført. Efter en udførelse af programmet er dataarealets størrelse et mål for, hvor mange variable programmet benytter.

Eksempel

```
SIZE
program         data            free
00310           00000           23131
```

13.115 SQR

RcComal80 funktion

Format
SQR (nudtr)

nudtr : et ikke-negativt udtryk

Anvendelse
Funktionen uddrager kvadratroden af nudtr.

Bemærkning
1. Forsøges kvadratroden uddraget af et negativt tal, fås fejludskriften 0103: KVADRATROD AF NEGATIVT TAL.

Eksempel

```
0010 ZONE 16
0020 PRINT SQR(25),SQR(25.1)
0030 END
0040


RUN
5                    5.00999001995
END
AT 0030
```

13.116 STOP

RcComal80 sætning

Format
STOP

Anvendelse
Sætningen standser udførelsen af et program.

Bemærkninger
1. Sætningen kan ikke udføres som kommando.

2. STOP-sætninger må placeres et vilkårligt sted i programmet. Når en af disse mødes udskrives:
   STOP
   AT XXXX
   hvor XXXX er linienummeret på STOP-sætningen.

3. Når STOP-sætningen er udført kan brugeren fortsætte programudførslen med CON-kommandoen.

13.117 STR$

RcComal80 funktion

Format
STR$ (nudtr)

nudtr : vilkårligt numerisk udtryk

Anvendelse
Funktionen konverterer et numerisk udtryk til en streng.

Bemærkninger
1. Funktionen omformer f.eks. tallet 1047 til strengen "1047".

2. Funktionen udfører det omvendte af VAL (se afsnit 13.127).

Eksempel 1

```
0010 FUNC cprnummerok(cprnr) CLOSED
0020   DIM tal$ OF 10
0030   tal$:=STR$(cprnr)
0040   WHILE len(tal$)<10 DO tal$:="0"+tal$
0050   sum:=0
0060   for i:=1 TO 7 DO sum:=sum+VAL(tal$(11-i:11-i))*i
0070   for i:=2 TO 4 DO sum:=sum+VAL(tal$(5-i:5-i))*i
0080   RETURN (sum MOD 11 = 0)
0090 ENDFUNC
0100 INPUT "Indtast et cprnummer :": nr
0110 IF cprnummerok(nr) THEN
0120   PRINT "CPR nummer OK"
0130 ELSE
0140   PRINT "CPR nummer forkert"
0150 ENDIF
0160 END
```

Eksempel 2

```
0010 FUNC e$(tal) CLOSED
0020   // Proceduren omformer tal til en streng, der har
0030   // formatet <fortegn>x.xxxxxxxxxxxxE<fortegn>nnn
0040   DIM fortegn$ OF 1,mantisse$ OF 14
0050   DIM ex$ OF 4,t$ OF 19
0060   IF tal>=0 THEN
0070     fortegn$:="+"
0080   ELSE
```

```
0090      fortegn$:="-"
0100    ENDIF
0110    t$:=STR$(ABS(tal))
0120    eposi:=("E" IN t$)
0130    IF eposi<>0 THEN
0140      ex$:=t$(eposi+1:LEN(t$))
0150      mantisse$:=t$(1:eposi-1)
0160      IF LEN(mantisse$)=1 THEN mantisse$:=mantisse$+"."
0170    ELSE
0180      pktpos:=("." IN t$)
0190      IF pktpos<>0 THEN
0200        t$:=t$(1:pktpos-1)+t$(pktpos+1:LEN(t$)) // . fjernes
0210      ELSE
0220        pktpos:=LEN(t$)+1
0230      ENDIF
0240      potens:=0
0250      IF t$<>"0" THEN
0260        potens:=pktpos-2; i:=1
0270        WHILE t$(i:i)="0" DO i:=i+1; potens:=potens-1
0280        t$:=t$(i:LEN(t$))
0290      ENDIF
0300      IF potens>=0 THEN
0310        ex$:="+0"+STR$(potens DIV 10)+STR$(potens MOD 10)
0320      ELSE
0330        ex$:="-0"+STR$(-potens DIV 10)+STR$(-potens MOD 10)
0340      ENDIF
0350      mantisse$:=t$(1:1)+"."
0360      IF LEN(t$)>1 THEN mantisse$:=mantisse$+t$(2:LEN(t$))
0370    ENDIF
0380    mantisse$:=mantisse$+"000000000000"
0390    RETURN fortegn$+mantisse$+"E"+ex$
0400  ENDFUNC e$
0410
0420  DIM talstr$ OF 20
0430  ZONE 30
0440  WHILE NOT EOD DO
0450    READ tal
0460    PRINT tal,e$(tal)
0470  ENDWHILE
0480  DATA -1,3.5E+070,-4E-032,0.000000001
0490  DATA 999999.99,0.1,0.01,0.001
0500  DATA 1,10,100,1000,10000,100000,1000000
```

```
RUN
-1                  -1.000000000000E+000
3.5E+070            +3.500000000000E+070
-4E-032             -4.000000000000E-032
0.000000001         +1.000000000000E-010
999999.99           +9.999999900000E+005
0.1                 +1.000000000000E-001
0.01                +1.000000000000E-002
0.001               +1.000000000000E-003
1                   +1.000000000000E+000
10                  +1.000000000000E+001
100                 +1.000000000000E+002
1000                +1.000000000000E+003
10000               +1.000000000000E+004
100000              +1.000000000000E+005
1000000             +1.000000000000E+006
```

13.118 SYS

RcComal80 funktion

Format
SYS(nudtr)

nudtr : et numerisk udtryk

Anvendelse
Funktionen returnerer systeminformation afhængig af værdien af nudtr:

Funktion	Information
SYS(0)	Fejlnummeret for sidste RUN-time fejl (har kun en værdi forskellig fra nul i en PROC-HANDLER).
SYS(1)	Strømnummeret for sidste strøm man har benyttet (har kun en værdi forskellig fra nul i en PROC-HANDLER).
SYS(2)	Linienummeret på den linie, hvori der opstod fejl (har kun en værdi forskellig fra nul i en PROC-HANDLER).
SYS(3)	Antal 20 msek siden opstart.
SYS(4)	Antal frie bytes i lageret.
SYS(5)	Den aktuelle ZONE-værdi.
SYS(6)	Den aktuelle MARGIN-værdi.
SYS(7)	Maskintype. Partner=3, PICCOLINE=4.

13.119 TAB

RcComal80 funktion

Format
TAB (nudtr)

nudtr : et numerisk udtryk i området 1 <= nudtr <= sidebredden angivet med MARGIN-sætningen

Anvendelse
Funktionen anvendes kun i PRINT-sætninger og bevirker tabulering til den kolonne, der er angivet ved nudtr.

Bemærkninger
1. Udskrivningskolonnerne nummereres fra 1.

2. Hvis en TAB-værdi er større end den aktuelle MARGIN-værdi, udskrives nudtr-1 mellemrum.

Eksempel

```
0010 MARGIN 80
0020 ZONE 0
0030 FOR i:=1 TO 60 DO PRINT USING "S":i MOD 10;
0040 PRINT
0050 WHILE NOT EOD DO
0060   READ pos
0070   PRINT TAB(pos);"* ";pos
0080 ENDWHILE
0090 DATA 1,7,40,38

RUN
12345678901234567890123456789012345678901234567789012345678 90
* 1
      * 7
                                          * 40
                                       * 38
```

13.120 TAN

RcComal80 funktion

Format
TAN (nudtr)

nudtr : et numerisk udtryk, der angiver et radianantal

Anvendelse
Funktionen udregner tangens til en vinkel udtrykt i radianer.

13.121 TEXT

RcComal80 grafik sætning/kommando

Format
TEXT sudtr

sudtr : vilkårligt strengudtryk

Anvendelse
Sætningen/kommandoen skriver teksten sudtr på den aktuelle grafikenhed.
Teksten skrives ud fra det løbende punkt.

Bemærkninger
1. Det løbende punkt ændres ikke med sætningen/kommandoen.

13.122 Tildeling

RcComal80 sætning/kommando

Format

```
{ nvar:= nudtr }                 [ ; { nvar:= nudtr } ]
{ svar:= sudtr }                 [   { svar:= sudtr } ]   ...
{ nvar:+ nudtr }                 [   { nvar:+ nudtr } ]
{ nvar:- nudtr }                 [   { nvar:- nudtr } ]
```

nvar : en numerisk variabel
nudtr : et numerisk udtryk
svar : en strengvariabel
sudtr : et strengudtryk

Anvendelse
Sætningen/kommandoen anvendes, når en variabel skal tildeles en værdi.

Bemærkninger
1. nvar, svar skal være indicerede, hvis de er erklæret således.

2. Angående formatet for numeriske udtryk og strengudtryk henvises til kapitel 4: Tal og tekst.

3. nvar:+ nudtr giver samme tildeling som nvar:= nvar + nudtr.

4. nvar:- nudtr giver samme tildeling som nvar:= nvar - nudtr.

Eksempel

```
0010 DIM navn$ OF 20
0020 navn$:="COMAL80"
0030 ejok:=(navn$<"COMAL")
0040 pi:=ATN(1)*4; slut:=FALSE
```

13.123 TIME$

RcComal80 funktion

Format
TIME$

Anvendelse
Funktionen returnerer det aktuelle tidspunkt.

Bemærkning
1. Tidspunktet angives på formatet TT:MM:SS, hvor TT angiver timer (f.eks. 17), MM angiver minutter (f.eks. 30) og SS angiver sekunder (f.eks. 50).

Eksempel

```
PRINT TIME$
13:59:12
```

13.124 TRUE

RcComal80 konstant

Format
TRUE

Anvendelse
TRUE er en konstant med værdien 1.

Eksempel

```
0010 // Eksempel på en uendelig løkke
0020 WHILE TRUE DO
0030 ...
0100 ENDWHILE
```

13.125 USE - FROM

RcComal80 sætning/kommando

Format
USE navn [FROM filnavn] [, navn [FROM filnavn]] ...

navn : navn på en pakke
filnavn : navn på en diskettefil, hvor pakken er gemt. Navnet angives i anførselstegn.

Anvendelse
Sætningen/kommandoen benyttes til at specificere, hvilke pakker programmet benytter. Hvis 'FROM filnavn' udelades, benyttes navn som filnavn.

Bemærkning
1. Hvis der ikke angives nogen type i filnavnet, tilføjes automatisk .PCK til navnet.

2. Parameteren navn i USE-sætningen eller USE-kommandoen bestemmer, hvad en pakke hedder i dette program. Dvs. det er muligt at have flere udgaver af den samme pakke, hvis den USEs med forskellige navne:
```
0010 USE st1 FROM "stack"
0020 USE st2 FROM "stack"
```

3. I et program skal navn i USE-sætninger være entydige, dvs. det er ikke muligt at to pakker kaldes det samme i et program.

4. Afgives USE som kommando, hentes pakken ind i programlageret (såfremt den ikke findes der i forvejen), og pakken initialiseres. Ved initialiseringen bliver procedurer og funktioner i pakken kendte, så de kan udføres på kommando-niveau. Navne på variable, procedurer etc., som var kendte før, er ikke længere kendte.

5. Gives USE som sætning, sker der følgende ved RUN-kommandoen: Pakken hentes ind i programlageret (såfremt den ikke findes der i forvejen) og initialiseres, inden den egentlige programudførelse starter. Ved initialiseringen bliver procedurer og funktioner i pakken kendte, så de kan udføres som sætninger i programmet og på kommando-niveau.
Under programudførelsen kan pakkens procedurer og funktioner benyttes på samme måde, som hvis de var erklærede som lukkede procedurer/funktioner på det sted, hvor USE-sætningen er.
Pakker, procedurer, funktioner etc., der var kendte før RUN-kommandoen, er ikke længere kendte.

6. Hvis der allerede er en pakke, som hedder navn, i programlageret, når der afgives en USE-sætning eller -kommando, bruges denne, og der hentes ikke en version ind fra disketten. Hvis man ønsker, at pakken skal hentes fra disketten, er det nødvendigt af fjerne alle pakker (ved kommandoen DISCARD).

7. Hvis en pakke, der også bruges udenfor, skal bruges i en CLOSED procedure eller funktion, er det nødvendigt at importere (v.hj.a. IMPORT eller GLOBAL) pakken og/eller de procedurer og funktioner, pakken stiller til rådighed. Hvis pakken importeres, kan pakkeprocedurerne og pakkefunktionerne kaldes ved punktum-notationen (se EXEC 13.40).

8. Hvis en pakke kun skal bruges i en CLOSED procedure eller funktion, kan pakken bruges ved at lave en USE-sætning i den lukkede procedure eller funktion.

Eksempel

```
USE program
kommando("graphics")
```

Dette bevirker, at grafik-styresystemet installeres inde fra RcComal80, da program er en pakke med en procedure, der hedder kommando. Proceduren kommando har en strengparameter, som skal være navnet på et CMD-program.

13.126 USER

RcComal80 kommando

Format
USER [usernr]

usernr : brugernr, der er et tal, som opfylder 0 <= usernr <= 15

Anvendelse
Kommandoen benyttes til at aflæse eller ændre brugernummer i styresystemet fra RcComal80. Uden parameter aflæses det nuværende brugernummer i styresystemet. Med parameter, ændres brugernummeret i styresystemet til parameterens værdi.

Bemærkninger
1. Eventuelle åbne strømme bliver lukket ved brug af kommandoen USER usernr.

13.127 VAL

RcComal80 funktion

Format
VAL(sudtr)

sudtr : et vilkårligt strengudtryk

Anvendelse
Funktionen udregner værdien af et tal, der optræder i en streng.

Bemærkninger
1. Funktionen gør det muligt at bruge strengvariable i INPUT-sætningen, og derefter omforme strengen til en numerisk værdi.

2. VAL omformer alle former for talrepræsentation (incl. decimal- og exponential-notation).

3. VAL udfører det modsatte af STR$.

Eksempel
```
PRINT VAL("12.45")+20
32.45
```

13.128 WHILE

RcComal80 sætning

Format
WHILE logisk udtryk DO simpel sætning

logisk udtryk : et udtryk, der, når det udregnes, har værdien sand (<>0) eller falsk (=0)

simpel sætning : en simpel RcComal80 sætning, dvs. CIRCLE, CLEAR, CLOSE, CLOSE GRAPHICS, CONTINUE, CREATE, DELETE, DRAW, DRAWTO, END, EXEC, GOTO, GPARM, GSX, INPUT, INPUT FILE, LOCATE, MOVE, MOVETO, OPEN FILE, OPEN GRAPHICS, PALETTE, PENCOLOR, PRINT, PRINT FILE, PRINT USING, READ, READ FILE, RESTORE, RETRY, RETURN, SELECT OUTPUT, STOP, TEXT, tildeling, WINDOW, WRITE FILE

Anvendelse
Benyttes til at gentage udførelsen af en sætning, så længe et udtryk er sandt.

Virkemåde
1. Hvis værdien af logisk udtryk er falsk (=0) hoppes til næste sætning efter WHILE-sætningen.

2. simpel sætning udføres.

3. Hop til trin 1.

Bemærkninger
1. Sætningen kan ikke udføres som kommando.

2. Da simpel sætning kun udføres, så længe logisk udtryk er sand (<>0), er det ikke sikkert, at simpel sætning bliver udført overhovedet.

3. Ønskes mere end en simpel sætning udført, skal konstruktionen WHILE-ENDWHILE (se afsnit 13.129) benyttes.

13.129 WHILE - ENDWHILE

RcComal80 struktur

Format
WHILE logisk udtryk DO
  sætningsliste
ENDWHILE

logisk udtryk : et udtryk, der når det udregnes, har værdien sand (<>0) eller falsk (=0)
sætningsliste : RcComal80 sætninger

Anvendelse
Konstruktionen benyttes til at gentage udførelsen af sætningsliste sålænge logisk udtryk er sand.

Virkemåde
1. Hvis værdien af logisk udtryk er falsk (=0) hoppes til sætningen efter ENDWHILE sætningen.

2. sætningsliste udføres.

3. Hop til trin 1.

Bemærkninger
1. Sætningen kan ikke udføres som kommando.

2. WHILE-ENDWHILE må gerne indeholde andre WHILE-sætninger og WHILE-ENDWHILE strukturer.

3. Hvis ENDWHILE mangler fås fejl 0096: FEJL I PROGRAMSTRUKTUR.

4. Hvis man hopper ind i en WHILE-ENDWHILE løkke udenom WHILE-sætningen fås fejl 0116: ULOVLIGT HOP.

5. Da sætningsliste kun udføres, sålænge logisk udtryk er sand (<>0), er det ikke sikkert, at sætningsliste udføres overhovedet.

Eksempel

```
0010 INPUT "Indtast et beløb (kroner.ører) :": beløb
0020 beløb:= INT(beløb*20+0.5)/20
0030 PRINT "Afrundet til nærmeste hele femøre :";beløb
0040 DIM navn$ OF 50
```

```
0050 WHILE NOT EOD DO
0060   READ kroner,navn$
0070   antal:= 0
0080   WHILE beløb>=kroner DO
0090     beløb:= beløb-kroner
0100     antal:= antal+1
0110   ENDWHILE
0120   PRINT antal;navn$
0130 ENDWHILE
0140 DATA 1000,"Tusindkronesedler"
0150 DATA 500,"Femhundredkronesedler"
0160 DATA 100,"Hundredkronesedler"
0170 DATA 50,"Halvtredskronesedler"
0180 DATA 20,"Tyvekronesedler"
0190 DATA 10,"Tikroner"
0200 DATA 5,"Femkroner"
0210 DATA 1,"Enkroner"
0220 DATA 0.25,"Femogtyveøre"
0230 DATA 0.1,"Tiøre"
0240 DATA 0.05,"Femøre"
```

13.130 WINDOW

RcComal80 grafik sætning/kommando

Format
WINDOW xvenst , xhøjre , yned , yop

xvenst : Et numerisk udtryk
xhøjre : Et numerisk udtryk
yned : Et numerisk udtryk
yop : Et numerisk udtryk

Anvendelse
Sætningen/kommandoen benyttes til at erklære det aktuelle grafik-vindue.

Bemærkninger
1. WINDOW kan ændres dynamisk. Dette har dog ingen indflydelse på symboler og tekst, der er skrevet.

2. Øverste venstre hjørne har koordinatsættet (xvenst,yop). Nederste højre hjørne har koordinatsættet (xhøjre,yned).

Eksempel

```
0010 FUNC f(x)
0020   RETURN 1/x
0030 ENDFUNC f
0040
0050 PROC h HANDLER
0060   IF (ERR>=102 AND ERR<=104) OR ERR=113 THEN
0070     IF ymax<0 THEN RETURN ymax/2
0080     RETURN ymax*2
0090   ENDIF
0100 ENDPROC h
0110
0120 ENABLE h
0130 INPUT "xmin, xmax, xstep =": xmin,xmax,xstep
0140 INPUT "ymin, ymax =": ymin,ymax
0150 OPEN GRAPHICS 1
0160 WINDOW xmin,xmax,ymin,ymax
0170 // tegn akser
0180 IF SGN(xmin)<>SGN(xmax) THEN
0190   MOVETO 0,ymin
0200   DRAWTO 0,ymax
0210 ENDIF
```

```
0220 IF SGN(ymin)<>SGN(ymax) THEN
0230   MOVETO xmin,0
0240   DRAWTO xmax,0
0250 ENDIF
0260 tegn:=FALSE
0270 FOR x:=xmin TO xmax STEP xstep DO
0280   fx:=f(x)
0290   IF fx>=ymin AND fx<=ymax THEN
0300     IF tegn THEN
0310       DRAWTO x,fx
0320     ELSE
0330       MOVETO x,fx
0340       tegn:=TRUE
0350     ENDIF
0360   ELSE
0370     tegn:=FALSE
0380   ENDIF
0390 NEXT x
```

13.131 WRITE FILE

RcComal80 sætning/kommando

Format

WRITE FILE strømnr [,recnr]: { nudtr / sudtr } [, { nudtr / sudtr }] ...

strømnr : et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm, 1 <= strømnr <= 5
recnr : et numerisk udtryk, hvis værdi angiver nummeret på en post (angives kun ved RANDOM-filer)
nudtr : et numerisk udtryk
sudtr : et strengudtryk

Anvendelse

Sætningen/kommandoen udskriver data i binært format i en datastrøm. Filen skal være åbnet (OPEN) i WRITE-, APPEND- eller RANDOM-mode.

Bemærkninger

1. Data kan læses igen med READ FILE-sætningen.

2. I binært format fylder en numerisk variabel 8 tegn (bytes) og en streng 2 tegn plus den aktuelle længde.

3. Hvis man forsøger at udskrive en record (i en RANDOM-fil), der er længere end den record-længde, der er specificeret i OPEN-sætningen, fås fejl 0221: END-OF-RECORD.

Eksempel

```
0010 PROC matwritefile(strøm,REF mat(,),dim1,dim2) CLOSED
0020   FOR i:=1 TO dim1 DO
0030     FOR j:=1 TO dim2 DO WRITE FILE strøm: mat(i,j)
0040   NEXT i
0050 ENDPROC matwritefile
0060
0070 PROC matwriterndfile(strøm,post,REF mat(,),dim1,dim2) CLOSED
0080   WRITE FILE strøm,post: mat(1,1)
0090   FOR j:=2 TO dim2 DO WRITE FILE strøm: mat(1,j)
0100   FOR i:=2 TO dim1 DO
0110     FOR j:=1 TO dim2 DO WRITE FILE strøm: mat(i,j)
0120   NEXT i
0130 ENDPROC matwriterndfile
```

13.132 ZONE

RcComal80 sætning/kommando

Format
ZONE nudtr

nudtr : et numerisk udtryk med værdi i intervallet 0 <= nudtr <= sidebredden angivet med MARGIN sætningen

Anvendelse
Sætningen/kommandoen benyttes til at angive kolonnebredden mellem elementer i en PRINT sætning adskilt med komma (,).

Bemærkninger
1. Standardværdien ved opstart er ZONE 0. Dette bevirker at PRINT-elementerne udskrives uden mellemrum.

2. Da den maximale ZONE-værdi er afhængig af MARGIN værdien, er det praktisk først at angive MARGIN og derefter ZONE.

3. Hvis MARGIN-værdien er nul, kan ZONE-værdien være vilkårlig.

Eksempel

```
0010 MARGIN 30
0020 ZONE 10
0030 FOR i:= 1 TO 10 DO PRINT i,
0040 PRINT
0050 ZONE 5
0060 FOR i:= 1 TO 10 DO PRINT i,
0070 PRINT
0080 END

RUN
1         2         3
4         5         6
7         8         9
10
1    2    3    4    5    6
7    8    9    10
```

A. Referencer

(1) SW1400 eller SW1400D eller SW1401D eller SW1458 eller SW1458D
PICCOLINE
Brugervejledning, Installation og Vedligeholdelse

(2) SW1500 eller SW1500D
Partner
Brugervejledning, Installation og Vedligeholdelse

(3) SW1699D
CCP/M-86 Manual Set (Engelsk)

(4) SW1698D
GSX-86 Manual Set (Engelsk)

(5) SW1498D
PICCOLINE
Programmers Guide (Engelsk)

(6) SW1695
Partner
Programmers Kit for C-DOS (Engelsk)
(Indeholder bl.a. Programmers Guide)

(7) SW1402 eller SW1402D
PICCOLINE
PolyPascal

(8) SW1502 eller SW1502D
Partner
PolyPascal

(9) SW1613
DR Assembler plus Tools

(10) MF916
PICCOLINE
ADAM brugervejledning

B. Fejlmeddelelser

De fejl, der kan opstå i RcComal80 systemet, kan deles i 3 grupper:

1. **Fejl fundet under programindtastning**

   Hvis en fejl konstateres under programindtastning, udskrives en fejlmeddelelse i skærmens øverste højre hjørne, og cursoren placeres umiddelbart efter det sted, hvor fejlen blev fundet. En liste over disse fejlmeddelelser findes i afsnit B.1.

2. **Fejl fundet under program-, eller kommando-udførelse**

   Fejlmeddelelserne til disse udskrives på følgende form:

   ```
   AT nnnn
   ERROR: eeee
   ```

   nnnn : linienummeret på sætningen hvor fejlen blev fundet.
   eeee : et tal under 200, hvis betydning er beskrevet i afsnit B.2.

   Hvis fejlen er opstået i en pakke, tilføjes følgende til den første linie:

   ```
   I PAKKEN pakkenavn
   ```

   hvor pakkenavn er navnet på pakken.

   Opstår fejlen under kommandoudførsel, udskrives der intet linienummer.

3. **Fejl fundet under disketteoperationer**

   Fejlmeddelelserne til disse har samme format som fejlmeddelelserne under punkt 2, men fejlnumrene er større end 200 og er beskrevet i afsnit B.3.

B.1 Fejl fundet under programindtastning

linie for lang
Fejlmeddelelsen betyder, at den linie man har indtastet ikke kan repræsenteres internt, hvorfor den må brydes op i flere linier.

ulovligt tegn
Systemet har fundet et tegn, det ikke accepterer andre steder end i strengkonstanter og kommentarer, f.eks. %. Cursoren er placeret efter det ulovlige tegn.

ulovligt linienummer
Et linienummer må ikke være mindre end 1 eller større end 9999.

ulovligt nummer
Printer- eller usernummer ligger udenfor det tilladte område.

stakoverløb
Der er så lidt ledigt lager tilbage, at systemet ikke kan omforme sætningen til det interne format.

variabel forventet
Systemet forventede en variabel på det sted, hvor cursoren er placeret, f.eks

```
0010 READ "niels"
```

')' forventet
Systemet forventede en parentesafslutning ')' på det sted, hvor cursoren er placeret, f.eks.

```
0010 a(1,1,1 := 10
```

fejl i indicering
Der refereres ikke korrekt til et element i et array, en streng eller en teksttabel.

operand forventet
Systemet forventede en operand på det sted, hvor cursoren er placeret.

```
0010 a:=10*
```

ulovlig type
Man har f.eks. angivet en streng, hvor en numerisk udtryk var forventet (eller omvendt).

```
0010 navn$ := 7
```

syntaks fejl
RcComal80 har genkendt den første del af linien som en sætning, men de sidste tegn giver ingen mening. Der kan f.eks. mangle et tegn, eller et nøgleord bruges som variabelnavn.

```
0010 a:=5  b:=6
```

fejl i konstant
Opskrivningen af en konstant følger ikke reglerne.

```
0010 a:=3.1E
```

navn for langt
Et navn må maximalt være på 16 tegn.

" forventet
Systemet mangler et anførselstegn (") til afslutningen af en strengkonstant.

```
0010 navn$ := "COMAL80
```

'(' forventet
Systemet forventede at møde en begyndelsesparentes på det sted, hvor cursoren er placeret.

```
0010 DIM ab(5)
```

',' forventet
Systemet forventede at møde et komma på det sted, hvor cursoren er placeret.

```
0010 OPEN FILE 1,"DATAFIL" READ
```

tildeling forventet
Systemet forventede at finde et tildelingstegn på det sted, hvor cursoren er placeret.

':' forventet
Systemet forventede at finde et kolon på det sted, hvor cursoren er placeret.

0010 INPUT "Skriv dit navn " navn$

'OF' forventet
OF skal skrives som afslutning af en CASE-sætning.

0010 CASE a

'TO' forventet
Systemet kan ikke finde TO i en FOR-sætning.

0010 FOR i:=1 STEP 2 DO

'DO' forventet
DO skal skrives i en WHILE- eller i en FOR-sætning.

0010 WHILE i>10

'THEN' forventet
THEN skal skrives i en IF-sætning.

0010 IF svar$="SLUT" PRINT "Farvel"

'REF' forventet
Taltabeller og teksttabeller skal specificeres som REF-parametre.

0010 PROC P(a(,))

'ON' forventet
Systemet forventede ON og en filangivelse i forbindelse med brug af pakker.

SAVEPACK pakke "pakke3.pck"

'FROM' forventet
Systemet forventede FROM og en filangivelse i forbindelse med brug af pakker.

LOADPACK pakke "pakke3.pck"

navn forventet
Systemet forventede et navn (f.eks. på en label) der, hvor cursoren er placeret.

```
0010 GOTO 1000
```

ulovlig kommando
Sætningen kan ikke udføres som kommando (se afsnit 9.2).

```
WHILE i<=10 DO PRINT i
```

for mange variable
Systemet har ikke plads til flere variabelnavne.

rekursion
Den indtastede sætning er så kompliceret, at RcComal80 systemet ikke kan håndtere den. Prøv at dele den i mindre sætninger.

ikke mere plads
RcComal80 systemets programlager er opbrugt. Prøv at fjerne pakker. Eller prøv at definere dele af programmet som externe procedurer eller funktioner.

ikke implementeret
Faciliteten er ikke implementeret i systemet.

B.2 Kommando- eller udførelsesfejl

0095 : CON IKKE TILLADT
Man forsøger at bruge CON i forbindelse med et program, man har rettet i.

0096 : FEJL I PROGRAMSTRUKTUR
Systemet kan ikke finde den tilhørende NEXT til FOR, ENDIF til IF, PROC til ENDPROC osv.

0097 : IKKE SAVE FIL
LOAD-kommandoen er anvendt på en fil, der ikke indeholder et SAVE'd program.

0098 : IKKE IMPLEMENTERET
Faciliteten er endnu ikke implementeret i systemet.

0099 : SYSTEMFEJL
I dette tilfælde bør man udfylde en SYSTEMRAPPORT, der beskriver fejlen, og sende denne til Regnecentralen.

0100 : ESCAPE
Opstår kun i forbindelse med en HANDLER. Hvis ESC trykkes ned, mens en brugerdefineret HANDLER er ENABLE'd, vil HANDLER'en blive kaldt, og SYS(0) vil have værdien 100.

0101 : ULOVLIG HELTALSVÆRDI
Et sted, hvor systemet forventer et heltal, har en variabel en værdi større end 32768 eller mindre end -32767.

0102 : LOG TIL IKKE-POSITIVT TAL
Forsøg på at udregne logaritmen af et ikke-positivt tal.

0103 : KVADRATROD AF NEGATIVT TAL
Forsøg på at uddrage kvadratroden af et negativt tal.

0104 : DIVISION MED NUL
Forsøg på at dividere med nul eller med et tal, der er så lille, at det opfattes som nul af systemet.

0106 : ARITMETISK OVERLØB
En udregning giver et resultat, der er udenfor RcCOMAL80's talområde, dvs. større end 9.99...E+126 eller mindre end -9.99...E-126.

0107 : ILLEGALT NUMMER
Tallet, der refereres til, er ikke repræsenteret korrekt internt. Fejlen optræder, hvis en tidligere fejlmelding er blevet ignoreret af en PROC-HANDLER.

0108 : INGEN PLADS
Programmet er for stort. Split det f.eks op i externe procedurer.

0109 : TYPEKONFLIKT
Fejl i f.eks. parametrene til en procedure. Eksempel:

```
0010 PROC p(i)
0020 ENDPROC p
0030 EXEC p("tekst")
```

0110 : VARIABEL IKKE ERKLÆRET
Alle tekstvariable, vektorer og matricer skal erklæres.

0111 : VARIABEL ALLEREDE ERKLÆRET
Etiketter, tekstvariable, numerisk variable, procedurer, funktioner og pakker må ikke have samme navn.

0112 : PARAMETER FEJL
Parametrene til en procedure passer ikke til de specificerede, f.eks.

```
0010 PROC tom
0020 ENDPROC tom
0030 EXEC tom(3)
```

0113 : FUNKTIONSVÆRDI UDEFINERET
Funktionens værdi er udefineret for det givne argument, f.eks.

```
0010 FUNC funk(i)
0020   IF i>10 then RETURN i
0030 ENDFUNC funk
0040 j:=funk(2)
```

0114 : ULOVLIG FILIDENTIFIKATOR
Stømnummeret er ikke 1,2,3,4 eller 5, f.eks.

```
0010 OPEN 10,"DATAFIL",READ
```

0115 : ULOVLIG CASE VÆRDI
Argumentet efter CASE er ikke specificeret i nogen WHEN-sætning, og OTHERWISE er ikke opgivet, f.eks.

```
0010 i:=6
0020 CASE i OF
0030 WHEN 1
0040   PRINT "1 er fundet"
0050 ENDCASE
```

0116 : ULOVLIGT HOP
Man må ikke hoppe ind i en konstruktion af typen IF-THEN-ELSE-ENDIF, REPEAT-UNTIL, WHILE-ENDWHILE, PROC-ENDPROC osv. F.eks.

```
0010 GOTO e
0020 IF FALSE THEN
0030 e:
0040   PRINT "FALSE"
0050 ENDIF
```

0117 : IKKE FLERE DATA
For mange variable i READ-sætningerne i forhold til antallet af DATA-elementer, f.eks.

```
0010 READ a,b
0020 DATA 2
```

0118 : FEJL I INPUT
Har brugeren ENABLE'd sin egen HANDLER, og der f.eks. indtastes tekst til en INPUT-sætning, der forventer et tal, vil HANDLER'en blive kaldt, og SYS(0) vil have værdien 118.

0119 : IKKE CLOSED PROC
En EXTERN procedure skal være lukket (CLOSED)

0120 : INDEX FEJL
Et array indiceres udenfor sine grænser.

0121 : FEJL I PRINT USING
Formatstrengen er forkert, f.eks.

```
0010 PRINT USING "§§.§§.§§" : 10
```

0122 : ILLEGAL KOORDINAT
Man har angivet en koordinat udenfor vinduet.

0123 : GRAFIK IKKE INSTALLERET
Man kan ikke benytte grafik-faciliteterne, hvis GSX ikke er loaded.

0124 : DRIVERFEJL
Den refererede grafiske enhed findes ikke i systemet, eller den kan ikke benyttes til at tegne på.

0151 : LAGERKRAV OVER 32 KB
Pakken er større end 32 Kbyte og kan derfor ligge i RcComal80's lager-område.

0152 : FIL IKKE KORREKT
Den angivne fil har et format, der viser, at den ikke kan indholde en pakke.

0153 : LAGERMANGEL
Filen, der indeholder pakken, er så stor, at den ikke kan være i det resterende af maskinens lager.

0154 : VERSIONSFEJL
Pakken har et forkert versionsnummer i forhold til den anvendte version af RcComal80. Pakken skal rettes til og genoversættes, så den passer til dette system.

0155 : PAKKETYPE IKKE KORREKT
Den refererede pakke er hverken skrevet i ASM86, RASM86, PolyPascal eller RcComal80. Pakken kan også være genereret forkert. Se kapitel 12.

0156 : FOR LANGT NAVN
Et navn i pakken er for langt. Undersøg om der er fejl i formatet for pakkehovedet.

0157 : FORKERT TEGN I NAVN
Der er et ulovligt tegn i et navn. Der må være en fejl i formatet for pakkehovedet.

0158 : PAKKENAVN EKSISTERER I FORVEJEN
Det refererede pakkenavn benyttes også af en anden pakke, der er i programlageret.

0159 : IKKE OPDATERET
En RcComal80-pakke bruger en anden pakke, som den ikke kan være testet sammen med. Dette kan skyldes, at der er ændret i pakkehovedet for den anden pakke.

0160 : FORKERT POLYPASCAL VERSION
Den refererede pakke er oversat med en forkert PolyPascal version.

0161 : PROCEDURE IKKE ERKLÆRET
En RcComal80-pakke har ikke en procedure eller funktion, svarende til en PUBLIC.

0162 : EXITPROCEDURE UKORREKT ERKLÆRET
Der er ikke erklæret en korrekt exitprocedure. F.eks kan fejlen være, at der er flere EXITPROC-sætninger, ingen procedure svarende til en EXITPROC-sætningen, eller at proceduren har parametre.

0180 - 0189
Reserveret til fejl i assemblerpakker.

0190 : HÅRD POLYPASCAL FEJL
Der er opstået en kørsels- eller IO-fejl under udførelse af en PolyPascal-pakke. Det er ikke muligt at kalde procedurer fra kommando-niveau eller bruge CON efter fejl 0190. Fejl 0190 må ikke behandles i en HANDLER.

0191 : KONVERTERINGSFEJL
Der er opstået en fejl under konvertering af tal eller strenge mellem RcComal80 og en PolyPascal-pakke.

0192 - 0199
Reserveret til fejl i PolyPascal-pakker.

B.3 I/O-Fejlmeddelelser

I/O-fejl er fejl, der er opstået i forbindelse med indlæsnings- eller udskrivnings-operationer.

0201 : END-OF-FILE
Forsøg på at læse en fil ud over end-of-file markeringen.

0203 : TIMEOUT PÅ ENHED
Den ydre enhed, man forsøgte at referere til, har ikke svaret indenfor en bestemt tidsperiode.

0204 : ULOVLIG OPERATION

0205 : DISKETTEFEJL
En fysisk fejl er opdaget på disketten, f.eks. paritetsfejl.

0206 : TIMEOUT
Systemet kan ikke "få kontakt" med disketten, selv ikke efter, at der er forsøgt flere gange indenfor ca. 2 sekunder.

0207 : DISKETTE OFFLINE
Disketteenheden er f.eks. slukket.

0209 : DISKETTE SKRIVEBESKYTTET
Disketten er skrivebeskyttet, og det er derfor umuligt at bruge kommandoerne WRITE FILE, PRINT FILE, SAVE og LIST.

0211 : ENHED OFFLINE
Den ydre enhed er enten slukket eller forbundet forkert.

0212 : FEJL I FILNAVN
Filnavnet indeholder f.eks. ulovlige tegn.

0213 : FIL EKSISTERER ALLEREDE
Der eksisterer allerede en fil med det angivne navn på disketten.

0214 : FIL EKSISTERER IKKE
Den fil, der refereres til, eksisterer ikke på den angivne diskette.

0215 : DISKEN ER FYLDT OP
Der er ikke mere plads på disketten.

0216 : FIL ALLEREDE I BRUG
Den refererede fil er allerede i brug (f.eks. i en anden konsol)

0217 : IKKE ÅBEN/ALLEREDE ÅBEN
Forsøg på at åbne en strøm med et strømnr, der allerede er åben, eller forsøg på at referere til et strømnr, der ikke er åben.

0218 : RESERVERET
Den ydre enhed (printer eller port) er reserveret af f.eks. et andet program i en anden konsol.

0219 : ÅBNE FILER PÅ ENHED

0220 : ULOVLIGT POST NUMMER
Postnummeret er enten negativt, 0 eller for stort.

0221 : END-OF-RECORD
Forsøg på at læse eller skrive udover record-størrelsen.

0222 : FILEN ER SKRIVEBESKYTTET
Den refererede fil er enten skrivebeskyttet eller forsynet med et password.

0223 : LINIE FOR LANG
Forsøg på at indlæse en linie, der er for lang. (INPUT FILE).

0224 : VÆRTFEJL
Kan kun opstå ved brug af lokalnet. Problemet skal søges på værtsmaskinen.

0225: BRUGERFEJL
Kan kun opstå ved brug af lokalnet. Problemet skal søges på en af brugermaskinerne.

0226: NETFEJL
Kan kun opstå ved brug af lokalnet.

0255 : I/O SYSTEM FEJL
En af de eksterne enheder virker ikke - se ref (1) eller ref (2).

C. Skærm, tastatur og lydkreds

C.1 Skærmstyring

Det er ved hjælp af kontroltegn muligt at udføre forskellige former for skærmstyring. Typiske eksempler kan være sletning af skærm, ændring af tegns lysintensitet, linieskift samt direkte styring af cursoren.

Udførelsen af de ønskede kontrolfunktioner sker ved anvendelse af CHR$(x) i PRINT sætninger.

De vigtigste kontrolfunktioner er nævnt i dette appendix. Derudover henvises til ref (1) eller ref (2): Installation og Vedligeholdelse, Appendix B : Styretegn til skærm.

X	CHR$(X)
4	Indsæt et tegn i linie
5	Slet et tegn i linie
7	'BELL', dvs. hørbart signal
8	Cursor en position til venstre (backspace)
9	Tabulator
10	Linieskift (LF): cursor en position ned
12	Slet skærm (FF): cursor til position (1,1)
13	Vognretur (CR): cursor til første position på linien
24	Cursor en position til højre
26	Cursor en position op
27	Sæt styrekode
29	Cursor til position (1,1) (home up)
30	Slet fra aktuel position til slutningen af linien (EOL)
31	Slet fra aktuel position til slutningen af skærmen (EOF)

En række af disse funktioner kan desuden udføres ved at sende CHR$(27) efterfulgt af et andet styretegn.

X	CHR$(27)+CHR$(X)
65	Cursor en position op
66	Cursor en position ned
67	Cursor en position mod højre
68	Cursor en position mod venstre
69	Slet skærmen
72	Cursor til position (1,1)
74	Slet fra aktuel position til resten af skærmen
75	Slet fra aktuel position til resten af linien
76	Indsæt linie (virker ikke i grafiktilstand)
77	Slet linie

80 Vælg alternativt tegnsæt
81 Vælg standard tegnsæt

Desuden er det muligt at fjerne henholdsvis genskabe statuslinien ved:

X CHR$(27)+CHR$(X)
48 Slet statuslinien nederst på skærmen (det er så muligt at bruge alle 25 linier)
49 Genskab statuslinien

C.1.1 Ændring af tegns udseende på skærmen

Tegn der skrives på skærmen kan bringes til at blinke og/eller fremtræde som invers skrift samt optræde i forskellige intensiteter. Dette styres ved at sende CHR$(27) efterfulgt af et styretegn, hvorefter uddata fra efterfølgende PRINT-sætninger vil rette sig efter de angivne koder.

X	CHR$(27)+CHR$(X)
103	Start understregning
104	Slut understregning
112	Start invers video
113	Slut invers video
114	Kraftig lysintensitet (kun Partner)
115	Start blink
116	Slut blink
117	Normal lysintensitet (kun Partner)
122	Reset faciliteter

C.1.2 Farver

Ændringer af farverne på skærmen foretages ved at angive forgrundsfarven (farven på teksten) og baggrundsfarven

CHR$(27)+"b"+CHR$(X) ændrer forgrundsfarven
CHR$(27)+"c"+CHR$(X) ændrer baggrundsfarven

X	Farveskærm	Monokrom skærm
0	Sort	Sort
1	Blå	
2	Grøn	
3	Cyan (blå + grøn)	
4	Rød	Normal intensitet (kun Partner)
5	Magenta (rød + blå)	
6	Gul (rød + grøn)	
7	Hvid (rød + blå + grøn)	

X	Farveskærm	Monokrom skærm
8	Grå	Lav intensitet (kun Partner)
9	Kraftig lysende blå	
10	Kraftig lysende grøn	
11	Kraftig lysende cyan	
12	Kraftig lysende rød	Høj intensitet (kun Partner)
13	Kraftig lysende magenta	
14	Kraftig lysende gul	
15	Kraftig lysende hvid	

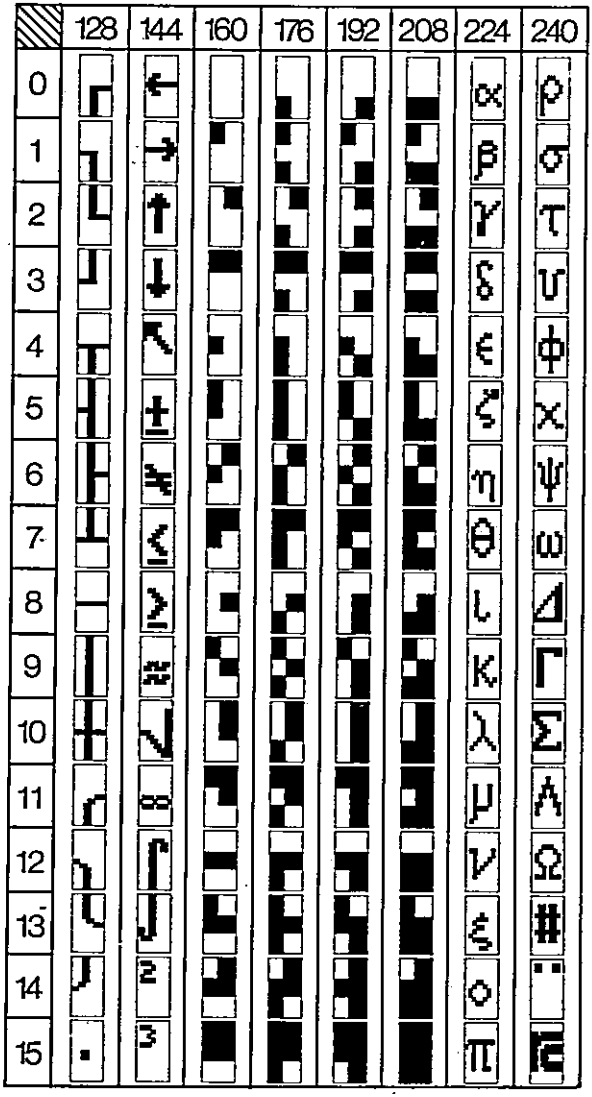
Eksempel
Forgrundsfarven sættes til grøn ved at udføre følgende:

```
0010 PRINT CHR$(27)+"b"+CHR$(2)
```

C.1.3 Tegnsæt

Standardtegnsættet til Partner og PICCOLINE er vist i fig. C-1 og C-2.

Figur C-1: PICCOLINE og Partner tegnsæt



Figur C-2 PICCOLINE og Partner tegnsæt

C.2 Tastatur

Ved anslag leverer tastaturet et 7-bit tegn svarende til ASCII-tabellen i appendix D.

C.2.1 Funktionstaster

Som beskrevet i kap 2.1 indeholder tastaturet en række taster, hvis funktion kan bestemmes af brugeren. Udover disse er der en del andre programmerbare taster (se Appendix C i ref. 1 og 2).

Dette gøres ved at udskrive følgende på "console" (skærmen)

CHR$(27)+":"+CHR$(tast)+nyt indhold+CHR$(0)

tast:	Koden for den tast, man ønsker at omprogrammere (se nedenstående skema)
nyt indhold:	En tekststreng, der indeholder det, man ønsker tasten skal sende. F1 - F12 tasterne kan programmeres med op til 20 tegn, resten kan programmeres med op til 4 tegn

Eksempel
```
10 PRINT CHR$(27)+":"+CHR$(59)+"F1 tast nedtrykket"+CHR$(0)
```

Bemærkning
Når ovenstående program har været kørt, "sender" F1-tasten teksten "F1 tast nedtrykket"

Hvis man omprogrammerer funktionstasterne vil de "nye" værdier gælde indtil man omprogrammerer på ny.

Gravering	Kode	Standard-indhold	Forklaring
F1	59	List<retur>	
F2	60	Auto<retur>	
F3	61	Con<retur>	
F4	62	Del	
F5	63	Delete "	
F6	64	Enter "	
F7	65	Print	
F8	66	Rename "	
F9	67	Renumber<retur>	
F10	68	Save<retur>	
F11	69	Load<retur>	
F12	70	Run<retur>	
↖	71	CHR$(29)	Home
↑	72	CHR$(26)	Cursor en position op
A1	73	CHR$(12)	Slet skærm
A2	74	CHR$(31)	Slet resten af skærmen
←	75	CHR$(8)	Cursor en pos til venstre
→	77	CHR$(24)	Cursor en pos til højre
A3	78	CHR$(30)	Slet resten af linien
A4	79	CHR$(5)	Slet et tegn
↓	80	CHR$(10)	Cursor en linie ned
→|	81	CHR$(9)	Tabuler
TEGN IND	82	CHR$(4)	Indsæt tegn
SLET TEGN	83	CHR$(5)	Slet tegn
PRINT	84		
SHIFT-A1	85		
SHIFT-A2	86		
SHIFT-A3	87	CHR$(20)	Udskriv forrige programlinie
SHIFT-A4	88	CHR$(21)	Udskriv næste programlinie
ALT-F11	89		
ALT-F12	90		
SHIFT-F11	91		
SHIFT-F12	92		
CTRL-F11	93		
CTRL-F12	94		
ALT-F1	97		
ALT-F2	98		
ALT-F3	99		
ALT-F4	100		
ALT-F5	101		
ALT-F6	102		
ALT-F7	103		
ALT-F8	104		
ALT-F9	105		
ALT-F10	106		
SHIFT-F1	107		
SHIFT-F2	108		

Gravering	Kode	Standard-indhold	Forklaring
SHIFT-F3	109		
SHIFT-F4	110		
SHIFT-F5	111		
SHIFT-F6	112		
SHIFT-F7	113		
SHIFT-F8	114		
SHIFT-F9	115		
SHIFT-F10	116		
CTRL-F1	117		
CTRL-F2	118		
CTRL-F3	119		
CTRL-F4	120		
CTRL-F5	121		
CTRL-F6	122		
CTRL-F7	123		
CTRL-F8	124		
CTRL-F9	125		
CTRL-F10	126		

C.3 Lydkreds

Partner og PICCOLINE er udstyret med en trestemmig tonegenerator. Man kommunikerer med tonegeneratoren via den "ydre enhed" SOUND. Nedenstående procedurer viser, hvordan man kan "spille" enten en bestemt frekvens eller en bestemt tone.

Lydkredsens kanaler er nummereret 0, 1 og 2.

```
0010 PROC init_sound
0020   sound_clock:= 2000000/32 // klokfrekvensen er 2MHz/32
0030   sound_stream:= 5 // vælg strømnr til tonegenerator
0040   sound_fac:= 2'(1/12)
0050   sound_c:= sound_clock/220*sound_fac'3
0060   OPEN FILE sound_stream,"Sound", WRITE
0070 ENDPROC init_sound
0080
0090 PROC hz(c,f) // kanal(0,1,2), frekvens
0100   sound_temp:= INT(sound_clock/f+0.5)
0110   PRINT FILE sound_stream: CHR$(128+c*32+sound_temp MOD 16);
0120   PRINT FILE sound_stream: CHR$(sound_temp DIV 16);
0130 ENDPROC hz
0140
0150 PROC atten(c,s) // kanal(0,1,2), styrke (0:høj,...,15:lav)
0160   PRINT FILE sound_stream: CHR$(144+c*32+s);
0170 ENDPROC atten
```

```
0180
0190 PROC silence
0200   atten(0,15)
0210   atten(1,15)
0220   atten(2,15)
0230   atten(3,15)
0240 ENDPROC silence
0250
0260 PROC tone(c,o,t) // kanal, oktav, tone
0270   // kanal er enten 0, 1 eller 2
0270   // oktav er mellem -2 og 2, 0 indeholder miderste c
0280   // tone er mellem 0 og 11 (c er 0, c# er 1, etc.)
0290   hz(c,sound_c*sound_fac't*2'o)
0300 ENDPROC tone
```

D. ASCII tegnsættet

ASCII tegnene samt deres decimale og hexadecimale værdier

DEC	HEX	TEGN	DEC	HEX	TEGN	DEC	HEX	TEGN	DEC	HEX	TEGN
0	00	NUL	32	20	SP	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	§	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	Æ	123	7B	æ
28	1C	FS	60	3C	<	92	5C	Ø	124	7C	ø
29	1D	GS	61	3D	=	93	5D	Å	125	7D	å
30	1E	RS	62	3E	>	94	5E	Ü	126	7E	ü
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

E. Store programeksempler

Dette afsnit indeholder en række store programeksempler, der illustrerer nogle af de mange muligheder man har for programmering i RcComal80.

E.1 Tænker du på et dyr ?

Det første programeksempel er et spil. Programmet stiller spilleren spørgsmål og forsøger at gætte hvilket dyr, spilleren tænker på. Hvis programmet gætter forkert, vil det spørge om forskellen mellem det dyr, den tænkte på, og det dyr spilleren tænkte på. Dermed kender programmet endnu et dyr.

Programmet her har den ulempe, at det ikke kan huske dyrene fra RUN til RUN.

```
0010 DIM SVAR$ OF 30, nytsvar$ OF 30, ind$ OF 30 // strengvariabel
0020 DIM spørgsmål$(100) OF 30 // teksttabel
0030 DIM træ(100,3) // matrix
0040 ophav:= 3; venst:= 1; højre:= 2; max:= 2; dommedag:= FALSE
0050 spørgsmål$(2):= "en elefant"
0060 træ(1,venst):= 2; træ(2,ophav):= 1
0070 REPEAT
0080   REPEAT
0090     INPUT "Tænker du på et dyr ? ": svar$
0100     IF svar$="nej" THEN GOTO farvel
0110   UNTIL svar$="ja"
0120   knude:= træ(1,venst); slut:= FALSE
0130   REPEAT
0140     REPEAT
0150       PRINT spørgsmål$(knude);
0160       INPUT " ? ": svar$
0170     UNTIL svar$="ja" OR svar$="nej"
0180     IF svar$="ja" THEN
0190       slut:= (NOT træ(knude,venst))
0200       IF træ(knude,venst) THEN
0210         knude:= træ(knude,venst)
0220       ENDIF
0230     ELSE
0240       slut:= (NOT træ(knude,højre))
```

```
0250          IF træ(knude,højre) THEN
0260            knude:= træ(knude,højre)
0270          ELSE
0280            EXEC indsknude
0290          ENDIF
0300        ENDIF
0310      UNTIL slut
0320    UNTIL dommedag
0330    farvel:
0340    PRINT "Nå men så farvel for denne gang"
0350    END
0360    PROC indsknude
0370      max:= max+1
0380      parent:= træ(knude,ophav)
0390      IF træ(parent,venst)=knude THEN
0400        træ(parent,venst):=max
0410      ELSE
0420        træ(parent,højre):= max
0430      ENDIF
0440      træ(knude,ophav):= max
0450      INPUT "Hvad er det så ? ": nytsvar$
0460      PRINT "Hvad skal jeg spørge om for at kende forskel på"
0470      PRINT spørgsmål$(knude);" og ";nytsvar$
0480      INPUT " ? ": ind$
0490      REPEAT
0500        PRINT "og hvad er svaret for ";nytsvar$
0510        INPUT " ? ": svar$
0520      UNTIL svar$="ja" OR SVAR$="nej"
0530      spørgsmål$(max):= ind$
0540      træ(max,ophav):= parent
0550      træ(max,venst):= (max+1)*(svar$="ja")+knude*(svar$="nej")
0560      træ(max,højre):= knude*(svar$"=ja")+(max+1)*(svar$="nej")
0570      max:= max+1
0580      spørgsmål$(max):= nytsvar$
0590      træ(max,ophav):= max-1
0600    ENDPROC indsknude
```

```
RUN
Tænker du på et dyr ? ja
en elefant ? nej
Hvad er det så ? en hund
Hvad skal jeg spørge om for at kende forskel på
en elefant og en hund ? har det en snabel
og hvad er svaret for en hund ? nej
Tænker du på et dyr ? ja
har det en snabel ? nej
en hund ? nej
Hvad er det så ? en kat
```

```
Hvad skal jeg spørge om for at kende forskel på
en hund og en kat ? får det killinger
og hvad er svaret for en kat ? ja
Tænker du på et dyr ? ja
har det en snabel ? nej
får det killinger ? ja
en kat ? ja
Tænker du på et dyr ? nej
Nå men så farvel for denne gang
END
at 0340
```

E.2 Tænker du på et dyr ? (filudgave)

Andet programeksempel svarer til det første bortset fra, at de tidligere dyr og spørgsmål gemmes i en fil. Før programmet kan udføres, skal datafilen oprettes. Det gøres med følgende program

```
0010 CREATE "anima",100 // tallet bestemmes helt af brugeren
0020 OPEN 1,"anima",RANDOM 64
0030 WRITE FILE 1,1:"",2,0,2
0040 WRITE FILE 1,2:"en elefant",0,0,1
0050 CLOSE
```

Herefter følger det egentlige program:

```
0010 DIM SVAR$ OF 38, nytsvar$ OF 38, ind$ OF 38// strengvariable
0020 DIM spørgsmål$ OF 38
0030
0040 dommedag:= FALSE
0050 OPEN FILE 1,"anima", RANDOM 64
0060 READ FILE 1,1: spørgsmål$,knude,højre,max
0070 REPEAT
0080   READ FILE 1,1: spørgsmål$,knude,højre,ophav
0090   REPEAT
0100     INPUT "Tænker du på et dyr? ":svar$
0110     IF svar$="nej" THEN GOTO farvel
0120   UNTIL svar$="ja"
0130   slut:= FALSE
0140   REPEAT
0150     READ FILE 1,knude: spørgsmål$,venst,højre,ophav
0160     REPEAT
0170       PRINT spørgsmål$;
```

```
0180        INPUT " ? ": svar$
0190      UNTIL svar$="ja" OR svar$="nej"
0200      IF svar$="ja" THEN
0210        slut:= (NOT venst)
0220        IF venst THEN
0230          knude:= venst
0240        ENDIF
0250      ELSE
0260        slut:= (NOT højre)
0270        IF højre THEN
0280          knude:= højre
0290        ELSE
0300          EXEC indsknude
0310        ENDIF
0320      ENDIF
0330    UNTIL slut
0340  UNTIL dommedag
0350  farvel:
0360  PRINT "Nå men så farvel for denne gang"
0370  READ FILE 1,1: spørgsmål$,venst,højre,ophav
0380  WRITE FILE 1,1: spørgsmål$,venst,højre,max
0390  CLOSE
0400  END
0410  PROC indsknude
0420    max:=max+1
0430    parent:=ophav
0440    READ FILE 1,parent: spørgsmål$,venst,højre,ophav
0450    IF venst=knude THEN
0460      venst:=max
0470    ELSE
0480      højre:=max
0490    ENDIF
0500    WRITE FILE 1,parent: spørgsmål$,venst,højre,ophav
0510    READ FILE 1,knude: spørgsmål$,venst,højre,ophav
0520    ophav:=max
0530    INPUT "Hvad er det så ? ": nytsvar$
0540    PRINT "Hvad skal jeg spørge om for at kende forskel på"
0550    PRINT spørgsmål$;" og ";nytsvar$;
0560    INPUT " ? ": ind$
0570    WRITE FILE 1,knude: spørgsmål$,venst,højre,ophav
0580    REPEAT
0590      PRINT "og hvad er svaret for ";nytsvar$
0600      INPUT " ? ": svar$
0610    UNTIL svar$="ja" OR SVAR$="nej"
0620    spørgsmål$:=ind$
0630    ophav:=parent
0640    venst:=(max+1)*(svar$="ja")+knude*(svar$="nej")
0650    højre:=knude*(svar$="ja")+(max+1)*(svar$="nej")
```

```
0660    WRITE FILE 1,max: spørgsmål$,venst,højre,ophav
0670    max:=max+1
0680    spørgsmål$:=nytsvar$
0690    ophav:=max-1
0700    WRITE FILE 1,max: nytsvar$,0,0,ophav
0710 ENDPROC indsknude
```

E.3 Data-indtastning

Her er et eksempel på en extern procedure til et procedurebibliotek.

Proceduren læser data fra skærmen i visse faste felter, som brugeren angiver. Parametrene til proceduren er følgende:

```
PROC dataentry(n,REF t$(),REF v$(),REF b$())
```

n antal inputfelter på skærmen
t$ teksttabel indeholdende ledetekst
v$ teksttabel indeholdende standardværdier ved kald og aktuelle værdier ved returnering
b$ teksttabel til beskrivelse af de enkelte felter, så
b$(i)(1:1) = CHR$(x-koordinat til feltets start)
b$(i)(2:2) = CHR$(y-koordinat til feltets start)
b$(i)(3:3) = CHR$(inputfeltets maximale længde)

I selve proceduren kan man benytte cursorpilene, Retur-tasten og de almindelige taster. Man vender tilbage til hovedprogrammet ved at trykke på SLET-TEGN-tasten.

Hvis det lyder lidt indviklet så prøv dette lille program, der benytter proceduren:

```
0010 PROC dataentry(n,REF t$(),REF v$(),REF b$()) EXTERNAL "de"
0020 ant:=4
0030 DIM tekst$(ant) OF 40,stdvær$(ant) OF 40,besk$(ant) OF 3
0040 FOR i:=1 TO ant DO
0050   READ tekst$(i),stdvær$(i),y,x,lgd
0060   besk$(i):=CHR$(x)+CHR$(y)+CHR$(lgd)
0070 NEXT i
0080 EXEC dataentry(ant,tekst$,stdvær$,besk$)
0090 PRINT CHR$(12)
```

```
0100 FOR i:=1 TO ant do PRINT stdvær$(i)
0110 DATA "Navn","",10,10,40
0120 DATA "Addr","",11,10,40
0130 DATA "By  ","",12,10,20
0140 DATA "Telf","(02)658000",20,10,10



0010 PROC dataentry(n,ref t$(),REF v$(),REF b$()) CLOSED
0020   PRINT CHR$(12)
0030   MARGIN 0
0040   DIM CH$ OF 1
0050   pilv:= 8; piln:= 10; pilh:= 24; pilo:= 26
0060   OPEN 1,"keyboard", READ
0070   FOR i:= 1 TO n DO
0080     PRINT AT(ORD(b$(i)(1:)),ORD(b$(i)(2:)));t$(i);
0090     FOR j:= 1 TO ORD(b$(i)(3:)) DO PRINT ".";
0100     PRINT AT(ORD(b$(i)(1:))+LEN(t$(i)),ORD(b$(i)(2:)));
0101     PRINT v$(i)
0110   NEXT i
0120   felt:= 1; pos:= 1
0130   PRINT AT(ORD(b$(felt)(1:))+LEN(t$(felt)),ORD(b$(felt)(2:)));
0140   REPEAT
0150     ch$:= GET$(1,1)
0160     char:= ORD(ch$)
0170     CASE TRUE OF
0180     WHEN char=pilv
0190       IF pos>1 THEN
0200         pos:= pos-1
0210         PRINT ch$;
0220       ENDIF
0230     WHEN char=pilh
0240       IF pos<=LEN(v$(felt)) AND pos<ORD(b$(felt)(3:3)) THEN
0250         pos:= pos+1
0260         PRINT ch$;
0270       ENDIF
0280     WHEN char=pilo
0290       felt:= felt-1
0300       IF felt=0 THEN felt:= n
0309       xxx:=ORD(b$(felt)(1:))+LEN(t$(felt))
0310       PRINT AT(xxx,ORD(b$(felt)(2:)));
0320       pos:= 1
0330     WHEN char=piln,char=13
0340       felt:= felt+1
0350       IF felt>n THEN felt:= 1
0359       xxx:=ORD(b$(felt)(1:1))+LEN(t$(felt))
0360       PRINT AT(xxx,ORD(b$(felt)(2:2)));
0370       pos:=1
```

```
0380      WHEN char>=32 AND char<=127
0390        IF pos<=ORD(b$(felt)(3:3)) THEN
0400          v$(felt),(pos:pos):= ch$
0410          pos:= pos+1
0420          PRINT ch$;
0430        ENDIF
0440      OTHERWISE
0450        PRINT CHR$(7);
0460      ENDCASE
0470    UNTIL char=5
0480    CLOSE 1
0490  ENDPROC dataentry
```

E.4 Sortering (PROC-EXTERNAL)

Her er et andet eksempel på en extern procedure, der "hører hjemme" i et procedure-bibliotek.

Proceduren sorterer en fil, der består af en række strenge udskrevet med WRITE FILE. Proceduren sorterer filen i alfabetisk rækkefølge.

Parametrene til proceduren er:

PROC sort(ifil$,ufil$,lgd)

ifil$	inputfil, skrevet i ovenstående format
ufil$	outputfil, må ikke eksistere i forvejen
lgd	den maximale længde af strengen i filen

```
0010 PROC sort(ifil$,ufil$,lgd) CLOSED
0020   PROC quicksort(fra,til)
0030     s:=1; stak(1,1):=fra; stak(1,2):=til
0040     REPEAT
0050       venstr:=stak(s,1); højre:=stak(s,2); s:=s-1
0060       REPEAT
0070         i:=venstr; j:=højre; x$:=navne$((i+j) DIV 2)
0080         REPEAT
0090           WHILE navne$(i)<x$ DO i:=i+1
0100           WHILE x$<navne$(j) DO j:=j-1
0110           IF i<=j THEN
0120             sk$:=navne$(i); navne$(i):=navne$(j)
```

```
0130              navne$(j):=sk$; i:=i+1; j:=j-1
0140            ENDIF
0150          UNTIL i>j
0160          IF j-venstr<højre-i THEN
0170            IF i<højre THEN
0180              s:=s+1; stak(s,1):=i; stak(s,2):=højre
0190            ENDIF
0200            højre:=j
0210          ELSE
0220            IF venstr<j THEN
0230              s:=s+1; stak(s,1):=venstr; stak(s,2):=j
0240            ENDIF
0250            venstr:=i
0260          ENDIF
0270        UNTIL venstr>=højre
0280      UNTIL s=0
0290    ENDPROC quicksort
0300
0310    PROC flet(slut)
0320      RENAME ufil$,"sort0001"
0330      pil1:=1; pil2:=1
0340      OPEN FILE 1,"sort0001", READ
0350      OPEN FILE 2,ufil$, WRITE
0360      READ FILE 1: n1$
0370      WHILE pil2<slut AND NOT EOF(1) DO
0380        IF n1$<navne$(pil2) THEN
0390          WRITE FILE 2: n1$
0400          READ FILE 1:n1$
0410        ELSE
0420          WRITE FILE 2: navne$(pil2)
0430          pil2:=pil2+1
0440        ENDIF
0450      ENDWHILE
0460      IF NOT EOF(1) THEN
0470        REPEAT
0480          WRITE FILE 2: n1$
0490          READ FILE 1: n1$
0500        UNTIL EOF(1)
0510      ELSE
0520        FOR i:=pil2 TO slut DO WRITE FILE 2: navne$(i)
0530      ENDIF
0540      CLOSE FILE 1
0550      CLOSE FILE 2
0560      DELETE "sort0001"
0570    ENDPROC flet
0580
0590    OPEN FILE 1,ufil$, WRITE
0600    CLOSE FILE 1
```

```
0610
0620   max:=INT((SYS(4)-4*lgd-1088)/(4+lgd))
0630   DIM navne$(max) OF lgd,n1$ OF lgd,n2$ OF lgd
0640   DIM x$ OF lgd,sk$ OF lgd,stak(10,2)
0650
0660   OPEN FILE 3,ifil$, READ
0670   antal:=0
0680   REPEAT
0690     n:=1
0700     WHILE (NOT EOF(3)) AND n<=max DO
0710       READ FILE 3: navne$(n)
0720       n:=n+1
0730     ENDWHILE
0740     n:=n-1
0750     IF EOF(3) THEN n:=n-1; antal:=antal-1
0760     IF n<>0 THEN
0770       EXEC quicksort(1,n)
0780       EXEC flet(n)
0790     ENDIF
0800   UNTIL EOF(3)
0810   CLOSE FILE 3
0820 ENDPROC sort
```

E.5 Liv

Her er et eksempel på simulering af amøbers vækst. Udgangspunktet er et kvadrat, der er befolket med en række amøber. Man ændrer fordelingen af amøber efter visse regler, hvorefter man udskriver den næste tilstand (generation). Således fortsættes indtil tilstanden er stationær eller et maximalt antal generationer er nået.

Reglerne for en amøbes udvikling er:

- hvis en amøbe har mere end 3 naboer, dør den af sult
- hvis en amøbe har mindre end 2 naboer, dør den af kedsomhed.
- hvis en amøbe har 2 eller 3 naboer, lever den videre
- hvis et tomt felt har præcis 3 naboer, fødes en ny amøbe.

```
0010 INPUT "Maximalt antal generationer >": maxgen
0020 READ stør
0030 DIM mat1(stør+2,stør+2),mat2(stør+1,stør+1)
0040
0050 gen:= 0
```

```
0060 stabil:= FALSE
0070 EXEC læsmat2
0080
0090 PRINT CHR$(12)
0100 EXEC udskriv
0110 WHILE gen<maxgen AND NOT stabil DO
0120   EXEC matlligmat2
0130   EXEC dannygeneration
0140   EXEC udskriv
0150 ENDWHILE
0160
0170 PROC læsmat2
0180   FOR i:= 2 TO stør+1 DO
0190   FOR j:= 2 TO stør+1 DO READ mat2(i,j)
0200   NEXT i
0210 ENDPROC læsmat2
0220
0230 PROC matlligmat2
0240   FOR i:= 2 TO stør+1 DO
0250     FOR j:= 2 TO stør+1 DO mat1(i,j):= (mat2(i,j)<>0)
0260   NEXT i
0270 ENDPROC matlligmat2
0280
0290 PROC udskriv
0300   PRINT AT(1,1);"Generation nr.";gen
0310   FOR j:= 1 TO stør+2 DO PRINT "+";
0320   PRINT
0330   FOR i:=2 TO stør+1 DO
0340     PRINT "+";
0350     FOR j:= 2 TO stør+1 DO
0360       IF mat2(i,j)=0 THEN
0370         PRINT " ";
0380       ELSE
0390         PRINT "*";
0400       ENDIF
0410     NEXT j
0420     PRINT "+"
0430   NEXT i
0440   FOR j:= 1 TO stør+2 DO PRINT "+";
0450   PRINT
0460 ENDPROC udskriv
0470
0480 PROC dannygeneration
0490   stabil:= TRUE
0500   FOR i:= 2 TO stør+1 DO
0510     venstre:=0
0520     midt:= mat1(i-1,2)+mat1(i,2)+mat1(i+1,2)
0530     FOR j:= 2 TO stør+1 DO
```

```
0540        højre:= mat1(i-1,j+1)+mat1(i,j+1)+mat1(i+1,j+1)
0550        n:= venstre+midt+højre-mat1(i,j)
0560        CASE n OF
0570        WHEN 2
0580          mat2(i,j):= mat1(i,j) // uændret tilstand
0590        WHEN 3
0600          mat2(i,j):= 1 // ny bakterie fødes
0610        OTHERWISE
0620          mat2(i,j):= 0
0630        ENDCASE
0640        IF mat1(i,j)<>mat2(i,j) THEN stabil:= FALSE
0650          venstre:= midt; midt:= højre
0660      NEXT j
0670    NEXT i
0680    gen:= gen+1
0690  ENDPROC dannygeneration
0700  DATA 5
0710  DATA 0,0,0,0,0
0720  DATA 0,1,1,1,0
0730  DATA 1,0,1,0,1
0740  DATA 0,1,1,1,0
0750  DATA 0,0,0,0,0
```

E.6 Tænk på et tal !

Dette programeksempel er også et spil. Spilleren skal tænke på et tal, hvorefter maskinen vil udskrive en række skemaer med tal og spørge, om det tænkte tal er blandt de udskrevne. Efter at have udskrevet skemaer et antal gange, fortæller maskinen hvilket tal spilleren tænkte på.

```
0010 PROC opstart
0020   toerpotens:= 6
0030   max:= afrund(2'toerpotens)
0040   PRINT CHR$(12);"Tænk på et tal mellem 0 og ";max-1
0050   DIM svar$ OF 3
0060   ZONE 10
0070   MARGIN 0
0080 ENDPROC opstart
0090
0100 FUNC afrund(x)
0110   RETURN INT(x+0.5)
0120 ENDFUNC afrund
0130
```

```
0140 PROC udvælgtal(potens)
0150   PRINT AT(1,2);CHR$(31)
0160   offer:= afrund(2'potens)
0170   pil:= 1
0180   FOR i:= 1 TO max DO
0190     IF (i DIV offer) MOD 2=1 THEN PRINT i,
0200   NEXT i
0210   PRINT
0220 ENDPROC udvælgtal
0230
0240 EXEC opstart
0250 svaret:= 0
0260 FOR gang:= 0 TO toerpotens-1 DO
0270   EXEC udvælgtal(gang)
0280
0290   REPEAT
0300     PRINT AT(1,20);CHR$(31);
0310     INPUT "Var tallet blandt de udskrevne (ja/nej) ": svar$
0320   UNTIL svar$="ja" OR svar$="nej"
0330   IF svar$="ja" THEN svaret:= svaret+afrund(2'gang)
0340 NEXT gang
0350 PRINT "Du tænkte på ";svaret
0360 END
```

E.7 Eksempel fra procedureafsnittet

Som lovet i kapitel 8 bringer vi her en total udskrift af en mulig løsning på problemet omkring elevregistreringen.

Først skal datafilerne oprettes. Det gøres f.eks. med følgende program:

```
0010 PRINT CHR$(12);"Oprettelse af kartoteksfiler"
0020 READ klasselgd,navnlgd,adresselgd
0030 READ postnrlgd,maxelev
0040
0050 DATA 5,40,40,40,500
0060
0070 DIM status$ OF maxelev
0080 FOR i:=1 TO maxelev DO status$:=status$+"L" // Ledig
0090
0100 DELETE "elevoplys"
0110 DELETE "elevdata"
0120
```

```
0130 postlængde:=klasselgd+2+navnlgd+2+adresselgd+2+postnrlgd+2
0140 CREATE "elevdata",postlængde*maxelev/1024
0150 OPEN FILE 1,"elevoplys", WRITE
0160 WRITE FILE 1: klasselgd,navnlgd,adresselgd,postnrlgd,maxelev
0170 WRITE FILE 1: status$
0180 CLOSE
0190
0200 PRINT "Slut på oprettelse"
0210 END
```

Hovedprogram:

```
0010 EXEC opstart
0020 REPEAT
0030   PRINT
0040   INPUT "Kommando: I(nd,R(et,S(let,F(ærdig ":k$
0050   PRINT
0060   CASE k$ OF
0070   WHEN "I","i"
0080     EXEC fåelevoplysn
0090     EXEC findledigtnr(elevnr)
0100     EXEC gemelev(elevnr)
0110   WHEN "R","r"
0120     EXEC fåelevnr
0130     EXEC hentelev(elevnr)
0140     EXEC retelev
0150     EXEC gemelev(elevnr)
0160   WHEN "S","s"
0170     EXEC fåelevnr
0180     EXEC hentelev(elevnr)
0190     EXEC sletelev(elevnr)
0200   WHEN "F","f"
0210     // slut på program
0220   OTHERWISE
0230     PRINT "*** Ulovlig kommando"
0240   ENDCASE
0250 UNTIL k$="F" OR k$="f"
0260 EXEC afslut
0270 END
0280
0290 PROC fåelevnr
0300   REPEAT
0310     INPUT "Elevnr : ":elevnr
0320     ok:=FALSE
0330     IF elevnr>=1 AND elevnr<=max THEN
0340       ok:=(status$(elevnr:elevnr)="O") // optaget
0350     ENDIF
```

```
0360      IF NOT ok THEN PRINT "*** Eleven eksisterer ikke"
0370    UNTIL ok
0380  ENDPROC fåelevnr
0390
0400  PROC fåelevoplysn
0410    REPEAT
0420      INPUT "Klasse     :": klasse$
0430      INPUT "Navn       :": navn$
0440      INPUT "Adresse    :": adresse$
0450      INPUT "Postnr by  :": postnrby$
0460      INPUT "Er oplysningerne korrekte ? (J/N) ": svar$
0470    UNTIL svar$="J" OR svar$="j"
0480  ENDPROC fåelevoplysn
0490
0500  PROC findledigtnr(REF nr)
0510    nr:=1
0520    WHILE nr<=max AND status$(nr:nr)="O" DO nr:=nr+1
0530    PRINT "Eleven har fået nummer ";nr
0540  ENDPROC findledigtnr
0550
0560  PROC hentelev(nr)
0570    READ FILE 1,nr: klasse$,navn$,adresse$,postnrby$
0580  ENDPROC hentelev
0590
0600  PROC gemelev(nr)
0610    WRITE FILE 1,nr: klasse$,navn$,adresse$,postnrby$
0620    status$(nr:nr):="O"
0630  ENDPROC gemelev
0640
0650  PROC sletelev(nr)
0660    EXEC skrivelev
0670    INPUT "Skal eleven slettes ? (J/N) ": svar$
0680    IF svar$="J" OR svar$="j" THEN status$(nr:nr):="S"
0690  ENDPROC sletelev
0700
0710  PROC skrivelev
0720    PRINT "Klasse     :";klasse$
0720    PRINT "Navn       :";navn$
0730    PRINT "Adresse    :";adresse$
0740    PRINT "Postnr by  :";postnrby$
0750  ENDPROC skrivelev
0760
0770  PROC retelev
0780    REPEAT
0790      EXEC skrivelev
0799      PRINT "Ændring: K(lasse,N(avn,A(dresse,";
0800      INPUT "P(ostnr by,F(ærdig ":f$
```

```
0810      CASE f$ OF
0820      WHEN "K","k"
0830        INPUT "Ny klasse      :": klasse$
0840      WHEN "N","n"
0850        INPUT "Nyt navn       :": navn$
0860      WHEN "A","a"
0870        INPUT "Ny adresse     :": adresse$
0880      WHEN "P","p"
0890        INPUT "Ny postnr by :": postnrby$
0900      OTHERWISE
0910        // Ingen aktion
0920      ENDCASE
0930    UNTIL f$="F" OR f$="f"
0940  ENDPROC retelev
0950
0960  PROC opstart
0970    PRINT CHR$(12);"K a r t o t e k s p r o g r a m"
0980    PRINT "------------------------------"
0990    DIM k$ OF 1,svar$ OF 1,f$ OF 1
1000    OPEN FILE 1,"elevoplys", READ
1010    READ FILE 1: klasselgd,navnlgd,adresselgd,postnrlgd,max
1020    DIM klasse$ OF klasselgd,navn$ OF navnlgd
1030    DIM adresse$ OF adresselgd,postnrby$ OF postnrlgd
1040    DIM status$ OF max
1050    READ FILE 1:status$
1060    CLOSE FILE 1
1070    postlængde:=klasselgd+2+navnlgd+2+adresselgd+2+postnrlgd+2
1080    OPEN FILE 1,"elevdata", RANDOM postlængde
1090    elevnr:=0
1100  ENDPROC opstart
1110
1120  PROC afslut
1130    CLOSE FILE 1
1140    OPEN FILE 1,"nyeoplys", WRITE
1150    WRITE FILE 1: klasselgd,navnlgd,adresselgd,postnrlgd,max
1160    WRITE FILE 1: status$
1170    CLOSE FILE 1
1180    DELETE "elevoplys"
1190    RENAME "nyeoplys","elevoplys"
1200    PRINT "Slut på kartoteksprogram"
1210  ENDPROC afslut
```

E.8 Elektronisk orgel

Ved hjælp af Partner og PICCOLINE's indbyggede tonegenerator kan man benytte datamaten som "orgel". Dette programeksempel viser hvordan. De to nederste rækker på tastaturet benyttes som klaviatur, c-tasten svarer til tonen c. Hvis man trykker på SHIFT-tasten samtidig med en anden tast, bliver tonen en oktav højere. Mellemrumstangenten afbryder en igangværende tone og returtasten afbryder programmet.

```
0010 PROC init_sound
0020   sound_clock:= 2000000/32 //  klokfrekvensen er 2MHz/32
0030   sound_stream:= 5 // vælg strømnr til tonegenerator
0040   sound_fac:= 2'(1/12)
0050   sound_c:= sound_clock/220*sound_fac'3
0060   OPEN FILE sound_stream,"Sound", WRITE
0070 ENDPROC init_sound
0080
0090 PROC hz(c,f) // kanal, frekvens
0100   sound_temp:= INT(sound_clock/f+0.5)
0110   PRINT FILE sound_stream: CHR$(128+c*32+sound_temp MOD 16);
0120   PRINT FILE sound_stream: CHR$(sound_temp DIV 16);
0130 ENDPROC hz
0140
0150 PROC atten(c,s) // kanal, styrke
0160   PRINT FILE sound_stream: CHR$(144+c*32+s);
0170 ENDPROC atten
0180
0190 PROC silence
0200   atten(0,15)
0210   atten(1,15)
0220   atten(2,15)
0230   atten(3,15)
0240 ENDPROC silence
0250
0260 PROC esc HANDLER
0270   PRINT "Error ";ERRTEXT$(ERR)
0280   PRINT "At ";SYS(2)
0290   silence
0300   CLOSE FILE sound_stream
0310 ENDPROC esc
0320
0330 FUNC freq(o,t)
0340   RETURN sound_c*sound_fac't*2'o
0350 ENDFUNC freq
0360
```

```
0370 init_sound
0380 ENABLE esc
0390 DATA "c","üazsxcfvgbnjmk,l./ø","C","ÜAZSXCFVGBNJMK<L>?Ø"
0400 DIM linie$ OF 130,k$ OF 1
0410 DIM skala(0:255),c$ OF 1
0420 FOR startoktav:= 0 TO 1 DO
0430   READ c$,linie$
0440   oct:= startoktav; t:= 0
0450   pos:= c$ IN linie$
0460   WHILE pos<=LEN(linie$) DO
0470     skala(ORD(linie$(pos:))):= freq(oct,t)
0480     t:= t+1; pos:= pos+1
0490     IF t>11 THEN t:= 0; oct:= oct+1
0500   ENDWHILE
0510   pos:= (c$ IN linie$); t:= 0; oct:= startoktav
0520   WHILE pos>=1 DO
0530     skala(ORD(linie$(pos:))):= freq(oct,t)
0540     t:= t-1; pos:= pos-1
0550     IF t<0 THEN t:= 11; oct:= oct-1
0560   ENDWHILE
0570 NEXT startoktav
0580
0590 niveau:= 15
0600 REPEAT  // Spilleløkke
0610   k$:= KEY$
0620   CASE ORD(k$) OF
0630   WHEN 0
0640     niveau:= niveau+0.1 // gør lyden svagere
0650     IF niveau>15 THEN niveau:= 15
0660     atten(1,niveau)
0670   WHEN 32
0680     niveau:= 15 // mellemrum stopper lyden
0690     atten(1,niveau)
0700   OTHERWISE
0710     lyd:= skala(ORD(k$))
0720     IF lyd<>0 THEN
0730       hz(1,skala(ORD(k$)))
0740       niveau:= 0
0750       atten(1,niveau)
0760     ENDIF
0770   ENDCASE
0780 UNTIL k$=CHR$(13) // retur afbryder spillet
0790 silence
0800 CLOSE FILE sound_stream
```

E.9 Enarmet tyveknægt

Dette programeksempel fungerer ligesom en enarmet tyveknægt, bortset fra, at det er lidt for rundhåndet med gevinster.

Programmet er ret stort, men det giver også spilleren mulighed for at holde de enkelte hjul for at forbedre sine gevinstchancer. Ved hjælp af skærmstyringen illuderer programmet de tre hjul, der drejer rundt.

Programmet benytter det semigrafiske tegnsæt til at danne spillesymbolerne. Disse hentes ind fra disketten. Før man kører programmet skal man derfor danne symbolerne. Det gøres med følgende program:

```
0010 OPEN FILE 1,"tyvsymbol", WRITE
0020 DIM t$ OF 18
0030 // Definering af grafikstrenge
0040 FOR i:=0 TO 5 DO
0050   FOR j:=1 TO 7 DO
0060     FOR k:=1 TO 18 DO
0070       READ tegnværdi
0080       t$(k:):=CHR$(tegnværdi+128)
0090     NEXT k
0100     WRITE FILE 1:tt$
0110   NEXT j
0120 NEXT i
0130 CLOSE
0140
0150 // grafik for 'BAR'
0160 DATA 32,124,124,124,124,124,124,124,124
0170 DATA 124,124,124,124,124,124,124,124,32
0180 DATA 32,127,35,35,35,43,127,39,35
0190 DATA 35,43,127,35,35,35,43,127,32
0200 DATA 32,127,32,127,53,32,127,32,127
0210 DATA 127,32,127,32,127,127,32,127,32
0220 DATA 32,127,32,115,49,40,127,32,115
0230 DATA 115,32,127,32,99,115,112,127,32
0240 DATA 32,127,32,127,53,32,127,32,127
0250 DATA 127,32,127,32,116,43,127,127,32
0260 DATA 32,127,112,112,112,120,127,112,127
0270 DATA 127,112,127,112,127,125,114,127,32
0280 DATA 32,47,47,47,47,47,47,47,47
0290 DATA 47,47,47,47,47,47,47,47,32
0300
```

```
0310 // grafik for 'KLØR'
0320 DATA 32,32,32,32,32,32,32,96,124
0330 DATA 124,48,32,32,32,32,32,32,32
0340 DATA 32,32,32,32,32,32,32,43,127
0350 DATA 127,39,32,32,32,32,32,32,32
0360 DATA 32,32,32,32,112,112,32,32,106
0370 DATA 53,32,32,112,112,32,32,32,32
0380 DATA 32,32,32,126,127,127,125,124,126
0390 DATA 125,124,126,127,127,125,32,32,32
0400 DATA 32,32,32,34,47,47,33,32,106
0410 DATA 53,32,34,47,47,33,32,32,32
0420 DATA 32,32,32,32,32,32,32,32,122
0430 DATA 117,32,32,32,32,32,32,32,32
0440 DATA 32,32,32,32,32,96,120,126,127
0450 DATA 127,125,116,48,32,32,32,32,32
0460
0470 // grafik for 'KLOKKE'
0480 DATA 32,32,32,32,32,120,126,127,127
0490 DATA 127,127,125,116,32,32,32,32,32
0500 DATA 32,32,32,32,106,127,127,127,127
0510 DATA 127,127,127,127,53,32,32,32,32
0520 DATA 32,32,32,32,106,127,127,127,127
0530 DATA 127,127,127,127,53,32,32,32,32
0540 DATA 32,32,32,32,106,127,127,127,127
0550 DATA 127,127,127,127,53,32,32,32,32
0560 DATA 32,32,32,32,106,127,127,127,127
0570 DATA 127,127,127,127,53,32,32,32,32
0580 DATA 32,32,112,124,127,127,127,127,127
0590 DATA 127,127,127,127,127,124,112,32,32
0600 DATA 32,32,32,32,32,32,32,43,127
0610 DATA 127,39,32,32,32,32,32,32,32
0615
0620 // grafik for 'BLOMME'
0630 DATA 32,32,32,32,32,32,32,32,32
0640 DATA 32,32,32,32,32,32,32,32,32
0650 DATA 32,32,32,32,112,124,124,127,127
0660 DATA 127,127,124,124,112,32,32,32,32
0670 DATA 32,32,120,127,127,127,127,127,127
0680 DATA 127,127,127,127,127,127,116,32,32
0690 DATA 32,32,127,127,127,127,127,127,127
0700 DATA 127,127,127,127,127,127,127,32,32
0710 DATA 32,32,43,127,127,127,127,127,127
0720 DATA 127,127,127,127,127,127,39,32,32
0730 DATA 32,32,32,32,35,47,47,127,127
0740 DATA 127,127,47,47,35,32,32,32,32
0750 DATA 32,32,32,32,32,32,32,32,32
0760 DATA 32,32,32,32,32,32,32,32,32
0770
```

```
0780 // grafik for 'ÆBLE'
0790 DATA 32,32,32,32,32,32,32,32,32
0800 DATA 32,96,38,32,32,32,32,32,32
0810 DATA 32,32,32,32,32,32,32,32,96
0820 DATA 56,33,32,32,32,32,32,32,32
0830 DATA 32,32,32,32,96,112,112,32,106
0840 DATA 96,112,112,112,32,32,32,32,32
0850 DATA 32,32,32,122,127,127,127,127,127
0860 DATA 127,127,127,127,127,48,32,32,32
0870 DATA 32,32,32,127,127,127,127,127,127
0880 DATA 127,127,127,127,127,53,32,32,32
0890 DATA 32,32,32,111,127,127,127,127,127
0900 DATA 127,127,127,127,127,33,32,32,32
0910 DATA 32,32,32,32,43,47,127,127,127
0920 DATA 127,127,47,39,33,32,32,32,32
0930
0940 // grafik for 'KIRSEBÆR'
0950 DATA 32,32,32,32,32,32,34,100,32
0960 DATA 96,52,32,32,32,32,32,32,32
0970 DATA 32,32,32,32,32,32,32,42,100
0980 DATA 37,32,32,32,32,32,32,32,32
0990 DATA 32,32,32,32,32,32,32,96,37
1000 DATA 41,112,112,112,32,32,32,32,32
1010 DATA 32,32,32,32,32,112,112,53,32
1020 DATA 104,127,127,127,125,32,32,32,32
1030 DATA 32,32,32,32,126,127,127,127,52
1040 DATA 34,111,127,127,39,32,32,32,32
1050 DATA 32,32,32,32,43,127,127,63,33
1060 DATA 32,32,32,32,32,32,32,32,32
1070 DATA 32,32,32,32,32,32,32,32,32
1080 DATA 32,32,32,32,32,32,32,32,32
```

Herefter følger det egentlige program:

```
0010 PROC esc HANDLER
0020   PRINT AT(1,22);
0030   IF ERR=100 THEN
0040     IF NOT escenabled THEN CONTINUE
0050     IF penge>=0 THEN
0060       PRINT CHR$(27);CHR$(128);"      Det var hyggeligt, ";
0070       PRINT "lad mig prøve at slå dig en anden gang !!";
0080     ELSE
0090       PRINT CHR$(27);CHR$(128);"    Det var hyggeligt ";
0100       PRINT "at vinde over dig !!";
0110     ENDIF
0120   ENDIF
0130 ENDPROC esc
```

```
0140 PROC opstart
0150   RANDOMIZE
0160   escenabled:=FALSE; maxi:=17
0170   ENABLE esc
0180   DIM t$(42) OF 18
0190   DIM svar$ OF 1,pos(3),snur(3),hjul(maxi,3)
0200   DIM sletlinie$ OF 1
0210   ZONE 0
0220   MARGIN 0
0230   textmarg:=53
0240   penge:=0; antal:=0; sletlinie$:=CHR$(30)
0250   i1:=RND(1,maxi); i2:=RND(1,maxi); i3:=RND(1,maxi)
0260   bar:=7*0+1; klør:=7*1+1; klokke:=7*2+1; blomme:=7*3+1
0270   æble:=7*4+1; kirsebær:=7*5+1
0280   PRINT CHR$(12)
0290   PRINT AT(1,1);"              G E V I N S T L I S T E"
0300   PRINT AT(1,2);"-------------------------";
0301   PRINT "-------------------------"
0310   PRINT AT(1,3);"! BAR          BAR          ";
0311   PRINT "BAR              200 MØNTER !"
0320   PRINT AT(1,4);"! KLØR         KLØR         ";
0321   PRINT "KLØR             150 MØNTER !"
0330   PRINT AT(1,5);"! KLOKKE       KLOKKE       ";
0331   PRINT "KLOKKE/BAR       18 MØNTER !"
0340   PRINT AT(1,6);"! BLOMME       BLOMME       ";
0341   PRINT "BLOMME/BAR       14 MØNTER !"
0350   PRINT AT(1,7);"! ÆBLE         ÆBLE         ";
0351   PRINT "ÆBLE/BAR         10 MØNTER !"
0360   PRINT AT(1,8);"! KIRSEBÆR     KIRSEBÆR     ";
0361   PRINT "KIRSEBÆR/BAR     10 MØNTER !"
0370   PRINT AT(1,9);"! KIRSEBÆR     KIRSEBÆR     ";
0371   PRINT "----------       5 MØNTER !"
0380   PRINT AT(1,10);"! KIRSEBÆR     --------     ";
0381   PRINT " ----------       2 MØNTER !"
0390   PRINT AT(51,2);"-----------------------------"
0400   FOR i:=3 TO 10 DO PRINT AT(80,i);"!";
0410   PRINT AT(1,11);
0420   FOR i:=1 TO 49 DO PRINT "-";
0430   PRINT " ";
0440   FOR i:=51 to 80 DO PRINT "-";
0450   PRINT CHR$(27);CHR$(140)
0460   x1:=8; x2:=31; x3:=54
0470   pos(1):=x1; pos(2):=x2; pos(3):=x3
0480   PRINT AT(1,14);CHR$(27);CHR$(132);
0481   PRINT AT(1,22);CHR$(27);CHR$(128);
0490   PRINT AT(textmarg,6);" Vent lige lidt,tak"
```

```
0500   // definering af grafikstrenge
0510   OPEN 1,"tyvsymbol", READ
0520   FOR i:=1 TO 42 DO READ FILE 1:t$(i)
0530   CLOSE
0540   FOR i:=1 TO 3 DO
0550     FOR j:=1 TO 17 DO READ hjul(j,i)
0560   NEXT i
0570   DATA 36,29,22,36,15,29,8,36,1,22,15,29,8,36,22,29,1
0580   DATA 22,29,8,36,15,29,1,36,22,29,8,36,22,1,29,15,36
0590   DATA 1,29,15,22,36,8,29,22,36,1,29,15,36,8,22,29,36
0600   PRINT AT(x1-3,13);CHR$(128+0);AT(x3+20,13);CHR$(128+1);
0610   PRINT AT(x1-3,21);CHR$(128+2);AT(x3+20,21);CHR$(128+3);
0620   FOR i:=1 TO 68 DO
0630     PRINT AT(x1-3+i,13);chr$(136);AT(x1-3+i,21);chr$(136);
0640   NEXT i
0650   FOR i:=1 TO 7 DO
0660     FOR j:=0 TO 3 DO PRINT AT(j*23+5,13+i);CHR$(137);
0670   NEXT i
0680   PRINT AT(x1+20,13);CHR$(132);AT(x3-3,13);CHR$(132);
0690   PRINT AT(x1+20,21);CHR$(135);AT(x3-3,21);CHR$(135);
0700   holdt:= FALSE
0710   FOR i:= 1 TO 3 DO snur(i):= TRUE
0720   PRINT AT(textmarg,3);"Tast 1,2 eller 3 for HOLD",
0730   PRINT AT(textmarg,4);"kun når  H O L D ",
0731   PRINT CHR$(27);CHR$(128),"lyser    ";
0740 ENDPROC opstart
0750
0760 PROC snurhjul
0770   antal:= antal+1; penge:= penge-1
0780   h1:=RND(3,8); h2:=RND(h1+1,h1+5); h3:=RND(h2+1,h2+5)
0790   h1:=h1*snur(1); h2:=h2*snur(2); h3:=h3*snur(3)
0800   REPEAT
0810   i1:=(i1+snur(1)) MOD maxi; h1:=h1-snur(1); snur(1):=h1>0
0820   i2:=(i2+snur(2)) MOD maxi; h2:=h2-snur(2); snur(2):=h2>0
0830   i3:=(i3+snur(3)) MOD maxi; h3:=h3-snur(3); snur(3):=h3>0
0840   t1:=hjul(i1+1,1); t2:=hjul(i2+1,2); t3:=hjul(i3+1,3)
0850   FOR linie:=14 TO 20 DO
0860   PRINT AT(x1,linie),t$(t1),
0870   PRINT AT(x2,linie),t$(t2),
0880   PRINT AT(x3,linie),t$(t3),
0890   t1:=t1+1; t2:=t2+1; t3:=t3+1
0900   NEXT linie
0910   UNTIL h1+h2+h3=0
0920   t1:=t1-7; t2:=t2-7; t3:=t3-7
0930   ENDPROC snurhjul
0940
```

```
0950 PROC gevinst
0960   vundet:= 0
0970   CASE t1 OF
0980   WHEN bar
0990     IF t2=bar AND t3=bar THEN vundet:=200
1000   WHEN kløv
1010     IF t2=kløv AND t3=kløv THEN vundet:=150
1020   WHEN klokke
1030     IF t2=klokke AND (t3=klokke OR t3=bar) THEN vundet:=18
1040   WHEN blomme
1050     IF t2=blomme AND (t3=blomme OR t3=bar) THEN vundet:=14
1060   WHEN æble
1070     IF t2=æble AND (t3=æble OR t3=bar) THEN vundet:=10
1080   WHEN kirsebær
1090     vundet:=2
1100     IF t2=kirsebær THEN
1110       vundet:=5
1120       IF t3=kirsebær THEN vundet:=10
1130     ENDIF
1140   ENDCASE
1150   IF vundet>0 THEN
1160     PRINT AT(textmarg,6);"Du har vundet ";vundet;
1170     PRINT " mønter.";
1180     penge:= penge+vundet
1190   ELSE
1200     PRINT AT(textmarg,6);"Du fik ingen gevinst          "
1210   ENDIF
1220   IF penge>=0 THEN
1230     PRINT AT(textmarg,9);"Gevinst ialt: "
1240     PRINT AT(textmarg,10);penge;" mønter";
1250   ELSE
1260     PRINT AT(textmarg,9);"Underskud ialt: "
1270     PRINT AT(textmarg,10);-penge;" mønter";
1280   ENDIF
1290   PRINT " på ";antal;" spil. "
1300 ENDPROC gevinst
1310
1320 PROC holdhjul
1330   PRINT AT(1,22);CHR$(30)
1340   PRINT AT(textmarg+7,4),CHR$(27),CHR$(128);
1350   FOR i:=1 TO 3 DO snur(i):=TRUE
1360   IF NOT holdt AND vundet=0 THEN
1370     holdt:=FALSE
1380     PRINT AT(textmarg+7,4),CHR$(27),CHR$(144),
1390     REPEAT
1400       REPEAT
1410         REPEAT
```

```
1420            svar$:=KEY$
1430          UNTIL ORD(svar$)>0
1440        UNTIL svar$ IN "123" OR ORD(svar$)=13
1450        IF svar$ IN "123" THEN
1460          i:=ORD(svar$)-ORD("0")
1470          snur(i):=NOT snur(i)
1480          IF NOT snur(i) THEN
1490            PRINT AT(pos(i)+2,22);CHR$(27),CHR$(144),
1500            PRINT "  H O L D     ",CHR$(27),CHR$(128)
1510          ELSE
1520            PRINT AT(pos(i)+2,22);"                ";
1530          ENDIF
1540        ENDIF
1550      UNTIL svar$=CHR$(13)
1560      FOR i:=1 TO 3 DO
1570        holdt:=holdt OR snur(i)=0
1580      NEXT i
1590    ELSE
1600      PRINT AT(textmarg+7,4);CHR$(27);CHR$(128)
1610      holdt:= FALSE
1620      REPEAT
1630        svar$:=KEY$
1640      UNTIL ORD(svar$)=13
1650    ENDIF
1660 ENDPROC holdhjul
1670
1680 MARGIN 0
1690 ZONE 0
1700 EXEC opstart
1710 REPEAT
1720    PRINT AT(textmarg,6);" Ok nu kører vi          "
1730    escenabled:=FALSE
1740    EXEC snurhjul
1750    EXEC gevinst
1760    escenabled:=TRUE
1770    EXEC holdhjul
1780 UNTIL FALSE
```

F. Automatisk opstart af programmer

Ved opstart er det muligt, at få et program indlæst og udført automatisk. Programmet gemmes (SAVE) under navnet "LOGON.CSV".

Når RcComal80 startes op ledes efter en fil med navn "LOGON.CSV" på den CCP/M unit, der er startet op fra (dvs. normalt unit A). Findes filen ikke, udskrives "RcComal80 rev x.xx" og systemet er klar til indtastning.

Findes filen derimod, indlæses programmet og udføres.

Ønsker man at loade et andet program end "LOGON" kan man angive filnavnet i kaldet af RcComal80, f.eks.

```
COMAL80 MENU
```

I dette tilfælde vil RcComal80 forsøge at LOADe og udføre en fil med navnet "MENU.CSV".

G. Pakker

Dette kapitel beskriver de pakker, der er med på distributionsdisketterne. Kapitlet er delt i tre afsnit med RcComal80-pakker, PolyPascal-pakker og assemblerpakker.

G.1 RcComal80-pakker

RcComal80-pakker består af en PCK-fil.

G.1.1 Liniegrafik (LILLEGSX)

Pakken ser således ud:

```
PACKAGE lillegsx
   PROC polyline(ant, REF pkti(,))
   PROC polylinetype(type)
ENDPACKAGE lillegsx
```

Pakken indkapsler to af GSX-ordrerne, således at brugeren kan kalde procedurerne, fremfor at skulle huske selve GSX-ordrerne.

PROC polyline(ant, REF pkti(,))

Der tegnes en linie fra første punkt i pkti til andet punkt i pkti, og videre til tredie punkt i pkti, osv. Parameteren ant angiver, hvor mange punkter, der er i parameteren pkti.

PROC polylinetype(type)

Ændrer linietypen. Der er fem muligheder (1 til 5), som er beskrevet under GSX i kapitel 13.

Fejlreaktioner

Ingen.

G.2 PolyPascal-pakker

PolyPascal-pakker er repræsenteret ved to filer på distributionsdisketterne. De to filer har samme primærnavn. Filtypen for kildeteksten er PAS og for den oversatte pakke PCK.

G.2.1 Aflæsning af mus (MUS)

Pakken ser således ud:

```
PACKAGE mus
   PROC musinit
   FUNC musstatus(REF dx, REF dy, REF keyinf)
ENDPACKAGE mus
```

PROC musinit

Initialiserer musen. Dette bliver desuden automatisk udført, som en del af initialiseringen af pakken.

FUNC musstatus(REF dx, REF dy, REF keyinf)

Aflæser status af musen:

0 Intet ændret
1 Der er trykket på en knap. Parameteren keyinf indeholder tastekoden for knappen (30, 31 eller 32)
2 Musen er flyttet. Parametrene dx og dy indeholder den relative ændring

Fejlreaktioner

Ingen.

G.3 Assemblerpakker

Assemblerpakker er repræsenteret ved to filer på distributionsdisketterne. Filerne har samme primærnavn. Filtypen for kildeteksten er A86 og for den oversatte pakke PCK.

G.3.1 Bitoperationer (BITOP)

Pakken ser således ud:

```
PACKAGE bitop
   FUNC bitand(arg1,arg2)
   FUNC bitor(arg1,arg2)
   FUNC bitxor(arg1,arg2)
   FUNC bitnot(arg1)
   FUNC bitshr(arg1,noofbits)
   FUNC bitshl(arg1,noofbits)
ENDPACKAGE bitop
```

Pakkens funktioner udfører logiske operationer på heltals-argumenter. Heltals-argumenterne skal kunne repræsenteres i 16 bit. Den øverste bit benyttes som fortegnsbit. Der benyttes 1-komplement repræsentation.

FUNC bitand(arg1,arg2)

Funktionen udfører en logisk and på bit-repræsentationen af de to heltalsargumenter. Resultatet leveres tilbage som et heltal.

Eksempel

```
print bitand(19,6)          0000000000010011
                            0000000000000110
2                           0000000000000010
```

FUNC bitor(arg1,arg2)

Funktionen udfører en logisk or på bit-repræsentationen af de to heltalsargumenter. Resultatet leveres tilbage som et heltal.

Eksempel

```
print bitor(19,6)           0000000000010011
                            0000000000000110
23                          0000000000010111
```

FUNC bitxor(arg1,arg2)

Funktionen udfører en logisk xor på bit-repræsentationen af de to heltalsargumenter. Resultatet leveres tilbage som et heltal.

Eksempel

```
print bitxor(19,6)          0000000000010011
                            0000000000000110
21                          0000000000010101
```

FUNC bitnot(arg1)

Funktionen udfører en logisk not på bit-repræsentationen af heltalsargumentet. Resultatet leveres tilbage som et heltal.

Eksempel

```
print bitnot(23)            0000000000010111
-24                         1111111111101000
```

FUNC bitshr(arg1,noofbits)

Funktionen udfører et logisk shift til højre på bit-repræsentationen af heltalsargumentet. Der skiftes så mange pladser, som noofbits angiver. Resultatet leveres tilbage som et heltal.

Eksempel

```
print bitshr(23,2)          0000000000010111
5                           0000000000000101
```

FUNC bitshl(arg1,noofbits)

Funktionen udfører et logisk shift til venstre på bit-repræsentationen af heltalsargumentet. Der skiftes så mange pladser, som noofbits angiver. Resultatet leveres tilbage som et heltal.

Eksempel

```
print bitshl(23,2)          0000000000010111
92                          0000000001011100
```

Fejlreaktioner:

0101 en parameter er så stor, at den ikke kan repræsenteres som et heltal i 16 bit

0180 parameteren noofbits er mindre end 0 eller større end 16

G.3.2 Kald af kommandofiler (PROGRAM)

Pakken ser således ud:

```
PACKAGE program
   PROC kommando(comstr$)
ENDPACKAGE program
```

kommando
Pakken indeholder kun proceduren kommando. Denne procedure har en kommando-tegnstreng comstr$, som parameter. Proceduren leder efter en CMD-fil, hvis navn er comstr$. Hvis den findes, udføres det, der ligger på denne fil.

Eksempel

Følgende RcComal80-program installerer grafik-styresystemet.

```
0010 USE program
0020 kommando ("graphics")
```

Bemærkning

Hvis der ikke findes en fil med det angivne navn, sker der intet.

Hvis kommando-tegnstrengen comstr$ indeholder et navn på en af de indbyggede CCP/M86 kommandoer sker der, som nævnt ovenfor intet, fordi disse kommandoer ikke findes på en fil. Man kan få en oversigt over de indbyggede CCP/M86 kommandoer i "PICCOLINE Brugervejledning, Betjening" eller ved at skrive "?" til styresystemet.

Hvis filen med navn comstr$, benytter skærmen, kan cursoren være uheldigt placeret efter udførelsen af filen.

Fejlreaktioner

0180 Programmet kan ikke kaldes, f.eks. fordi filen ikke findes, eller fordi der ikke er plads

G.3.3 Kopi af grafik-skærmbillede på Rc603 printer (COPY603)

Pakken ser således ud:

```
PACKAGE copy603
   PROC skærmkopi
   PROC skærmudsnit(startx,starty,antalx,antaly,indryk)
ENDPACKAGE copy603
```

Pakkens procedurer kan udskrive kopier af et grafik-skærmbillede på en Rc603 printer.

PROC skærmkopi

Proceduren udskriver en kopi af hele grafik-skærmbilledet på en tilknyttet Rc603-printer.

PROC skærmudsnit(startx,starty,antalx,antaly,indryk)

Proceduren udskriver en del af grafik-skærmbilledet på en tilknyttet Rc603-printer. Parametrene angiver hvilken del, der skal udskrives, og hvor langt udskrivningen skal rykkes ind på printeren.

De første 4 parametre angår grafik-skærmbilledets punkter på skærmen. Der måles ud fra øverste venstre hjørne på skærmen. Startx, starty angiver pixel-koordinatsæt for udsnittets startpunkt. (En pixel er et punkt på grafik-skærmbilledet). Antalx angiver, hvor mange kolonner pixels, der skal medtages i x-aksens retning (mod højre på skærmen). Antaly angiver, hvor mange rækker pixels, der skal medtages i y-aksens retning (nedad på skærmen).

Parameteren indryk angiver, hvor stor indrykning, der skal laves på printeren, før skærmudsnittet tegnes. Det anbefales, at give parameteren indryk værdien 30.

Parametrene skal ligge i intervallerne:

	50 Hertz (PICCOLINE-skærm)	60 Hertz (Partner-skærm)
startx	0 - 560	0 - 720
starty	0 - 256	0 - 352
antalx	0 - 560	0 - 720
antaly	0 - 256	0 - 352
indryk	0 - 999	0 - 999

Bemærkning

Udskriften bliver "underlig", hvis man ikke har installeret grafik-systemet, og har brugt RcComal80 sætningen/kommmandoen OPEN GRAPHICS med en parameter, der angiver skærmen.

Udskriften bliver forkert, hvis man skifter konsol under udførelsen af procedurerne.

Fejlreaktioner

Ingen.

G.3.4 Kopi af grafik-skærmbillede på Rc602 eller Rc605 printer (COPY602)

Pakken ser således ud:

```
PACKAGE copy602
   PROC skærmkopi
   PROC skærmudsnit(startx,starty,antalx,antaly,indryk)
ENDPACKAGE copy602
```

Pakkens procedurer kan udskrive kopier af et grafik-skærmbillede på en Rc602 eller en Rc605 printer.

Pakkens procedurer fungerer som beskrevet i copy603-pakken.

G.3.5 Aflæs skriverstatus (STATUS)

Pakken ser således ud:

```
PACKAGE status
   FUNC skriver(skrivernr)
   FUNC aktuel
ENDPACKAGE status
```

Pakkens funktioner kan aflæse status af skrivere. Bemærk, at funktionerne ikke fungerer i forbindelse med skrivere navngivet over lokalnet.

FUNC skriver(skrivernr)

Funktionen aflæser status af skriveren med nummer skrivernummer. Gyldige skrivernumre er for PICCOLINE 0,1,2 og for Partner 0 og 1. Returværdien angiver status. Følgende værdier er mulige:

0 skriver klar
1 ulovligt skrivernummer
2 skriver optaget af andet program
3 fælles skriver reserveret af anden PICCOLINE
4 skriver ikke klar

FUNC aktuel

Funktionen aflæser status af den aktuelle skriver. Returværdien angiver status. Følgende værdier er mulige:

0 skriver klar
1 ulovligt skrivernummer
2 skriver optaget af andet program
3 fælles skriver reserveret af anden PICCOLINE
4 skriver ikke klar

Fejlreaktioner

Ingen

G.3.6 Styringspakke (ADAM)

Pakken ser således ud:

```
PACKAGE adam
   PROC analogin(channo, gain, REF value)
   PROC analogout(value)
   PROC motorstep(motorno, steptype)
   PROC relæskift(nytilstand, relænummer)
   PROC læskontakt(kontaktno, REF value)
ENDPACKAGE adam
```

Pakkens procedurer bruges til måling og styring i forbindelse med ADAM modulet (MF916) til PICCOLINE. ADAM pakkens procedurer svarer til de RcComal80 procedurer, der er beskrevet i eksemplerne i PICCOLINE ADAM brugervejledningen (ref. 10).

PROC analogin (channo, gain, REF value)

Denne procedure måler den analoge spænding på kanalen channo, med forstærkningen gain; resultatet afleveres i variablen value. Analog måling er beskrevet i kapitel 7 i PICCOLINE ADAM brugervejledning (ref. 10).

PROC analogout (value)

Denne procedure giver med value i området 0..255 en analog spænding ud på 0..10 Volt. Analog styring er beskrevet i kapitel 7 i PICCOLINE ADAM brugervejledning.

PROC motorstep (motorno, steptype)

Denne procedure kører motoren motorno et step af typen steptype.

steptype	betydning
0	helstep frem
1	halvstep frem
2	helstep baglæns
3	halvstep baglæns

Stepmotorstyring er beskrevet i kapitel 8 i PICCOLINE ADAM brugervejledning (ref. 10).

PROC relæskift (nytilstand, relænummer)

Denne procedure får relæet relænummer til at trække, hvis nytilstand=1, og til at slippe, hvis nytilstand=0. Relæstyring er beskrevet i kapitel 4 i PICCOLINE ADAM brugervejledning.

PROC læskontakt (kontaktno, REF value)

Denne procedure returnerer stilling af kontakten kontaktno i variablen value. Der returneres værdien .1, hvis kontakten er trukket ellers 0. Digital input er beskrevet i kapitel 5 i PICCOLINE ADAM brugervejledning.

Fejlreaktioner

Ingen

G.3.7 Behandling af hardware-porte (PORTIO)

Pakken ser således ud:

```
PACKAGE portio
   PROC pin(port, REF d)
   PROC pout(port,d)
ENDPACKAGE portio
```

Pakkens procedurer kan udskrive på eller læse fra et portnummer.

PROC pin (port, REF d)

Denne procedure læser fra I/O-porten port og afleverer resultatet i parameteren d.

PROC pout (port, d)
Proceduren skriver værdien d på I/O-porten port.

Fejlreaktioner

Ingen

G.3.8 Automatisk modem-opkald (OPKALD)

Pakken ser således ud:

```
PACKAGE opkald
   FUNC kald(kaldstr$, REF fejlindex, REF timeout)
ENDPACKAGE opkald
```

FUNC kald (kaldstr$, REF fejlindex, REF timeout)

Denne funktion udfører et automatisk opkald vha. et PICCOLINE modem, MF917.

kaldstr$ skal indeholde en streng af kommandoer til modemet. Der findes følgende kommandoer:

L	Løft rør
Q	Læg rør
Kt	Vent højst t sekunder på klartone, hvis t=0 eller udelades, benyttes 6 sek.
!	"Sort knap", dvs. vælg bylinie
Tn.	Toneopkald til abonnent (n er nummeret)
St	Vent højst t sekunder på svartone, hvis t=0 eller udelades, benyttes 20 sek.
Wt	Vent t sekunder, hvis t=0 eller udelades, benyttes 1 sek.
Rt	Vent højst t sekunder på opkald udefra, hvis t=0 eller udelades, benyttes evig tid.
A	Send svartone
Mc	Sæt transmissionsmøde (c er et ciffer) c=0: (TYPE A, 300bps FD) c=1: (TYPE B, 300bps FD) c=2: (V23, 1200/75 bps) c=3: (V23, 75/1200 bps)

Eksempel
kaldstr$ = "L K T04 32 22 22. S M3" giver et automatisk opkald til Teledata.

Returværdien for funktionen angiver med et tal, hvorledes opkaldsforsøget forløb:

0	Opkald OK
1	Der er ikke installeret modem
2	Der er ikke nogen klartone
3	Klartonen er ustabil
4	Tal forventes

5	Ulovligt tegn i opkaldsnummer
6	Modem tilstandsfejl
7	Ingen svartone
8	Ustabil svartone
9	Syntaksfejl i kommandostreng

fejlindex
Hvis der under kaldet opstod en fejl, angiver fejlindex det sted i kommandostrengen, hvor fejlen blev opdaget.

timeout
Hvis der under kaldet er overskredet en tidsparameter, vil timeout indeholde værdien 1. Ellers vil timeout indeholde værdien 0.

Fejlreaktioner

Ingen

G.3.9 Strengkonvertering (STRKONV)

Pakken ser således ud

```
PACKAGE strkonv
   PROC lower(REF mystr$)
   PROC upper(REF mystr$)
ENDPACKAGE strkonv
```

Pakkens procedurer kan konvertere en streng til henholdsvis små bogstaver og store bogstaver.

PROC lower(REF mystr$)

Proceduren lower konverterer alle store bogstaver i parameteren mystr$ til små bogstaver, mens de øvrige tegn ikke ændres.

PROC upper(REF mystr$)

Proceduren upper konverterer alle små bogstaver i parameteren mystr$ til store bogstaver, mens de øvrige tegn ikke ændres.

Eksempel

```
0010 USE strkonv
0020 DIM navn$ OF 10
0030 ...
.....
.....
0120 INPUT "Hvad er svaret ": svar$
0130 lower(svar$)
0140 IF svar$ = "ja" THEN
0150    ...
.....
.....
```

Ved at bruge proceduren lower i linie 130 undgår man, at betingelsen i linie 140 bliver kompliceret, idet "ja" kan staves på 4 måder, når man kan bruge små og store bogstaver.

Fejlreaktioner

Ingen

G.3.10 Gemme og hente grafikbilleder (BILLEDE)

Pakken ser således ud

```
PACKAGE billede
   PROC gembillede(filnavn$)
   PROC hentbillede(filnavn$)
ENDPACKAGE billede
```

Pakkens procedurer kan gemme et grafik-skærmbillede på en fil og hente et grafik-skærmbillede fra en fil på disketten. Procedurerne virker både på PICCOLINE og Partner, idet der anvendes filtype PIC hvis der er en 50Hz-skærm (PICCOLINE-skærm) på, og der anvendes filtype PAR hvis der er en 60Hz-skærm (Partner-skærm). Procedurerne låser konsollen, således at det ikke er muligt at skifte konsol, mens procedurerne udføres.

PROC gembillede(filnavn$)

Proceduren gemmer det aktuelle grafik-skærmbillede på den angivne fil.

PROC hentbillede(filnavn$)

Proceduren henter grafik-skærmbillede fra den angivne fil.

Fejlreaktioner

0181 Fejl i filnavn

0182 Fejl ved åbning af filen

0183 Konsollen er ikke i forgrund

0184 Fejl ved behandling af filen

0185 Fejl ved læsning fra filen

0186 Fejl ved skrivning på filen

G.3.11 Låse konsol (KONSOL)

Pakken ser således ud

```
PACKAGE konsol
   FUNC lås
   PROC låsop
ENDPACKAGE konsol
```

Pakkens funktion bruges til at låse for konsolskift. Pakken er f.eks. anvendelig i forbindelse med pakkerne COPY603 og COPY602, så man ikke utilsigtet kommer til at skifte konsol under udskrift af et grafik-skærmbillede, og derved ødelægger udskriften.

FUNC lås

Funktionen låser konsollen. Funktionsværdien angiver
- 0 Konsollen er ikke i forgrunden og kan derfor ikke låses
- 1 Konsollen er låst

PROC låsop

Proceduren låser konsollen op, så det igen er muligt at skifte konsol. Hvis konsollen ikke er låst, sker der intet.

Fejlreaktioner

Ingen.

H. PAKKEHJÆLPEFILER

Dette afsnit indeholder udskrifter af de to hjælpefiler POLYPAS.PAS og ASSEMBL.A86, der anvendes, når man vil lave pakker i PolyPascal eller assembler. De anvendes i afsnit 12.3 og 12.4.

H.1 POLYPAS.PAS

POLYPAS.PAS anvendes, når man vil lave PolyPascal-pakker, som beskrevet i afsnit 12.3.

```
CONST
    versno : INTEGER    = $8324;
    ppasmodif           = 41;
    valuepar            = 0;
    refpar              = 1;
    realpar             = 0;
    stringpar           = 2;
    proc                = 5;
    realfunc            = 9;
    stringfunc          = 10;
    ppaserr             = 190;
    konverr             = 191;

TYPE
    navne               = STRINGÆ16Å;

VAR
    (* de fire reserver må ikke slettes *)
    reserver,
    reserver1,
    reserver2,
    reserver3           : INTEGER;
    hovedfil            : FILE OF BYTE;
    helptal             : ARRAY Æ1..8Å OF BYTE;


(* ------------------------------ *)
(* RUTINER TIL UDSKRIVNING AF HOVED *)

PROCEDURE openfile(filnavn : navne);
BEGIN
```

```
    assign(hovedfil, filnavn);
    rewrite(hovedfil);
END;

PROCEDURE closefile;
BEGIN
    close(hovedfil);
END;

PROCEDURE skriv_byte(tal : byte);
BEGIN
    write(hovedfil,tal);
END;

PROCEDURE skriv_navn(varnavn : navne);
VAR
    i : integer;
BEGIN
    skriv_byte(len(varnavn));
    FOR i:=1 TO len(varnavn) DO
      skriv_byte(ord(varnavnÆiÅ));
END;

PROCEDURE skriv_offset(offset : integer);
BEGIN
    skriv_byte(lo(offset));
    skriv_byte(hi(offset));
END;

PROCEDURE skriv_packtype;
BEGIN
    skriv_offset(ppasmodif);
END;

PROCEDURE skriv_versionsno;
BEGIN
    skriv_offset(versno);
END;

PROCEDURE skriv_proc;
BEGIN
    skriv_byte(proc);
END;

PROCEDURE skriv_realfunc;
BEGIN
    skriv_byte(realfunc);
END;
```

```
PROCEDURE skriv_stringfunc;
BEGIN
    skriv_byte(stringfunc);
END;

PROCEDURE skriv_typeogdim(partype, pardim : byte);
BEGIN
    skriv_byte(partype);
    skriv_byte(pardim);
END;

PROCEDURE skriv_reserver;
BEGIN
    skriv_offset(ofs(reserver));
END;


(* ------------------------------ *)
(* RUTINER TIL PARAMETER BEHANDLING *)

TYPE
    stringmax           = STRINGÆ255Å;


PROCEDURE comalerror(errorkode : integer);
BEGIN CODE
    $8B,$76,$04,            (* MOV SI,4ÆBPÅ  ; ERRORKODE*)
    $BB,$20,$00,            (* MOV BX,32                *)
    $8B,$2E,reserver3,      (* MOV BP,RESERVER3         *)
    $1E,                    (* PUSH DS                  *)
    $8E,$5E,$04,            (* MOV DS,4ÆBPÅ             *)
    $FF,$5E,$00,            (* CALLF DWORD PTR ÆBPÅ     *)
    $1F,                    (* POP DS                   *)
    $8B,$2E,reserver3,      (* MOV BP,RESERVER3         *)
    $83,$ED,$06,            (* SUB BP,6                 *)
    $8B,$E5,                (* MOV SP,BP                *)
    $CB;                    (* RETF                     *)
END;

(* UNDERRUTINER *)

PROCEDURE henttal(segadr, ofsadr : integer; VAR tal : real);
VAR
    i,
    expo : integer;
    help : real;

    FUNCTION konv(segadr, ofsadr, no : integer): integer;
```

```
    VAR
      j,
      k    : integer;
    BEGIN
      j:=no div 2;
      k:=mem[segadr:ofsadr+j];
      IF no mod 2 = 0 THEN konv:=k div 16
      ELSE konv:=k mod 16;
    END; (* konv *)

BEGIN
    help:=0.0;
    FOR i:=1 TO 13 DO
    BEGIN
      help:=help*10+konv(segadr, ofsadr, i);
    END;
    IF konv(segadr,ofsadr,0) <> 0 THEN help:=-help;
    IF help <> 0 THEN
    BEGIN
      help:=help*pwrten(-13);
      expo:=mem[segadr:ofsadr+7]-128;
      IF (expo>63) OR (expo<-63) THEN
      BEGIN
        comalerror(konverr);
      END;
      help:=help*pwrten(expo);
    END;
    tal:=help;
END; (* henttal *)


PROCEDURE gemtal(segadr, ofsadr : integer; tal : real);
VAR
    expo,
    k,
    ciffer      : integer;
    mystr       : string[18];

    PROCEDURE nibble(segadr, ofsadr : integer; no, value : byte);
    VAR
      j,
      k         : integer;
    BEGIN
      j:=no div 2;
      k:=mem[segadr:ofsadr+j];
      IF no mod 2 = 0 THEN mem[segadr:ofsadr+j]:=value*16+k mod 16
      ELSE mem[segadr:ofsadr+j]:=(k div 16)*16+value;
    END; (* nibble *)
```

```
    FUNCTION somtal(ch :char): integer;
    BEGIN
      somtal:=ord(ch)-ord('0');
    END;

BEGIN
    str(tal,mystr);
    IF mystrÆ2Å='-' THEN nibble(segadr,ofsadr,0,8)
    ELSE nibble(segadr,ofsadr,0,0);
    k:=1;
    WHILE (k<=13) DO
    BEGIN
      IF k>=12 THEN ciffer:=0
      ELSE
        IF k=1 THEN ciffer:=somtal(mystrÆ3Å)
        ELSE ciffer:=somtal(mystrÆk+3Å);
      nibble(segadr, ofsadr, k, ciffer);
      k:=k+1;
    END;
    expo:=somtal(mystrÆ17Å)*10+somtal(mystrÆ18Å);
    IF mystrÆ16Å='-' THEN expo:=-expo;
    expo:=129+expo;
    memÆsegadr:ofsadr+7Å:=expo;
END; (* gemtal *)

PROCEDURE hentstring(segadr, ofsadr : integer; VAR mystr : stringmax);
VAR
    i,
    antal,
    myseg,
    myofs       : integer;

BEGIN
    antal:=memwÆsegadr:ofsadr+2Å;
    IF antal>255 THEN comalerror(konverr);
    myseg:=seg(mystr);
    myofs:=ofs(mystr);
    FOR i:=1 TO antal DO
    BEGIN
      memÆmyseg:myofs+iÅ:=memÆsegadr:ofsadr+i+3Å;
    END;
    memÆmyseg:myofsÅ:=antal;
END; (* hentstring *)

PROCEDURE gemstring(segadr, ofsadr : integer; mystr : stringmax);
VAR
    i,
```

```
    antal,
    myseg,
    myofs       : integer;

BEGIN
    antal:=len(mystr);
    IF antal > memwÆsegadr:ofsadrÅ THEN comalerror(konverr);
    myseg:=seg(mystr);
    myofs:=ofs(mystr);
    FOR i:=1 TO antal DO
    BEGIN
      memÆsegadr:ofsadr+i+3Å:=memÆmyseg:myofs+iÅ;
    END;
    memwÆsegadr:ofsadr+2Å:=antal;
END; (* gemstring *)


(* DE VÆSENTLIGE RUTINER *)

PROCEDURE getrealpar(no : byte; VAR tal : real);
VAR
    stackadr : integer;
BEGIN
    stackadr:=memwÆseg(reserver):ofs(reserver)+6Å;
    henttal(memwÆsseg:stackadr+4Å,memwÆsseg:stackadr+4+no*2Å,tal);
END; (* getrealpar *)

PROCEDURE putrealpar(no : byte; tal : real);
VAR
    stackadr : integer;
BEGIN
    stackadr:=memwÆseg(reserver):ofs(reserver)+6Å;
    gemtal(memwÆsseg:stackadr+4Å,memwÆsseg:stackadr+4+no*2Å,tal);
END; (* putrealpar *)

PROCEDURE getstringpar(no : byte; VAR mystr : stringmax);
VAR
    stackadr : integer;
BEGIN
    stackadr:=memwÆseg(reserver):ofs(reserver)+6Å;
    hentstring(memwÆsseg:stackadr+4Å,memwÆsseg:stackadr+4+no*2Å,mystr);
END; (* getstringpar *)

PROCEDURE putstringpar(no : byte; mystr : stringmax);
VAR
    stackadr : integer;
BEGIN
    stackadr:=memwÆseg(reserver):ofs(reserver)+6Å;
```

```
    gemstring(memwÆsseg:stackadr+4Å,memwÆsseg:stackadr+4+no*2Å,mystr);
END; (* putstringpar *)


(* CODE RUTINER *)

PROCEDURE returnfar;
BEGIN CODE
    $8B,$2E,reserver3,          (* MOV BP,RESERVER3          *)
    $83,$ED,$06,                (* SUB BP,6                  *)
    $8B,$E5,                    (* MOV SP,BP                 *)
    $CB;                        (* RETF                      *)
END;

PROCEDURE comalprgfunckey;
BEGIN CODE
    $55,                        (* PUSH BP                   *)
    $BB,$1F,$00,                (* MOV BX,31                 *)
    $8B,$2E,reserver3,          (* MOV BP,RESERVER3          *)
    $1E,                        (* PUSH DS                   *)
    $8E,$5E,$04,                (* MOV DS,4ÆBPÅ              *)
    $FF,$5E,$00,                (* CALLF DWORD PTR ÆBPÅ      *)
    $1F,                        (* POP DS                    *)
    $5D;                        (* POP BP                    *)
END;

PROCEDURE returnresult(resseg, resofs, mylen : integer);
BEGIN CODE
    $8B,$46,$08,                (* MOV AX,8ÆBPÅ  ; RESSEG    *)
    $8B,$4E,$06,                (* MOV CX,6ÆBPÅ  ; RESOFS    *)
    $8B,$56,$04,                (* MOV DX,4ÆBPÅ  ; MYLEN     *)
    $8B,$2E,reserver3,          (* MOV BP,RESERVER3          *)
    $83,$ED,$06,                (* SUB BP,6                  *)
    $8B,$E5,                    (* MOV SP,BP                 *)
    $CB;                        (* RETF                      *)
END;

PROCEDURE return_realresult(tal : real);
BEGIN
    gemtal(seg(helptal),ofs(helptal),tal);
    returnresult(seg(helptal),ofs(helptal),0);
END;

PROCEDURE return_stringresult(mystr : stringmax);
VAR
    leng    : integer;
BEGIN
    leng:=len(mystr);
```

```
    returnresult(seg(mystr),ofs(mystr)+1,leng);
END;


(* BEHANDLING AF POLYPASCAL-FEJL *)

PROCEDURE error(errno, errofs : integer);

    PROCEDURE hexwrite(no, antalfelter : integer);

      PROCEDURE hexciffer(cif : integer);
      BEGIN
        IF cif<=9 THEN write(cif:1)
        ELSE
        BEGIN
          write(chr(ord(cif-10)+ord('A')));
        END;
      END;

    BEGIN
      IF antalfelter = 1 THEN
      BEGIN
        hexciffer(no);
      END
      ELSE
      BEGIN
        hexwrite(no div 16, antalfelter-1);
        hexciffer(no mod 16);
      END;
    END;

BEGIN
    write('PolyPascal ');
    CASE hi(errno) OF
      0 : write('programafbrydelse (Ctrl C) ');
      1 : BEGIN
            write('I/0 fejl ');
            hexwrite(lo(errno),2);
          END;
      2 : BEGIN
            write('kørselsfejl ');
            hexwrite(lo(errno),2);
          END;
    END;
    write(' ved PC=');
    hexwrite(errofs,4);
    comalerror(ppaserr);
END;
```

```
PROCEDURE init_errorhandler;
BEGIN
    ehofs:=ofs(error);
END;
```

H.2 ASSEMBL.A86

ASSEMBL.A86 anvendes, når man vil lave assemblerpakker, som beskrevet i afsnit 12.4.

```
VERSNO      EQU     8324h
ASSEMBLER   EQU     28h
VALUE       EQU     0
REF         EQU     1
REAL        EQU     0
STRING      EQU     2
PROC        EQU     5
REALFUNC    EQU     9
STRINGFUNC  EQU     10
ASSERR      EQU     180

CSEG $

    ; FUNCTION get_real_par(no : byte): integer;

    ; (* henter parameter no fra COMAL og konverterer tallet og
    ;    og lægger det i BX *)

    ;           CALL                RETURN
    ; CL        no
    ; BX                            get_real_par

REALPAR:
    MOV     BP,RESERVER+6       ;
    XOR     CH,CH               ;
    SHL     CX,1                ;
    ADD     BP,CX               ;
    MOV     SI,4ÆBPÅ            ;   ptr := adresse_par(no);
    MOV     BX,1                ;
    CALL    COMALRUT            ;   comal_fix(ptr);
    RET                         ;


    ; PROCEDURE get_string_par(no : byte, adr : integer);
```

```
    ; (* henter parameter no fra COMAL og lægger den i adressen
    ;    DI udpeger og fremefter *)

    ;          CALL                RETURN
    ; CL       no
    ; DI       adr

STRPAR:
    MOV        BP,RESERVER+6       ;
    PUSH       DS                  ;
    POP        ES                  ;   extra_segment := pakkens
data_segment;
    PUSH       DS                  ;   gem pakkens data_segment;
    MOV        DS,4ÆBPÅ            ;   data_segment :=
                                   ;         COMAL's data_segment;
    XOR        CH,CH               ;
    SHL        CX,1                ;
    ADD        BP,CX               ;
    MOV        SI,4ÆBPÅ            ;   ptr := adresse_par(no);
    INC        SI                  ;
    INC        SI                  ;   skip_max_længde;
    LODS       AX                  ;
    MOV        CX,AX               ;   hent_nuværende_længde;
    CLD                            ;
    REP        MOVSB               ;   flyt strengen;
    POP        DS                  ;   data_segment :=
                                   ;         pakkens data_segment;
    RET                            ;


    ; PROCEDURE put_real_resultat(resultat : integer,
    ;                             REF resultat_segm, resultat_offset : integer);

    ; (* konverterer resultatet til COMAL-format  og returnerer værdien *)

    ;          CALL                RETURN
    ; BX       resultat
    ; AX                           resultat_segm
    ; CX                           resultat_offset

REALRES:MOV    SI,BX               ;
    MOV        BX,0                ;
    CALL       COMALRUT            ;   comal_float(resultat);
    RET                            ;
```

```
    ; PROCEDURE put_real_parameter(resultat : integer, no : byte);

    ; (* konverterer resultatet til COMAL-format  og lægger resultatet
    ;    på adressen for parameter no *)

    ;          CALL                RETURN
    ; BX       resultat
    ; CL       no

PUTREALPAR:
    PUSH     CX                  ;
    CALL     REALRES             ;   realres(resultat,res_segm,res_offset);
    MOV      SI,CX               ;   from := res_offset;
    MOV      BP,RESERVER+6       ;(* res_segm = COMAL's data_segment *)
    POP      CX                  ;
    XOR      CH,CH               ;
    SHL      CX,1                ;
    ADD      BP,CX               ;
    MOV      DI,4ÆBPÅ            ;   to := adresse_par(no);
    PUSH     DS                  ;   gem pakkens data_segment;
    MOV      ES,AX               ;   extra_segment :=
                                 ;         COMAL's data_segment;
    MOV      DS,AX               ;   data_segment :=
                                 ;         COMAL's data_segment;
    MOV      CX,8                ;
    REP      MOVSB               ;   movebytes(from,to,8);
    POP      DS                  ;   hent pakkens data_segment;
    RET                          ;


    ; PROCEDURE finito;
    ; (* return far *)

FINITO:
    MOV      BP,RESERVER+6       ;
    SUB      BP,6                ;
    MOV      SP,BP               ;   return_far;
    RETF                         ;


    ; PROCEDURE test_error(error_kode : integer);

    ; (* tester om error_kode er forskellig fra nul og sætter
    ;    i så fald fejlkoden i COMAL *)
```

```
    ;           CALL                RETURN
    ; AX        error_kode

    ; PROCEDURE error(error_kode : integer);

    ; (* sætter fejlkoden i COMAL *)

    ;           CALL                RETURN
    ; AX        error_kode

TSTERR:
    OR          AX,AX               ;
    JNZ         ERROR               ;   IF error_kode <> 0 THEN
    RET                             ;           comal_set_error(error_kode);

ERROR:      MOV     SI,AX           ;
    MOV     BX,32                   ;
    CALL    COMALRUT                ;   comal_set_error(error_kode);
    RET                             ;

    ; PROCEDURE comal_rutine(rutine_no, arg1, arg2 : integer,
    ;                                 REF res_offset, res_segment : integer);

    ; (* kalder en af de COMAL-rutiner, der er til rådighed for pakken *)

    ;           CALL                RETURN
    ; BX        rutine_no
    ; SI        arg1
    ; DI        arg2
    ; CX                            res_offset
    ; AX                            res_segment

COMALRUT:
    MOV         BP,RESERVER+6       ;
    PUSH        DS                  ;   gem pakkens data_segment;
    MOV         DS,4ÆBPÅ            ;   data_segment :=
                                    ;           COMAL's data_segment;
    CALLF       DWORD PTR ÆBPÅ      ;   kald COMAL-rutinen;
    PUSH        SI                  ;   gem resultat_offset;
    PUSH        DS                  ;   gem resultat_segment;
    CALL        TSTERR              ;   test om der er fejl
                                    ;           under COMAL-rutinen;
    POP         AX                  ;   resultat_segment;
```

```
    POP     CX              ;   resultat_offset;
    POP     DS              ;   data_segment :=
                            ;           pakkens data_segment;
    RET                     ;

DSEG $

    ORG     100h

    RESERVER RW     4

CSEG $
```

Stikordsregister

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Y

Z

LÆSERBEMÆRKNINGER

Titel: RcComa180, Brugervejledning RCSL Nr.: 99000927

A/S Regnecentralen af 1979 bestræber sig på at forbedre kvalitet og brugbarhed af sine publikationer. For at opnå dette ønskes læserens kritiske vurdering af denne publikation.

Kommenter venligst manualens fuldstændighed, nøjagtighed, disposition, anvendelighed og læsbarhed:

Angiv fundne fejl (reference til sidenummer):

Hvordan kan manualen forbedres:

Andre kommentarer:

Navn: Stilling:

Firma:

Adresse:

Dato:

På forhånd tak!

PN: 99200248

..... Fold her .....

..... Riv ikke - Fold her og hæft .....

Frankeres
som
brev

§ REGNECENTRALEN
af 1979
Informationsafdelingen
Lautrupbjerg 1
2750 Ballerup