# PICCOLINE

*RcComal 80*

*Brugervejledning*

# PICCOLINE

*RcComal 80*
*Brugervejledning*

# RcComal80
# Brugervejledning

Introduktion
Opstart og indtastning
af programmer
Symboler og reserverede
ord
Tal og tekst

Kontrolstrukturer
Indlæsning og udskrivning
Tal - og tekst - tabeller
Procedurer og funktioner
Kommandoer
Grafik

RcComal80 nøgleord

Referencer
Fejlmeddelelser
Ydre enheder
Installering af RcComal80
ASCII Tegnsættet

Store programeksempler
Automatisk opstart af
programmer

PN: 99109997

Nøgleord: RC750, RcComa180, Partner, Piccoline, Piccolo.

Resumé: Denne vejledning beskriver RcComa180 på Partner, Piccoline og Piccolo mikrodatamaterne.

Udgave: Januar 1985

CCP/M-86 og CP/M er registrerede varemærker fra Digital Research.

**Copyright © 1985 A/S Regnecentralen af 1979**
**RC Computer A/S**
**Udgivet af A/S Regnecentralen af 1979, København**

**Brugere af denne manual gøres opmærksom på, at specifikationerne heri uden forudgående varsel kan ændres af RC. RC er ikke ansvarlig for typografiske fejl eller regnefejl, som kan forekomme i denne manual, og er ikke ansvarlig for skader forårsaget af benyttelsen af dette dokument.**

# RcComal80

## Brugervejledning

**Indholdsfortegnelse**


1. Introduktion.....11

2. Opstart og indtastning af programmer.....13
   2.1 Tastaturet.....14
      2.1.1 Partner- og Piccoline-tastaturet.....14
      2.1.2 Piccolo-tastaturet.....16
   2.2 Indtastning af programmer.....18
      2.2.1 Kommandoer til programindtastning.....20
         2.2.1.1 NEW.....20
         2.2.1.2 AUTO.....20
         2.2.1.3 RENUMBER.....21
         2.2.1.4 DEL.....21
      2.2.2 Kommandoer til programudførelse (RUN og CON).22

3. Symboler og reserverede ord.....23
   3.1 Identifikatorer.....23
   3.2 Reserverede ord.....23

4. Tal og tekst.....25
   4.1 Numeriske variable.....25
      4.1.1 Tildeling.....25
   4.2 Beregningsudtryk.....26
      4.2.1 Regler for de almindelige regneoperationer...28
      4.2.2 Beregningspræcision.....28
   4.3 Strengvariable.....29
      4.3.1 Erklæring af strengvariable.....29
   4.4 Strengbehandling.....29
      4.4.1 Strengkonstanter.....29
      4.4.2 LEN-funktionen.....30
      4.4.3 Delstrenge.....31
      4.4.4 Sammensætning af strenge.....31
   4.5 Logiske udtryk.....32
      4.5.1 IN-operatoren.....34
   4.6 Numeriske udtryk.....34
      4.6.1 Aritmetiske-, sammenlignings- og logiske operatorer.....35

5. Kontrolstrukturer.....37
   5.1 Betingede sætninger.....37
      5.1.1 IF-sætninger.....37
      5.1.2 Multiforgreninger (CASE).....38
   5.2 Løkkestrukturer.....41
      5.2.1 Tælleslø jfer (FOR-NEXT).....41
      5.2.2 REPEAT-konstruktionen.....43
      5.2.3 WHILE-konstruktionen.....44

6. Indlæsning og udskrivning.....45

6.1 INPUT-sætningen....45
6.2 PRINT-sætningen....46
6.3 Datastrømme....47
6.3.1 Filer og ydre enheder....47
6.3.2 Skrivning og læsning af programfiler....49
6.3.3 Skrivning og læsning af datafiler....50
6.3.3.1 Sekventielle filer....51
6.3.3.2 Filer med direkte tilgang....53
6.3.3.3 Ydre enheder....55
6.3.4 Fjernelse af filer....56
6.3.5 Omdøbning af filer....56
6.4 Oversigt over I/O sætninger....56

7. Tal- og tekst-tabeller....57
7.1 Taltabeller....57
7.1.1 Taltabelkomponenter....57
7.1.2 Erklæring af taltabeller....57
7.2 Teksttabeller....58
7.2.1 Teksttabelkomponenter....58
7.2.2 Erklæring af teksttabeller....59

8. Procedurer og funktioner....61
8.1 Simple procedurer....62
8.2 Parameteroverførsel....63
8.3 REF angivelse....64
8.4 Lukkede procedurer....66
8.5 Externe procedurer....69
8.6 Handler....70
8.7 Funktioner....72

9. Kommandoer....75
9.1 Kommandoer til programindtastning....75
9.2 Sætninger udført som kommandoer....76
9.2.1 Datamaten som bordkalkulator....77
9.2.2 Fejlfinding i programmer....77
9.2.3 Datastrømsindlæsning/udskrivning....77
9.3 Diskkommandoer (MOUNT og DIR)....78
9.4 Datastrømskommandoer....78
9.5 BYE....80

10. Grafik....81
10.1 Opstart af grafik....81
10.2 Grafik i RcComa180....82
10.2.1 Valg af grafisk enhed....82
10.2.2 Definition af det grafiske vindue....82
10.2.3 Tegning af streger....83
10.2.4 Afslutning af det aktuelle billede....84

11. RcComa180 nøgleord....85

11.1 ABS....87
11.2 AND....88
11.3 AT....89
11.4 ATN....91
11.5 AUTO....93
11.6 BYE....95
11.7 CASE-WHEN-ENDCASE....96
11.8 CHAIN....98
11.9 CHR$....99
11.10 CIRCLE....100
11.11 CLEAR....102
11.12 CLOSE....103
11.13 CLOSE GRAPHICS....104
11.14 CON....105
11.15 CONTINUE....106
11.16 COPY....107
11.17 COS....108
11.18 CREATE....109
11.19 DATA....111
11.20 DATE$....113
11.21 DEL....114
11.22 DELETE....115
11.23 DIM....117
11.24 DIR....119
11.25 DISABLE....121
11.26 DIV....122
11.27 DRAW....123
11.28 DRAWTO....125
11.29 EDIT....126
11.30 ENABLE....127
11.31 END....128
11.32 ENTER....129
11.33 EOD....130
11.34 EOF....131
11.35 ERR....133
11.36 ERRTEXT$....134
11.37 Etikette....135
11.38 EXEC....136
11.39 EXP....137
11.40 FALSE....138
11.41 FOR....139
11.42 FOR-NEXT....141
11.43 FUNC-ENDFUNC....142
11.44 FUNC-EXTERNAL....144
11.45 GET$....146
11.46 GLOBAL....148
11.47 GOTO....149
11.48 GPARM....150
11.49 GSX....151

11.50 IF-THEN.....160
11.51 IF-THEN-ENDIF.....161
11.52 IF-THEN-ELSE-ENDIF.....162
11.53 IMPORT.....164
11.54 IN.....165
11.55 INPUT.....166
11.56 INPUT FILE.....167
11.57 INT.....169
11.58 KEY$.....170
11.59 LEN.....171
11.60 LIST.....172
11.61 LOAD.....174
11.62 LOCATE.....175
11.63 LOG.....177
11.64 MARGIN.....178
11.65 MOD.....179
11.66 MOUNT.....180
11.67 MOVE.....181
11.68 MOVETO.....182
11.69 NEW.....183
11.70 NOT.....184
11.71 OPEN.....185
11.72 OPEN GRAPHICS.....187
11.73 OR.....188
11.74 ORD.....189
11.75 OUT.....190
11.76 PALETTE.....191
11.77 PENCOLOR.....192
11.78 PI.....193
11.79 PREFIX.....194
11.80 PRINT.....195
11.81 PRINT FILE.....196
11.82 PRINT FILE USING.....197
11.83 PRINT USING.....198
11.84 PROC-ENDPROC.....200
11.85 PROC-EXTERNAL.....202
11.86 PROC-HANDLER.....203
11.87 RANDOMIZE.....204
11.88 READ.....206
11.89 READ FILE.....207
11.90 RENAME.....208
11.91 RENUMBER.....209
11.92 REPEAT - UNTIL.....211
11.93 RESTORE.....216
11.94 RETRY.....218
11.95 RETURN.....219
11.96 RND.....220
11.97 RUN.....222
11.98 SAVE.....223

11.99 SCREEN$....224
11.100 SELECT OUTPUT....225
11.101 SGN....226
11.102 SIN....227
11.103 SIZE....229
11.104 SQR....230
11.105 STOP....231
11.106 STR$....232
11.107 SYS....235
11.108 TAB....236
11.109 TAN....237
11.110 TEXT....238
11.111 Tildeling....239
11.112 TIME$....240
11.113 TRUE....241
11.114 VAL....242
11.115 WHILE....243
11.116 WHILE-ENDWHILE....244
11.117 WINDOW....246
11.118 WRITE FILE....248
11.119 ZONE....249

A. Referencer....251

B. Fejlmeddelelser....253
B.1 Fejl fundet under programindtastning....254
B.2 Kommando- eller udførelsesfejl....257
B.3 I/O-Fejlmeddelelser....260

C. Ydre enheder....263
C.1 Partner og Piccoline skærmstyring....263
C.1.1 Ændring af tegns udseende på skærmen....266
C.1.2 Farver....266
C.1.3 Tegnsæt....267
C.2 Partner og Piccoline tastatur....267
C.2.1 Funktionstaster....267
C.3 Lydkreds på Partner og Piccoline....269
C.4 IEEE-kort på Piccoline....270
C.5 Piccoloens ydre enheder....271
C.5.1 Piccolo Skærmstyring....271
C.5.1.1 Blink og invers skrift....272
C.5.1.2 Semigrafisk tegnsæt....273
C.5.1.3 Skærm - tegngenerering og konvertering....273
C.5.2 Tastatur - tegngenerering og konvertering...278
C.6 Piccolo HW-porte....278
C.7 Styring af skrivere....279
C.7.1 RC861-skriveren....279
C.7.2 RC862,RC867 skriveren....280

D. Installering af RcComa180.....281
D.1 Partner og Piccoline installation.....281
D.2 Piccolo installation.....281
D.2.1 COMALCNV.....282

E. ASCII tegnsættet.....285

F. Store programeksempler.....287
F.1 Enarmet tyveknægt.....287
F.2 Tænker du på et dyr ?.....293
F.3 Tænker du på et dyr ? (filudgave).....295
F.4 Data-indtastning.....297
F.5 Sortering (PROC-EXTERNAL).....299
F.6 Semigrafik.....301
F.7 Liv.....303
F.8 Tænk på et tal !.....305
F.9 Eksempel fra procedureafsnittet.....306
F.10 Elektronisk orgel.....310

G. Automatisk opstart af programmer.....313

Stikordsregister.....315

# 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 A/S Regnecentralen 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 Piccolo, Rc Piccoline samt 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 2-10, 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 11, der er et referenceafsnit, hvori alle RcComal80-nøgleord er beskrevet præcist.

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

Den uerfarne programmør bør starte med at læse afsnit 2-10, hvorimod programmøren med COMAL-erfaring kan gå direkte til afsnit 11.

Manualen indeholder en række programmer som eksempler. Disse er alle indtastet og afprøvet på Piccolo, 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 small 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 RcComa180 skal man installere RcComa180-systemet på ens systemdisk. Dette gøres som beskrevet i appendix D. 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 RcComa180. Dette gøres enten

- fra menu systemet eller
- med TMP kommandoen

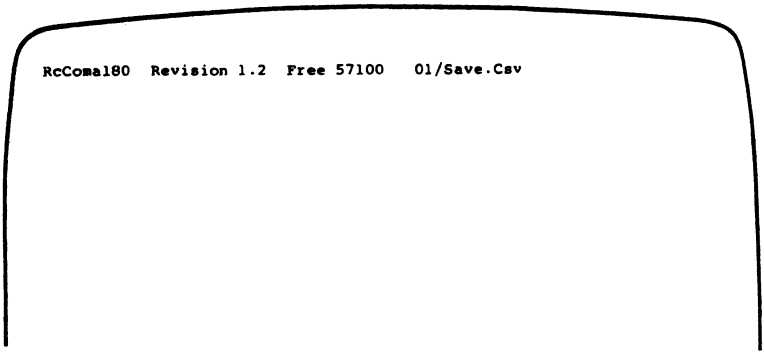
  GRAPHICS<retur>

Hvis man ønsker at benytte grafik-kommandoer på en Piccolo med indbygget grafikkort, skal man ikke foretage sig noget specielt.

Herefter kan RcComa180 startes op. Dette gøres med kommanden:

COMAL80<retur>

RcComa180 startes nu op, og f.eks. følgende skærmbillede fremkommer :



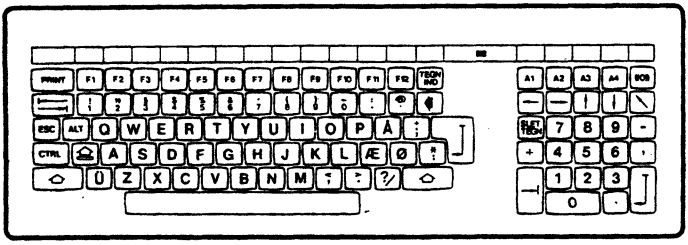

Cursoren (den blinkende firkant eller understregning) fremkommer på skærmen, som tegn på, at systemet er klar til indtastning.

Der kan herefter indtastes program-sætninger eller kommandoer.

## 2.1 Tastaturet

### 2.1.1 Partner- og Piccoline-tastaturet

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




Tasterne omfatter en gruppe lyse og en gruppe 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 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 på denne måde. 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 på en tilsluttet skriver af skærmbilledet. Benyttes altid sammen med CRTL-tasten (CRTL+PRINT). Denne facilitet kan <u>ikke</u> anvendes under RcComa180.

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/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

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

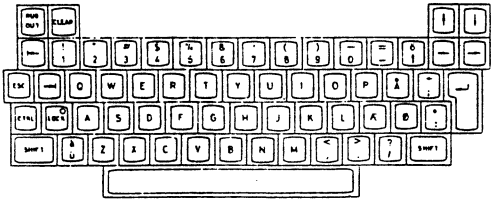
Funktionstasterne vil virke som beskrevet som standard, med mindre tasternes funktion bliver ændret af brugeren (se C.2.1).

RcComa180 på Partner og Piccoline er skærmorienteret, dvs. at man kan benytte de fire cursorpile til at "hoppe" rundt i skærmbilledet, hvorved man kan "genbruge" linier, der står på skærmen. Sålænge en linie står et sted på skærmen, kan man indtaste den til systemet ved hjælp af pilene og Retur-tasten.

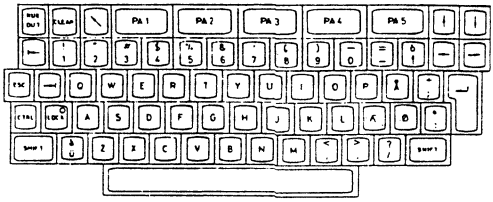
Alle taster pånær SHIFT, LOCK, CRTL og ALT repeterer, hvis de holdes nedtrykket i mere end 0,7 sek. Repetitionshastigheden vil langsomt stige, således at den maksimale repetitionshastighed opnås efter ca. 20 repetitioner.

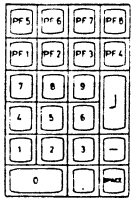
### 2.1.2 Piccolo-tastaturet

På Piccolo eksisterer to typer tastaturer, det almindelige (RC721):



og det udvidede (RC722):





Tastaturerne indeholder de almindelige taster, man kender fra en skrivemaskine, plus følgende:

| | |
|---|---|
| RUB OUT | Slet sidst indtastede tegn på linie |
| CLEAR | Slet skærmen og stil cursoren i øverste vensre hjørne. |
| ↑ | Flyt cursor en linie op |
| ↓ | Flyt cursor en linie ned |
| → | Flyt cursor en position til højre |
| ← | Flyt cursor en position til venstre |
| ↵ | Retur, ny linie. Tasten bruges altid til at afslutte sætninger eller kommandoer til systemet. |
| CTRL | Control-tasten. Trykkes ned sammen med et bogstav for at sende en specialkode til systemet. |
| ESC | Afbryder programudførelse |
| ⊢⊣ | Indsæt tegn i linie |
| ⊣⊢ | Slet tegn i linie |

Følgende taster eksisterer kun på RC722:

| | |
|---|---|
| \ | Flyt cursor til øverste venstre hjørne. |

| | |
|---|---|
| PA1-PA5, PF1-PF8 | Brugerdefinerbare specialtaster. Sender special-koder (se appendix C). |
| SPACE | Mellemrumstast |

**Bemærk at på tastaturet til RC703 er nogle af de almindelige taster ikke de samme som på RC702.**

RcComa180 på Piccolo er <u>skærmorienteret</u>, dvs. at man kan benytte de fire cursorpile til at "hoppe" rundt i skærmbilledet, hvorved man kan "genbruge" linier, der står på skærmen. Sålænge en linie står et sted på skærmen, kan man indtaste den til systemet ved hjælp af pilene og Retur-tasten.

Udover de beskrevne specialtaster, er følgende control-koder nyttige:

| | |
|---|---|
| CTRL+ \| | Slet resten af linien |
| CTRL+ RUB OUT | Slet resten af skærmen |

CTRL+... betyder, at CTRL tasten skal holdes nedtrykket mens den anden tast aktiveres.

Holdes en tast nedtrykket i mere end 0.7 sek., repeteres tegnet med 0.1 sek. intervaller, indtil tasten slippes igen.

## 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 RcComa180-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 selve kommentaren.

**Eksempel**

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

Linien skal afsluttes med en Retur.

Når linien er afsluttet, vil RcComa180-systemet undersøge, om det er en korrekt RcComa180-sætning (man siger, at systemet syntaksanalyserer linien). Er linien korrekt, lagres den i program-lageret.

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.

**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 RcComa180 systemet.
Når man er færdig med indtastningen, kan man udføre programmet ved at skrive

RUN

efterfulgt af tryk på Retur.

**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 CLEAR (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 "genbruge" linien.

### 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.

### 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 blive 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.

### 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, er kommandoen

DEL 70

Ønsker man at fjerne linierne fra linie 0110 til linie 0270, er kommandoen

DEL 110,270

Ønsker man at fjerne linie 1000 og resten af programmet, er kommandoen

DEL 1000,

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

DEL ,250

### 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 og genstarte programmet med kommandoen

CON

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

SELECT OUTPUT "printer"
CON

# 3. Symboler og reserverede ord

## 3.1 Identifikatorer

Identifikatorer benyttes til at navngive forskellige størrelser i et RcComa180-program. En identifikator består af et bogstav efterfulgt af op til 15 bogstaver og cifre eller tegnet understregning ( _ ), altså et navn på maximalt 16 tegn.

Bemærk der skelnes ikke mellem store og små bogstaver i identifikatornavne. RcComa180-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

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

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

## 3.2 Reserverede ord

Følgende ord er reserverede i RcComa180. 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
DIV
DO
DRAW
DRAWTO
DUMP

EDIT
ELSE
ENABLE
END

ENDCASE
ENDFUNC
ENDIF
ENDPROC
ENDWHILE
ENTER
EOD
EOF
ERR
ERRTEXT
EXEC
EXP
EXTERNAL

FALSE
FILE
FOR
FUNC

GET
GLOBAL
GOTO
GPARM
GRAPHICS
GSX

HANDLER

IF
IMPORT
IN
INPUT
INT

KEY

LEN
LIST
LOAD

LOCATE
LOG

MARGIN
MOD
MOUNT
MOVE
MOVETO

NEW
NEXT
NOT

OF
OPEN
OR
ORD
OTHERWISE
OUT
OUTPUT

PALETTE
PENCOLOR
PI
PREFIX
PRINT
PROC

RANDOM
RANDOMIZE
READ
REF
RENAME
RENUMBER
REPEAT
RESTORE
RETRY
RETURN
RND

RUN

SAVE
SCREEN
SELECT
SGN
SIN
SIZE
SQR
STEP
STOP
STR
SYS

TAB
TAN
TEXT
TIME
THEN
TO
TRUE

UNTIL
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 RcComa180. 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 RcComa180 skelner man registrene fra hinanden ved hjælp af et navn. Et tal-register kaldes i RcComa180 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 | |

#### 4.1.1 Tildeling

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

```
10 pris := 10.85
```

Bemærk, at man i RcComa180 benytter 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 RcComa180 angives en tierpotens ved bogstavet E efterfulgt af potensen.

**Eksempel 1**

| **Tal** | **RcComa180 notation** |
|---|---|
| $1.4\ 10^{100}$ | 1.4E+100 |
| -0.0000000001234 | -1.234E-10 |
| $200\ 10^{15}$ | 2E17 |

På 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 <u>matematisk</u> lighedstegn, da f.eks. følgende sætning er lovlig:

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

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

## 4.2 Beregningsudtryk

Man kan f.eks. foretage simple udregninger i RcComa180 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). RcComa180 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 (↑ på Piccolo) |
| 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 RcComa180:

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

Endelig kan man benytte funktioner i regneudtryk. RcComa180 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 helt 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)

### 4.2.2 Beregningspræcision

Når man foretager beregninger i RcComa180, regnes der med 13 betydende cifre. Tiereksponenten kan gå fra -128 til 126, dvs. RcComa180 kan regne i følgende positive talområde:

1.000000000000E-128 ≤ n ≤ 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 manuel.

## 4.3 Strengvariable

Indtil nu har vi set på, hvorledes man foretager udregninger i RcComa180. Man kan dog også behandle tekster i sproget. Tekster kaldes normalt for strenge.

RcComa180 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 symbolet 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 afsn 11.23). 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** På Partner og Piccoline kan man angive et enkelt anførselstegn 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 text$ OF 80
0020 text$:="en lang tekst med mellemrum"
0030 PRINT len(text$)
0040 text$:="0123456789"
0050 PRINT len(text$)
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 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$ := "RcComa180 brugermanual"
```

| Delstreng | Indhold |
|---|---|
| tekst$(1:7) | RcComa1 |
| tekst$(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$:="700"
0030 tekst$:="rc"+nr$
0040 PRINT tekst$
RUN
rc700
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 RcComa180.

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 RcComa180:

| | |
|---|---|
| < | 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 E.

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 udsagn.

**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
```

Da dette udtryk vil blive udregnet således (hvis tal f.eks. er 20):

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

Man kalder ordet AND for en logisk operator.

Følgende logiske operatorer findes i RcComa180 :

| | |
|---|---|
| AND | logisk "og" |
| OR | logisk "eller" |
| NOT | logisk "ikke" |

### 4.5.1 IN-operatoren

I forbindelse med strengbehandling findes en nyttig operator, IN-operatoren. Den angiver om en streng optræder 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 |

## 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 og vil senere blive forkortet med symbolet nudtr.

**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 (↑ på Piccolo) |
| 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

## 5.1 Betingede sætninger

### 5.1.1 IF-sætninger

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

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, hvis 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 ud-
                  tryk er sand (beløb>100) udskrives:
                  Beløbet er større end 100
                  Der gives 10 % rabat
                  Hvorefter beløbet sættes lig 90 % af det gam-
                  le 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 logiske udtryk er sandt, og en anden række, hvis udtrykket er falskt, kan man bruge IF-ELSE-ENDIF konstruktionen.

**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 <u>ikke</u> 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 benytte 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 PROC indsæt, PROC udskriv og PROC 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 <u>ingen</u> WHEN-sætning indeholder et udtryk, der er lig nøgleudtrykket, <u>udskrives en fejlmelding</u>.

Dette kan <u>dog undgås</u> 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 en 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:=200
0040 ELSE
0050   IF brevvægt<=100 THEN
0060     porto:=270
0070   ELSE
0080     IF brevvægt<=250 THEN
0090       porto:=400
0100     ELSE
0110       IF brevvægt<=500 THEN
0120         porto:=700
0130       ELSE
0140         porto:=1000
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:=200
0050 WHEN brevvægt<=100
0060   porto:=270
0070 WHEN brevvægt<=250
0080   porto:=400
0090 WHEN brevvægt<=500
0100   porto:=700
0110 OTHERWISE
0120   porto:=1000
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 RcComa180 eksisterer 3 forskellige løkkestrukturer, de er alle beskrevet i det følgende.

### 5.2.1 Tælleslø jfer (FOR-NEXT)

Begrundelsen for at have tælleslø jfer 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:=i/10+1 TO SIN(y*2)*p DO ...
```

Man skal blot være opmærksom på, at udtrykket <u>kun</u> 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 sløjfen gentager en række sætninger på grundlag af en tællevariabel, kan man også gentage en række sætninger indtil et logiske udtryk (en betingelse) er sand.

**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-sløjfen. Sætningerne i linie 0090-0110 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 sætningen 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 : Formatteret udskrift, der bevirker, at tallene kommer til at stå under hinanden (enere under enere, tiere under tiere osv.). På maskiner uden paragraftegnet benyttes nummertegnet (#).

## 6. Indlæsning og udskrivning

Dette afsnit skal opfattes som en kort fremstilling af de muligheder man har for udskrift og indlæsning på skærm, skriver, tastatur og diskfiler.

En fuldstændig gennemgang af kommandoerne findes i kap 11 i følgende afsnit :

- **AT** (11.3)
- **CLOSE** (11.12)
- **CREATE** (11.18)
- **DELETE** (11.22)
- **EOF** (11.34)
- **GET$** (11.45)
- **INPUT** (11.55)
- **INPUT FILE** (11.56)
- **KEY$** (11.58)
- **OPEN** (11.71)
- **PREFIX** (11.79)
- **PRINT** (11.80)
- **PRINT FILE** (11.81)
- **READ FILE** (11.89)
- **SCREEN$** (11.99)
- **SELECT OUTPUT** (11.100)
- **TAB** (11.108)
- **WRITE FILE** (11.118)

### 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 eller 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, får 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;"Piccolo"

Når ovenstående sætning udføres, fås følgende udskrift:

RC700 Piccolo

## 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 CP/M's unit begreb, A i CP/M svarer til 1, B til 2, C til 3, osv.

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.

**Bemærk** På Partner og Piccoline må man ikke anvende bogstaverne Æ, Ø og Å i filnavne.

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 interns blive lavet om til store bogstaver. Benytter man SAVE og LOAD-kommandoerne tilføjer RcComa180 automatisk typen CSV.

| Lovlige filnavne | Ulovlige filnavne |
|---|---|
| 1/HANOI | 9/Mitprogram |
| 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 RcComa180 udfører systemet en prefix-kommando svarende til den unit man startede RcComa180 op fra. Hvis man startede RcComa180 op fra unit A i CP/M eller C-CP/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 | Terminalporten på maskinen. Inddata skal afsluttes med et linieskift. |
| portnr/PORT | En af systemets HW-porte. |

På Partner og Piccoline findes desuden følgende ydre enheder:

| | |
|---|---|
| 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. |
| 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 afsn 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

### 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 i en bestemt rækkefølge. 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 man har åbnet filen, kan man enten læse eller skrive 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 det sidste dataelement er læst. 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 et 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,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 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 n 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 disken igen med sætningen

DELETE filnavn

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

DELETE-kommandoen har ingen virkning på kassettebåndsfiler.

### 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 RcComa180.

| 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 afsn 11.23). 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 afsn 11.23). 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 procedurebegrebet 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
- slette elevers data
- rette i oplysningerne for en elev

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 elevoplysn
    slet elevoplysninger
INDTIL kommando=færdig
afslut
```

Programmet kunne se således ud i RcComa180:

```
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 <u>procedurenavn</u>, 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 <u>externe</u> 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 alt 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 sløjfen 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)="0" 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 F.

Hvis en parameter i en procedure er en vektor, en matrix eller en teksttabel, skal ordet REF altid angives foran parameteren.

**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 ud, 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.

| 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) |
| Array | PROC p(REF v()) | EXEC p(v) |
| | PROC p(REF m(,,,)) | EXEC p(m) |
| Streng | PROC p(s$) | EXEC p("tekst") eller EXEC p(s$) |
| | PROC p(REF s$) | EXEC p(s$) |
| Teksttabel | PROC p(REF t$()) | EXEC p(t$) |

## 8.4 Lukkede procedurer

Ofte bliver procedurer små selvstændige programmer med egne variabelnavne. For at undgå konflikt med hovedprogrammets variable, kan procedurene 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 sløjfer indeni hinanden. Når stjerner er blevet udført n gang, vil x have værdien 10, og slutbetingelsen for sløjfen 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 det samme som hovedprogrammets j.

linie 0030 : i sættes lig 10. Da i <u>ikke</u> 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 diskfiler, der indeholder de tilhørende procedurer. Procedurene er skrevet 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, i hvilken fil proceduren befinder sig.
linie 0020 : Erklæring af, i hvilken fil proceduren befinder sig, samt en angivelse af, hvilke parametre proceduren har.
linie 0030 : Kald af PROC a
linie 0040 : Kald af PROC stjerne

Flere externe procedure må ikke gemmes i samme 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 Handler

En HANDLER er en RcComa180 struktur, der har form som en procedure. Forskellen er blot, at man ikke kan "kalde" en HANDLER med EXEC, men at RcComa180 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 fejlmed.
0100 DISABLE
0110 ...
```

**Bemærkninger**

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

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

linie 0060 : Skærmen slettes

linie 0070 : HANDLERen fejl aktiveres.
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 aktiv ad gangen. ENABLEr man en HANDLER medens en anden HANDLER er ENABLEd, DISABLEs den første HANDLER automatisk.

Man kan 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 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, hvis værdi er lig nummeret på den fejl, der kaldte HANDLERen. Hvis fejl 0213 (FIL EKSISTERER ALLEREDE) opstår, skal programudførslen fortsætte med linien efter den linie hvori fejlen opstod
linie 0040 : HANDLERen fejl aktiveres
linie 0050 : Hvis filen med navnet randomfil eksisterer, vil HANDLERen blive kaldt.
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 normalfejlmeddelelse. 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

Udover de indbyggede funktioner i RcComa180 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 : Funktionen kan udskrives som enhver anden numerisk variabel eller funktion

linie 0110 : FUNC kan optræde alle steder, hvor der forventes et numerisk udtryk.

Desuden kan brugeren definere strengfunktioner.

**Eksempel**

```
0010 FUNC uppercase$(1$) CLOSED
0020   FOR i:=1 TO LEN(1$) DO
0030     IF 1$(i:)>=CHR$(96) THEN
0040       1$(i:):=CHR$(ORD(1$(i:)-32)
0050     ENDIF
0060   NEXT i
0070   RETURN 1$
0080 ENDFUNC uppercase$
0090 PRINT uppercase$("RcComa180")
0100 PRINT uppercase$("RcComa180")(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 RcComa180, 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 systemkommandoer er:

| | | | |
|---|---|---|---|
| - | AUTO | (11.5) | Automatisk linienummerering |
| - | BYE | (11.6) | Returner til CP/M |
| - | CON | (11.14) | Fortsæt programudførelse |
| - | COPY | (11.16) | Kopier fil |
| - | DEL | (11.21) | Slet programlinie(r) |
| - | DELETE | (11.22) | Slet fil |
| - | DIR | (11.24) | Udskriv fortegnelse over filer |
| - | EDIT | (11.29) | Ret programlinie(r) |
| - | ENTER | (11.32) | Indlæs program |
| - | LIST | (11.60) | Udskriv program |
| - | LOAD | (11.61) | Hent program |
| - | MOUNT | (11.66) | Monter diskette |
| - | NEW | (11.69) | Slet program- og data-lager |
| - | RENAME | (11.90) | Omdøb fil |
| - | RENUMBER | (11.91) | Omnummerer linienumre i program |
| - | RUN | (11.97) | Udfør program |
| - | SAVE | (11.98) | Gem program |
| - | SIZE | (11.103) | Udskriv forbrugt lager |

En systematisk gennemgang af kommandoerne fås i kapitel 11, og dette kapitel skal udelukkende opfattes som en kort introduktion.

### 9.1 Kommandoer til programindtastning

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

De eneste "nye" kommandoer, er kommandoer til rettelse af et program, samt en kommando til at udskrive hvor meget lager, der er brugt.

Hvis man skal rette mange linier i et program, er det ikke altid nok at LISTe programmet på skærmen, og derefter foretage rettelserne ved hjælp af cursor-pilene og Retur-tasten. I stedet 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, RUB OUT, indsæt-tegn og slet-tegn tasterne. Når rettelserne er foretaget tastes Retur, og den næste linie 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.

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    23131
```

hvilket betyder, at det program, man har indtastet fylder 310 tegn i lageret og at der på nuværende tidspunkt er 23131 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 RcComa180 sætninger kan udføres samtidig med at de indtastes. Dette gøres ved udeladelse af linienummeret i programlinien.

Denne facilitet er speciel nyttig i forbindelse med

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

Der er dog en række RcComa180 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 RcComa180-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 lukker 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 på disk nr 1 med kommandoerne

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

## 9.3 Diskkommandoer (MOUNT og DIR)

Hvis man på en Piccolo skifter diskette eller ønsker at benytte disketteenhed nummer 2, skal man markere dette overfor systemet. Hvis man har skiftet disketten i enhed nummer 1, gøres dette med kommandoen:

```
MOUNT 1
```

Hvis disketten i enhed nummer 2 ønskes benyttet, eller hvis disketten er skiftet, skrives

```
MOUNT 2
```

Ønsker man at udskrive en oversigt over filerne på disk nummer 1 er kommandoen

```
DIR 1
```

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

Begge kommandoer kan desuden benyttes som 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 (og underafsnit).

Den eneste "nye" kommando er 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 diskfil. 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 diskfil. 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.

## 9.5 BYE

Med kommandoen

BYE

forlader man RcComa180 og returnerer til styresystemet.

## 10. Grafik

RcComa180 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.

På Piccolo skal man have installeret et specielt grafik-kort for at kunne udføre kommandoerne.

Samtlige grafikkommandoer er:

- **CIRCLE** (11.10)
- **CLEAR** (11.11)
- **CLOSE GRAPHICS** (11.13)
- **DRAW** (11.27)
- **DRAWTO** (11.28)
- **GPARM** (11.48)
- **GSX** (11.49)
- **LOCATE** (11.62)
- **MOVE** (11.67)
- **MOVETO** (11.68)
- **OPEN GRAPHICS** (11.72)
- **PALETTE** (11.76)
- **PENCOLOR** (11.77)
- **TEXT** (11.110)
- **WINDOW** (11.117)

En detaljeret gennemgang af grafikkommandoerne findes i kapitel 11, 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
  'G Start grafik'

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

Herefter er det muligt at starte grafikken op.

På Piccolo udstyret med grafik-kort skal der ikke foretages noget specielt, før man kan udføre grafik-programmer.

## 10.2 Grafik i RcComa180

Selve fremgangsmåden for grafik i RcComa180 er følgende:

1. Man vælger den grafiske enhed, dvs. skærm, plotter eller lignende. På Piccolo kan man kun vælge skærmen.

2. Det grafiske vindues ramme defineres

3. Tegning indenfor vinduet

4. Afslutning af det aktuelle billede.

### 10.2.1 Valg af grafisk enhed

I RcComa180 kan man tegne på en grafisk enhed 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 |

Læg mærke til, at disse tal er uafhængige af RcComa180. Tallene defineres i GSX i det modul, der hedder ASSIGN.SYS.

Bemærk på Piccolo kan man <u>kun</u> vælge skærmen.

### 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.

```
ymax -+-------------------------+-
      !                         !
      !                         !
ymin -+-------------------------+-
      !                         !
     xmin                      xmax
```

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. RcComa180 nøgleord

Dette afsnit skal betragtes som et referenceafsnit, som beskriver alle nøgleordene i RcComa180, dvs sætninger, kommandoer, funktioner, strukturer og operatorer. Dog er de almindelige regneoperatorer (+,-,*,/,') og sammenligningsoperatorerne (=,<>, <,>,<=,>=) ikke beskrevet.

Beskrivelsen af nøgleordene har følgende format:

**11.x RcComa180 nøgleord**
**Type**
**Format**
**Operator prioritet**
**Anvendelse**
**Virkemåde**
**Bemærkninger**
**Eksempler**

hvor

| | | |
|---|---|---|
| x | : | afsnitsnummeret |
| RcComa180 nøgleord | : | et eller flere reserverede RcComa180 ord |
| Type | : | om nøgleordet er en sætning, kommando, 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 <u>ikke skal</u>, men <u>kan</u> indtastes
- Tre punktummer ... angiver, at det <u>foregående</u> argument kan gentages
- Symboler der er <u>understregede</u> beskrives senere i afsnittet

| | | |
|---|---|---|
| Operator prioritet | : | et tal, der angiver i hvilken rækkefølge operatorerne udføres |
| Anvendelse | : | en kort beskrivelse af nøgleordet |
| Virkemåde | : | en beskrivelse af virkemåden for mere indviklede RcComa180 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 F. |

**BEMÆRK:** Ikke alle afsnit anvendes ved alle nøgleord.

## 11.1 ABS

RcComa180 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 positivt 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
```

## 11.2 AND

RcComa180 operator

Format
nudtr1 AND nudtr2

nudtr1: et vilkårligt numerisk udtryk opfattet som logisk udtryk (0: falsk, <>0: sand).
nudtr2 : et vilkårligt numerisk udtryk opfattet som logisk udtryk (0: falsk, <>0: sand).

**Operatorprioritet** = 7 (se afsn 4.6.1)

**Anvendelse**
Den logiske operator AND er sand (<>0), hvis både nudtr1 og nudtr2 er sande (<>0). Hvis enten nudtr1 eller nudtr2 er falsk (=0), bliver resultatet også falsk (=0).

**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
```

## 11.3 AT

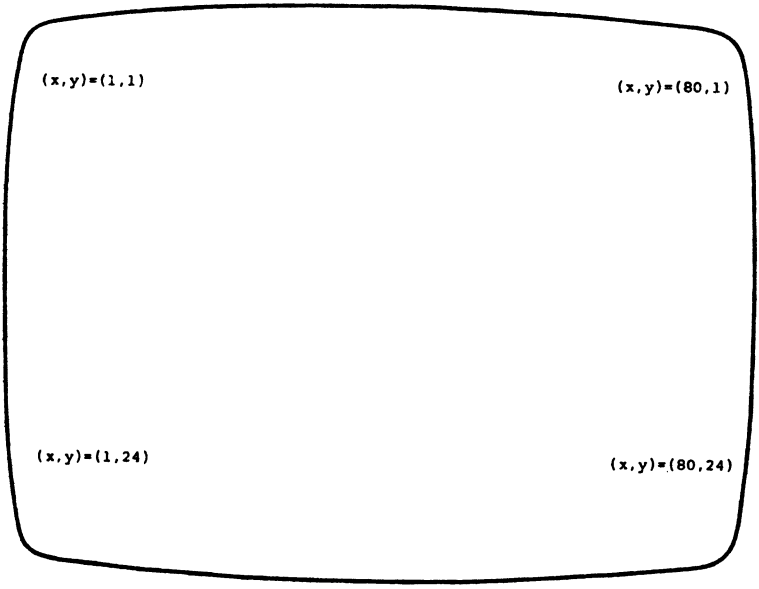
RcComa180 funktion

**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 <= 24 eller 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.På Partner og Piccoline må nudtr2 højst være 24. På Piccolo må nudtr2 højst være 25.

**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);"!";// 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
```

## 11.4 ATN

RcComa180 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
```

| 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 |

## 11.5 AUTO

RcComa180 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 er ved at blive indtastet.

**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. |

## 11.6 BYE

RcComa180 sætning/kommando

**Format**
BYE

**Anvendelse**
Sætningen/kommandoen anvendes til at returnere til styresystemet.

**Eksempel**
Med nedenstående logon-program (se appendix G) kan ikke alle og enhver starte RcComa180 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
```

## 11.7 CASE-WHEN-ENDCASE

RcComa180 struktur

**Format**
CASE udtryk0 OF
WHEN udtrykla ,udtryklb...
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 :RcComa180 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 fejl 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
```

## 11.8 CHAIN

RcComa180 sætning

**Format**
CHAIN filnavn

filnavn :strengudtryk, der angiver et filnavn.

**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 | Bemærk, at filnavnet i status- |
| AT 0030 | linien er ændret til programa. |

## 11.9 CHR$

RcComa180 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 E.

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
```

## 11.10 CIRCLE

RcComa180 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
```

## 11.11 CLEAR

RcComa180 grafik sætning/kommando

**Format**
CLEAR

**Anvendelse**
Sætningen sletter grafikbilledet.

## 11.12 CLOSE

RcComa180 sætning/kommando

**Format**
CLOSE [ [ FILE ] strømnr ]

strømnr :et numerisk udtryk, hvis værdi angiver nummeret på en datastrøm.

**Anvendelse**
Sætningen/kommandoen bruges til at lukke en eller alle datastrømme.

**Bemærkninger**
1. Anføres CLOSE uden strømnummer, 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
```

## 11.13 CLOSE GRAPHICS

RcComa180 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.

## 11.14 CON

RcComa180 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 før CON.

3. Hvis man retter i et program og derefter skriver CON, fås fejludskriften 0095: CON IKKE TILLADT.

## 11.15 CONTINUE

RcComa180 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
```

## 11.16 COPY

RcComa180 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.

**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"
```

## 11.17 COS

RcComa180 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
```

## 11.18 CREATE

RcComa180 sætning/kommando

**Format**
CREATE filnavn , længde

filnavn :navnet på en diskette fil, 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. Har man skiftet diskette uden at udføre MOUNT-sætningen fås fejl 208: DISKETTE ER BLEVET SKIFTET.

5. Er der ikke plads til flere filer på disketten fås fejl 215: 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
```

## 11.19 DATA

RcComa180 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 så kan læses ved hjælp af READ-sætningen.

**Bemærkninger**
1.Sætningen kan ikke udføres som kommando.

2.Ved starten af et programs udførsel (RUN) gennemgås programmet sekventielt, alle DATA-sætningerne kædes sammen til en dataliste og en datapegepind sættes til at pege på det første element i den første DATA-sætning.

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
```

## 11.20 DATE$

RcComa180 funktion

**Format**
DATE$

**Anvendelse**
DATE$ er en funktion der returnerer den aktuelle dato.

**Bemærkninger**
1. DATE$ har formen ÅÅÅÅ.MM.DD, hvor ÅÅÅÅ er året (f.eks. 1984), MM er måneden (f.eks. 09), DD er dagen (f.eks. 30)

## 11.21 DEL

RcComa180 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, | slette linierne 0040, 0050, 0060 |
| del ,20 | slette linierne 0010, 0020 |
| del 30,40 | slette linierne 0030, 0040 |

## 11.22 DELETE

RcComa180 sætning/kommando

Format

DELETE filnavn

filnavn :Navnet på en diskettefil, der ønskes slettet.

**Anvendelse**
Sætningen/kommandoen sletter filen filnavn fra diskettens katalog.

**Bemærkninger**
1. Selv om filen ikke eksisterer, udskrives der ikke fejlmeddelelse.

2. Er disketten skrivebeskyttet udskrives fejlen 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 fejl
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
```

## 11.23 DIM

RcComa180 sætning/kommando

**Format**

```
DIM { array      } [ , { array      }]
    { streng     }      { streng     }
    { tekstabel  }      { tekstabel  }      ...
```

| | |
|---|---|
| array: | navn(grænse ,grænse ... ) |
| streng : | navn$ OF nudtr |
| tekstabel: | 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 tekstabeller.

**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)

## 11.24 DIR

RcComa180 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 CP/M unit A,B,C,...

**Bemærkning**
1. Normalt udskrives listen på skærmen. Dette kan dog ændres med SELECT OUTPUT sætningen.

2. Angives nudtr ikke, forventes 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. RcComa180 vil 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"<br>DIR 1 | Udskrift af navnene på diskette nr 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
```

## 11.25 DISABLE

RcComa180 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
```

## 11.26 DIV

RcComa180 operator

**Format**
nudtr1 DIV nudtr2

nudtr1: vilkårligt numerisk udtryk
nudtr2: vilkårligt numerisk udtryk forskellig fra nul.

Operator prioritet = 3

**Anvendelse**
Operatoren foretager en heltalsdivision

**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
```

## 11.27 DRAW

RcComa180 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).

**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
```

## 11.28 DRAWTO

RcComa180 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).

**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
```

## 11.29 EDIT

RcComa180 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 syntaks analyseres og lagres. Derefter udskrives den næste sætninger, 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.

## 11.30 ENABLE

RcComa180 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
```

## 11.31 END

RcComa180 sætning

**Format**
END

**Anvendelse**
Benyttes til at markere afslutningen af et program.

**Bemærkninger**
1.Sætningen kan ikke udføres som kommando.

2.RcComa180 kræver ikke, at programmet indeholder en END sætning, da programudførslen 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
```

## 11.32 ENTER

RcComa180 kommando

**Format**
ENTER filnavn

filnavn :navnet på en diskettefil angivet i anførselstegn.

**Anvendelse**
Kommandoen bruges til at sammenflette programlinierne i en diskettefil med programlinierne i programlageret.

**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.

| Eksempel | Kommentar |
|---|---|
| 20 // program a | |
| 30 a:=7 | |
| list "eksempela" | Nu er programmet gemt 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. |

## 11.33 EOD

RcComa180 funktion

**Format**
EOD

**Anvendelse**
EOD er en logisk funktion uden argumenter, som har værdien SAND (<>0) 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
```

## 11.34 EOF

RcComa180 funktion

**Format**
EOF(strømnr)

strømnr: et udtryk, hvis værdi er lig nummeret på en datastrøm.

**Anvendelse**
EOF er en logisk funktion, som antager værdien SAND (<>0) 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
```

## 11.35 ERR

RcComa180 funktion

**Format**
ERR

**Anvendelse**
Bruges kun i PROC-HANDLER. Funktionen returnerer med nummeret for sidste RUN-time fejl.

**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
```

## 11.36 ERRTEXT$

RcComa180 funtion

**Format**
ERRTEXT$(nudtr)

nudtr: Et positivt numerisk udtryk

**Anvendelse**
Funktionen returnerer fejlmeddelelsen, der svarer til et givet fejlnummer.

## 11.37 Etikette

RcComa180 sætning

**Format**
navn:

navn: Et navn lavet efter samme regler som 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, RESTORE.

2.En variabel og en etikette må ikke have det samme navn.

**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
```

## 11.38 EXEC

RcComa180 sætning/kommando

**Format**

[EXEC]navn [ (par [,par] ... ) ]

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 afsnit 8: Procedurer og funktioner

## 11.39 EXP

RcComa180 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
```

## 11.40 FALSE

RcComa180 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.
```

## 11.41 FOR

RcComa180 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 RcComa180 sætning, normalt CIRCLE, CLOSE, CREATE, DELETE, DRAW, DRAWTO, EXEC, GSX, INPUT, INPUT FILE, LOCATE, MOUNT, MOVE, MOVETO, OPEN FILE, PENCOLOR, PRINT, PRINT USING, PRINT FILE, READ, READ FILE, RESTORE, SELECT OUTPUT, TEXT, tildeling, 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, bliver den sat 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

## 11.42 FOR-NEXT

RcComa180 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: | RcComa180 sætninger |

**Anvendelse**

Konstruktion benyttes til at repetere udførslen 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 fejl 0096: FEJL I PROGRAMSTRUKTUR.

4. Hvis man hopper ind i en FOR-NEXT løkke udenom FOR-sætningen fås fejl 0116: ULOVLIGT HOP.

## 11.43 FUNC-ENDFUNC

RcComa180 struktur

**Format**

```
FUNC navn[( par [ ,parl.. )][CLOSED]
  sætningsliste
ENDFUNC navn
```

```
                  [REF ] nvar
                  [REF ] svar
par:              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:    RcComa180 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 variablerne i resten af programmet (de globale variable). Ønsker man at benytte en variabel, der optræder i resten af programmet, benyttes GLOBAL-sætningen. 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.

**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
```

## 11.44 FUNC-EXTERNAL

RcComa180 sætning

**Format**

FUNC navn[(par [ ,par] ... )] EXTERNAL filnavn

```
                [REF]  nvar
        {       [REF]  svar                  }
par:    {       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.

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.

## 11.45 GET$

RcComa180 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, hexta1$ OF 16
0020 DIM buf$ OF 512, svar$ OF 3
0030 hexta1$:= "0123456789ABCDEF"
0040 INPUT "Unitnr : ": unit$
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$(i:i)) MOD 16 + 1
0260       PRINT hexta1$(i1:i1);hexta1$(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
```

## 11.46 GLOBAL

RcComa180 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.

## 11.47 GOTO

RcComa180 sætning

**Format**
GOTO navn

navn: Navnet på en etikette.

**Anvendelse**
Sætningen anvendes til at foretage et hop i programudførslen.

**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, PROC-ENDPROC, REPEAT-UNTIL, WHILE-ENDWHILE og FOR-NEXT.

3. Man må ikke hoppe ud af PROC-ENDPROC.

**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
```

## 11.48 GPARM

RcComa180 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. 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:<br>0: Enheden er i stand til at producere præcist skalerede billeder<br>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.

## 11.49 GSX

RcComa180 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 Programmer's Guide.

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 11.117)

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 11.117)

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.

**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 af markeringen i efterfølgende POLYLINE 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 POLYMARKER 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(1) : 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):= y
0030   pind(3,1):= x1; pind(3,2):= y1; pind(4,1):= x0; pind(4,2):= y
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):= y
0030   pind(3,1):= x1; pind(3,2):= y1; pind(4,1):= x0; pind(4,2):= y
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):= y
0100   pind(3,1):= x1; pind(3,2):= y1; pind(4,1):= x0; pind(4,2):= y
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
```

## 11.50 IF-THEN

RcComa180 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 RcComa180 sætning, dvs. CIRCLE, CLEAR, CLOSE, CLOSE GRAPHICS, CONTINUE, CREATE, DELETE, DRAW, DRAWTO, END, EXEC, GOTO, GPARM, GSX, INPUT, INPUT FILE, LOCATE, MOUNT, 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 udførslen 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
```

## 11.51 IF-THEN-ENDIF

RcComa180 struktur

**Format**

IF logisk udtryk THEN
  sætningsliste
ENDIF

logisk udtryk:et udtryk, som kan have værdien sand (<>0) eller falsk (=0)
sætningsliste:RcComa180 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
```

## 11.52 IF-THEN-ELSE-ENDIF

RcComa180 struktur

**Format**

IF logisk udtryk THEN
  sætningsliste 1
ELSE
  sætningsliste 2
ENDIF

logisk udtryk :et udtryk, som kan have værdien sand (<>0) eller falsk (=0).

sætningsliste 1:en række RcComa180 sætninger der vil blive udført, hvis værdien af logisk udtryk er sand (<>0).

sætningsliste 2:en række RcComa180 sætninger der vil blive udført, hvis værdien af logisk udtryk er falsk (=0).

**Anvendelse**

Konstruktionen benyttes til at udføre en af to blokke RcComa180 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 programudførelsen med første sætning efter ENDIF.

**Bemærkning**

1.Hvis man hopper ind i sætningsliste 1 eller sætningsliste 2 udenom IF-THEN-ELSE, udskrives fejlmeddelelsen 116: 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
```

## 11.53 IMPORT

RcComa180 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.

## 11.54 IN

RcComa180 operator

**Format**
sudtr1 IN sudtr2

(sudtr1 IN sudtr2) returnerer et ikke negativt heltal
sudtr1: strengudtryk
sudtr2: strengudtryk

**Operatorprioritet** = 5

**Anvendelse**
Operatoren finder positionen for sudtr1 i sudtr2.

**Bemærkninger**
1.Operatoren returnerer positionen for sudtr1's første tegn i sudtr2.

2.Findes sudtr1 ikke i sudtr2 returneres værdien 0.

3.Hvis sudtr1 er tom returneres med længden af sudtr2 plus 1.

**Eksempel**
```
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
```

## 11.55 INPUT

RcComa180 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 AT afsnit 11.3).

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.

## 11.56 INPUT FILE

RcComa180 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 E) 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 afsn. 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
```

## 11.57 INT

RcComa180 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
```

## 11.58 KEY$

RcComa180 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$ (se appendix F.4).

**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
```

## 11.59 LEN

RcComa180 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 8
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
```

## 11.60 LIST

RcComa180 kommando

**Format**

```
LIST   [{ lnr1 [ , ]  }]   [ filnavn ]
        {,   lnr2     }
         { lnr1,lnr2  }
```

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.

2. Kombinationerne af parametrene til LIST-komamndoen har følgende betydning:

| | |
|---|---|
| LIST | Udskriver hele programmet i programlageret. |
| 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 print
0020 print
0030 print
0040 print
0050 print
0060 print
```

Da vil:

| | | |
|---|---|---|
| 1. | LIST | udskrive linierne 0010, 0020, 0030, 0040, 0050, 0060 |
| 2. | LIST 20 | udskrive linien 0020 |
| 3. | LIST 40, | udskrive linierne 0040, 0050, 0060 |
| 4. | LIST ,20 | udskrive linierne 0010, 0020 |
| 5. | LIST 30,40 | udskrive linierne 0030, 0040 |

**Eksempel 2**

Ønskes programmet i programlageret udskrevet på printeren udføres kommandoerne

```
LIST "printer"
```

## 11.61 LOAD

RcComa180 kommando

**Format**
LOAD [ filnavn ]

filnavn: navnet på en fil 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 lukker evt. åbne filer.

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 på Partner og Piccoline.

**Eksempel**
LOAD "/2/MITPROG"

## 11.62 LOCATE

RcComa180 grafik sætning/kommando

**Format**
LOCATE tast ,xkoor ,ykoor

tast : en stregnvariabel
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.

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 !"
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$="!"
```

## 11.63 LOG

RcComa180 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
```

## 11.64 MARGIN

RcComa180 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.

**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
```

## 11.65 MOD

RcComa180 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**

**Anvendelse**
Operatoren udregner resten ved en heltalsdivision

**Bemærkninger**
1.Definitionen på a MOD b er lig
a-INT(a/ABS(b))*ABS(b)

2.RcComa180 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
```

## 11.66 MOUNT

RcComa180 sætning/kommando

**Format**
MOUNT nudtr

nudtr: Et numerisk udtryk, hvis værdi angiver et unitnr (dvs. 1,2,3,...).

**Anvendelse**
Sætningen/kommandoen benyttes til at angive overfor systemet at der er monteret en ny diskette i drive nr. nudtr, hvorpå der må læses og skrives.

**Bemærkning**
1. Udskiftes en diskette i et drive, skal man udføre en MOUNT, førend man kan skrive eller læse på disketten. Er maskinen udstyret med 8" disketter fås ellers fejlmeldingen 0208: DISKETTE ER BLEVET SKIFTET.

2. Man må ikke skifte diskette og udføre MOUNT før alle filer på disketten er lukket (CLOSEd), ellers fås fejl 0219: ÅBNE FILER PÅ ENHED.

## 11.67 MOVE

RcComa180 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).

**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
```

## 11.68 MOVETO

RcComa180 grafik sætning/kommando

**Format**
MOVETO xkoor ,ykoor

xkoor,ykoor: numeriske udtryk

**Anvendelse**
Sætningen/kommandoen flytter det løbende punkt (x,y) til punktet (xkoor,ykoor).

**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
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
```

## 11.69 NEW

RcComa180 sætning/kommando

**Format**
NEW [ filnavn ]

filnavn :navnet på en diskettefil angivet i anførselstegn.

**Anvendelse**
Sætningen/kommandoen bruges til at slette programmet og data i maskinens lager, samt lukke eventuelle åbne filer.

**Kommentarer**
1.NEW anvendes, når brugeren ønsker at indtaste et nyt program.

2.MARGIN- og ZONE-værdierne sættes til 0 ved NEW.

3.Alle åbne datastrømme lukkes ved NEW.

4.Angives filnavn vil filnavnet blive udskrevet i statuslinien, ellers vil den indeholde navnet
1/SAVE.$$$ på Piccolo, og
01/Save.Csv på Partner og Piccoline

## 11.70 NOT

RcComa180 operator

**Format**
NOT nudtr

nudtr: vilkårligt numerisk udtryk opfattet som logisk udtryk (0:falsk, <>0: sand).

**Operatorprioritet = 6**

**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
```

## 11.71 OPEN

RcComa180 sætning/kommando

**Format**

```
                                  READ
OPEN FILE strømnr, filnavn, { WRITE      }
                            { APPEND     }
                                  RANDOM recl
```

strømnr: Vilkårligt numerisk heltal i intervallet 1 <= strømnr <= 5
filnavn: Navnet på en ydre enhed eller en fil.
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 (random access) 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 (8 bit bytes). Følgende udregningsregler skal anvendes:

   - en numerisk variabel fylder 8 tegn

- en streng fylder "længden af strengen + 2 " tegn

6.Iøvrigt henvises til afsnit 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
```

## 11.72 OPEN GRAPHICS

RcComa180 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. Sætningen/kommandoen har ingen virkning på Piccolo

2. nudtr = 1 vælger normalt skærmen

3. Sammenhængen mellem nudtr og det faktiske grafik-enheder fremgår af GSX-modulet ASSIGN.SYS.

4. Der gælder normalt følgende regler for nummerering af enheder:
   1-10 : Skærm
   11-20 : Plottere
   21-30 : Skrivere
   31-40 : Andre grafikenheder

## 11.73 OR

RcComa180 operator

**Format**
nudtr1 OR nudtr2

nudtr1: vilkårligt numerisk udtryk opfattet som logisk udtryk (0= falsk, <>0= sand).
nudtr2: vilkårligt numerisk udtryk opfattet som logisk udtryk (0=falsk, <>0= sand).

**Operatorprioritet = 8**

**Anvendelse**
Den logiske operator OR er sand (<>0), hvis enten nudtr1 er sand (<>0), eller nudtr2 er sand (<>0), eller begge er sande.

Udtrykket er altså kun falsk (=0), hvis både nudtr1 og nudtr2 er falske.

**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
```

## 11.74 ORD

RcComa180 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 E.

2. ORD-funktionen må indgå i et vilkårligt numerisk udtryk.

## 11.75 OUT

RcComa180 sætning/kommando

**Format**
OUT filnavn

filnavn: navnet på en datastrøm.

**Anvendelse**
Sætningen/kommandoen anvendes til at vælge udskriftsdatastrøm.

**Bemærkning**
1. Sætningen/kommandoen er identisk med sætningen/kommandoen SELECT OUTPUT(11.100).

## 11.76 PALETTE

RcComa180 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. 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
```

## 11.77 PENCOLOR

RcComa180 grafik sætning/kommando

**Format**
PENCOLOR nudtr

nudtr : et positivt numerisk udtryk

**Anvendelse**
Sætningen/kommandoen vælger hvilken "farve" man skal tegne med (i DRAW, DRAWTO, CIRCLE og TEXT sætningerne).

**Bemærkning**
1. Hvis nudtr er 0 vælges baggrundsfarven.

## 11.78 PI

RcComa180 konstant

**Format**
PI

**Anvendelse**
PI er en konstant med værdien 3.141592653590

## 11.79 PREFIX

RcComa180 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 unit man startede RcComa180 op fra. Hvis man f.eks. startede RcComa180 op fra CP/M unit A vil systemet sætte præfixet til "1/".

3. Ønsker man ikke at benytte præfixet i et filnavn, skal filnavnet blot indledes med "/".

## 11.80 PRINT

RcComa180 sætning/kommando

**Format**

PRINT[{ nudtr / sudtr / printfkt } [ { , / ; } { nudtr / sudtr / printfkt }] ... [{ , / ; }]]

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 (11.3), TAB(11.105)).

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.

## 11.81 PRINT FILE

RcComa180 sætning/kommando

**Format**

PRINT FILE str: [{ nudtr / sudtr / printfkt } [{ , / ; }  { nudtr / sudtr / printfkt }] ... [{ , / ; }]]

str: et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm
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 i 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 for PRINT (11.77).

## 11.82 PRINT FILE USING

RcComa180 sætning/kommando

**Format**
PRINT FILE strømnr: USING sudtr: nudtr [,nudtr]... [;]

strømnr: et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm.
sudtr: streng udtryk, format streng
nudtr: numerisk udtryk

**Anvendelse**
Sætningen/kommandoen anvendes til at udskrive værdier i en datastrøm i et bestemt format.

**Bemærkninger**
1.Se bemærkningerne til PRINT FILE (11.78) og PRINT USING (11.80).

## 11.83 PRINT USING

RcComa180 sætning/kommando

**Format**
PRINT USING sudtr: nudtr [ ,nudtr ]... [ ; ]

sudtr: streng udtryk, format streng.
nudtr: numerisk udtryk

**Anvendelse**
Sætningen/kommandoen anvendes til at udskrive værdier i et bestemt format.

**Bemærkninger**
1. sudtr må være et vilkårligt strengudtryk, der udskrives direkte, idet # og . ændres efter følgende regler:
   **NB: : angiver mellemrum i udskriftsformatet i det følgende.**

   **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 formattet, 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.

3. På maskiner uden paragraftegn (#) benyttes nummertegnet ( ).

**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
```

## 11.84 PROC-ENDPROC

RcComa180 struktur

**Format**

```
PROC navn   [ ( par   [ ,par] ... ) ]      [ CLOSED ]
  sætningsliste
ENDPROC navn
                    [REF] nvar
par:            {   [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 teksttabelnavn
sætningsliste:RcComa180 sætninger.
```

**Anvendelse**

Konstruktionen benyttes til at definere en procedure.

**Bemærkninger**

1. navn skal være forskellig fra alle andre navne på funktioner, etiketter, reelle 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 i resten af programmet, benyttes GLOBAL-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 reelle parameter) blive ændret, samtidig med at den formelle parameter ændres nede i proceduren. På denne måde kan man returnere resultater via parametrene.

**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
0070 PROC init
0080   MARGIN 0
```

```
0080   piccolo:=(SYS(7)=1)
0090   PRINT CHR$(12)
0100   IF piccolo THEN PRINT AT(1,2);CHR$(27);CHR$(132)
0110   DIM højde(3),pos(3)
0120   højde(1):=antal; højde(2):=0; højde(3):=0
0130   pos(1):=5; pos(2):=30; pos(3):=55
0140   DIM skive$(antal) OF 22,tom$ OF 22,blok$ of 1
0150   IF piccolo THEN
0160     blok$:=CHR$(127)
0170   ELSE
0180     blok$:=CHR$(223)
0190   ENDIF
0200   tom$:="                      "
0210   skive$(1):="          "+blok$+blok$+"          "
0220   FOR i:=2 TO antal DO
0230     skive$(i):=skive$(i-1)(2:11)+blok$
0240     skive$(i)(12:22):=blok$+skive$(i-1)(12:21)
0250   NEXT i
0260   FOR y:=20 TO 21-antal STEP -1 DO
0270     PRINT AT(pos(1),y);skive$(antal+y-20);
0280   NEXT y
0290 ENDPROC init
0300
0310 PROC vis(n,fra,til)
0320   x:=pos(fra)
0330   FOR y:=20-højde(fra) TO 20-antal STEP -1 DO
0340     PRINT AT(x,y);skive$(n);AT(x,y+1);tom$;
0350   NEXT y
0360   retning:=SGN(til-fra)
0370   WHILE x<>pos(til) DO
0380     PRINT AT(x,20-antal);skive$(n);
0390     x:=x+retning
0400   ENDWHILE
0410   FOR y:=20-antal TO 20-højde(til) DO
0420     PRINT AT(x,y);skive$(n);AT(x,y-1);tom$;
0430   NEXT y
0440   højde(fra):=højde(fra)-1; højde(til):=højde(til)+1
0450 ENDPROC vis
0460
0470 PRINT "Tårnene i HANOI"
0480 PRINT
0490 INPUT "Antal skiver (ml. 1 og 10) ":antal
0500 EXEC init
0510 EXEC flyt(antal,1,2,3)
0520 PRINT AT(1,21);
```

## 11.85 PROC-EXTERNAL

RcComa180 sætning

Format
PROC navn [ ( par [,par ]... ) ] EXTERNAL filnavn

```
                [REF]  nvar
                [REF]  svar
par:        {   REF    ntabl ([,]...)   }
                REF    stabl ([,]...)
```

navn: Navn på proceduren (max. 16 tegn)
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.

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.

## 11.86 PROC-HANDLER

RcComa180 struktur

**Format**
PROC navn HANDLER
  sætninger
ENDPROC navn

navn: navnet på handleren (max. 16 tegn).
sætninger: RcComa180 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 11.30).

3. Man kan returnere på 4 forskellige måder fra en PROC-HANDLER:

   a. ENDPROC sætningen nås, hvilket bevirker, at programudførslen standses med en sædvanlig fejludskrift.

   b. Hvis en CONTINUE-sætning udføres, vil programudførslen fortsætte med sætningen efter den sætning, der forårsagede fejlen. (Se afsnit 11.15).

   c. Hvis en RETRY-sætning udføres, vil programudførslen fortsætte med den sætning, der forårsagede fejlen (se afsnit 11.94).

   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.

## 11.87 RANDOMIZE

RcComa180 sætning

**Format**
RANDOMIZE

**Anvendelse**
Sætningen får "tilfældigtal-generatoren" til at starte på et nyt sted i følgen af tilfældige tal.

**Bemærkninger**
1. RND funktionen genererer normalt den samme følge af tilfældige tal, startende med den samme værdi. Dette er praktisk under indkøringen af programmer. Når programmet er testet er det derimod praktisk at starte et nyt sted i følgen. For at opnå dette, skal brugeren angive 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

## 11.88 READ

RcComa180 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.

7. EOD-funktionen bliver sand samtidig med, at det sidste element i DATA-sætningerne bliver læst.

## 11.89 READ FILE

RcComa180 sætning/kommando

**Format**

READ FILE strømnr [,recnr]: { nvar / svar } [ ,{ nvar / svar } ] ...

strømnr: Et taludtryk, hvis værdi angiver nummeret på en åben datastrøm.
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
```

## 11.90 RENAME

RcComa180 sætning/kommando

**Format**
RENAME filnavn1, filnavn2

filnavn1:Navnet på en diskfil, der skal omdøbes.
filnavn2:Det nye navn til filnavn1.

**Anvendelse**
Sætningen/kommandoen anvendes til at omdøbe en diskfil.

**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.

## 11.91 RENUMBER

RcComa180 kommando

**Format**

```
                  [{ lnr   [,]          }]
RENUMBER          [{ , 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 |

## 11.92 REPEAT - UNTIL

RcComa180 struktur

**Format**
REPEAT
  sætningsliste
UNTIL logisk udtryk

sætningsliste:RcComa180 sætninger
logisk udtryk:et udtryk, der kan have 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+i
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."
0020 REPEAT
0030   INPUT "Antal søjler (max 12):": antal
0040 DIM blok$ OF 10,grafik$ OF 2, normal$ OF 2,box$ OF 1
0050 piccolo:=(SYS(7)=1)
0060 IF piccolo THEN
0070   grafik$:=CHR$(27)+CHR$(132)
0080   normal$:=CHR$(27)+CHR$(128)
0090   box$:=CHR$(127)
0100 ELSE
0110   grafik$:=" ";normal$:=" ";box$:=CHR$(223)
0120 ENDIF
0130 blok$:= grafik$+box$+box$+box$+box$+normal$
0140 FOR i:= 1 TO antal DO INPUT "Værdi (ml. 0 og 20):": søjler(i)
0150
0160 PRINT CHR$(12);AT(1,22);normal$
```

```
0170 FOR i:= 1 TO 79 DO PRINT "-";// vandret streg
0180 FOR i:= 1 TO antal DO PRINT AT(i*6+2,23);søjler(i);
0190 værdi:= 1
0200 REPEAT
0210   skrevetsøjle:= FALSE
0220   FOR i:= 1 TO antal DO
0230     IF søjler(i)>=værdi THEN
0240       skrevetsøjle:= TRUE
0250       PRINT AT(i*6,22-værdi);blok$
0260     ENDIF
0270   NEXT i
0280   værdi:= værdi+1
0290 UNTIL NOT skrevetsøjle OR værdi>20
0300 PRINT AT(1,22);
0310 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
```

## 11.93 RESTORE

RcComa180 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
```

## 11.94 RETRY

RcComa180 sætning

**Format**
RETRY

**Anvendelse**
Sætningen anvendes kun i en HANDLER. Sætningen bevirker, at programudførslen fortsætter med den sætning, der forårsagede kaldet af PROC-HANDLER.

**Eksempel**
```
0010 PROC checkdiskette HANDLER
0020   IF SYS(0)=214 THEN
0030     PRINT "*** Fil eksisterer ikke"
0040     PRINT "*** Indsæt korrekt diskette i "
0050     INPUT "*** disketteenhed nummer 1 og tast vognretur": S$
0060     MOUNT 1
0070     RETRY
0080   ENDIF
0090 ENDPROC checkdiskette
0100 DIM S$ OF 1
0110 ENABLE checkdiskette
0120 OPEN FILE 1,"datafil",READ
0130 DISABLE
0140 CLOSE
0150 END
```

## 11.95 RETURN

RcComa180 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 blev 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):":ta
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
```

## 11.96 RND

RcComa180 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 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
0.8394904533458
0.8203299841557
STOP
AT 0030
RUN
0.1116003194179
0.8394904533458
0.8203299841557
STOP
AT 0030
CON
0.0370770494076
0.6477562514615
0.8871226414774
```

RUN kommandoen repeterer de samme tal pány, CON fortsætter følgen.

```
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;". forsøg."
0150   ENDCASE
0160 UNTIL gæt=tal
0170 END
```

## 11.97 RUN

RcComa180 kommando

**Format**
RUN [ filnavn]

filnavn: navnet på en diskettefil angivet i anførselstegn.

**Anvendelse**
Kommandoen starter udførslen af det program, der ligger i programlageret, eller den LOAD'er og udfører et program, der er SAVE'd på en diskettefil.

**Bemærkninger**
1. Datalageret slettes når RUN kommandoen afgives.

2. Programudførslen starter i programlinien med det laveste linienummer.

3. Programnavnet angivet med filnavn skal være gemt med kommandoen SAVE.

4. Ønskes udskriften standset midlertidigt, trykkes på mellemrumstasten. Ved næsten tryk startes den atter.

## 11.98 SAVE

RcComa180 kommando

**Format**
SAVE [ filnavn ]

filnavn: navnet på en fil 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 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**

```
SAVE "/1/MITPROGRAM"
```

## 11.99 SCREEN$

RcComa180 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.

**Eksempel**
Følgende program "roterer" skærmen baglæns

```
0010 WHILE TRUE DO SCREEN$:=SCREEN$(1921:2000)+SCREEN$(1:1920)
```

## 11.100 SELECT OUTPUT

RcComa180 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 stadig 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
```

## 11.101 SGN

RcComa180 funktion

**Format**
SGN (nudtr)

nudtr: et numerisk udtryk

**Anvendelse**
Funktionen har værdien +1 hvis nudtr er større end nul, 0 hvis nudtr er lig nul og -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
```

```
6 er positiv
-6 er negativ
0 er nul
END
AT 0140
```

## 11.102 SIN

RcComa180 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

## 11.103 SIZE

RcComa180 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
```

## 11.104 SQR

RcComa180 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 103: 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
```

## 11.105 STOP

RcComa180 sætning

**Format**
STOP

**Anvendelse**
Sætningen standser udførslen 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.

## 11.106 STR$

RcComa180 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.

**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
```

## 11.107 SYS

RcComa180 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. Piccolo=1, RC855=2, Partner=3, Piccoline=4. |

## 11.108 TAB

RcComa180 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 "#":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
123456789012345678901234567890123456789012345677890123456789 0
* 1
      * 7
                                          * 40
                                        * 38
```

## 11.109 TAN

RcComa180 funktion

**Format**
TAN (nudtr)

nudtr : et numerisk udtryk, der angiver et radianantal.

**Anvendelse**
Funktionen udregner tangens til en vinkel udtrykt i radianer.

## 11.110 TEXT

RcComa180 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.

2. Sætningen/kommandoen kan kun anvendes på enten Partner eller Piccoline

### 11.111 Tildeling

RcComa180 sætning/kommando

**Format**

```
{ nvar:= nudtr } [ ;{ nvar:= nudtr}]...
{ svar:= sudtr }   { svar:= sudtr}
```

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.

**Eksempel**

```
0010 DIM navn$ OF 20
0020 navn$:="COMAL80"
0030 ejok:=(navn$<"COMAL")
0040 pi:=ATN(1)*4; slut:=FALSE
```

## 11.112 TIME$

RcComa180 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).

## 11.113 TRUE

RcComa180 konstant

**Format**
TRUE

**Anvendelse**
TRUE er en konstant med værdien 1

**Eksempel**

```
0010 // Eksempel på en uendelig løkke
0020 WHILE TRUE DO
0100 ENDWHILE
```

## 11.114 VAL

RcComa180 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 talrepresentation (incl. decimal- og exponentiel-notation).

3. VAL udfører det modsatte af STR$.

## 11.115 WHILE

RcComa180 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 RcComa180 sætning, f.eks. CIRCLE, DRAW, DRAWTO, EXEC, INPUT, INPUT FILE, LOCATE, MOVE, MOVETO, PRINT, PRINT FILE, READ, READ FILE, TEXT, tildeling, 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 benyttes.

## 11.116 WHILE-ENDWHILE

RcComa180 struktur

**Format**
WHILE logisk udtryk DO
  sætningsliste
ENDWHILE

logisk udtryk:et udtryk, der når det udregnes, har værdien sand (<>0) eller falsk (=0).
sætningsliste:RcComa180 sætninger.

**Anvendelse**
Konstruktionen benyttes til at gentage udførelsen af sætnings liste 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- og WHILE-ENDWHILE sætninger.

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"
```

## 11.117 WINDOW

RcComa180 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 grafikvindue.

**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 akser0180 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
```

## 11.118 WRITE FILE

RcComa180 sætning/kommando

Format

WRITE FILE strømnr [,recnr] : { nudtr | sudtr } [ , { nudtr | sudtr } ] ...

strømnr: et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm.
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.Dataene 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
```

## 11.119 ZONE

RcComa180 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 derefer 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
```

```
1         2         3
4         5         6
7         8         9
10
1    2    3    4    5    6
7    8    9    10
```

## A. Referencer

(1) RCSL Nr. 42-i2328:
RC702/RC703 mikrodatamatsystem
Brugervejledning

(2) RCSL NR. 42-i2340:
RcComa180 Referencekort

(3) RCSL NR. 42-i2190:
CP/M for the RC702 Microcomputer System
Users Guide

(4) RCSL NR. 42-i2320:
CP/M - RC703
Users Guide

(5) SW1698D
GSX 86 Manual Set

(6) SW1697D
Programmers Guide

(7) SW1500D
RcPartner Brugervejledning

(8) RCSL NR. 42-i2341:
CP/M Referencekort

(9) SW1400D
RcPiccoline Brugervejledning

## B. Fejlmeddelelser

De fejl, der kan opstå i RcComa180 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-udførsel.

   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.

   Opstår fejlen under kommandoudførsel, udskrives der intet linienummer.

3. Fejl fundet under disketteoperationer.

   Fejlmeddelelserne til disse har samme format som fejlmeddelelserne under pkt 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

**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**

**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 L$ := 7
```

**syntaks fejl**
RcComa180 har genkendt den første del af linien som en sætning, men de sidste tegn giver ingen mening, eller der mangler 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 1$ := "COMAL80
```

**'(' forventet**
Systemet forventede at møde en begyndelsesparentes på det sted, hvcor cursoren er placeret.

```
0010 DIM a,b(5)
```

**',' forventet**
Systemet forventede at møde et komma på det sted, hvor cursoren er placeret.

```
0010 OPEN FILE 1,"DATAFIL" READ
```

**':=' forventet**
Systemet forventede at finde et tildelingstegn på det sted, hvor cursoren er placeret.

```
0010 a=5
```

**':' forventet**
Systemet forventede at finde et kolon på det sted, hvor cursoren er placeret.

```
0010 1$(1,2,3):="TEKST"
```

**'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 TO 10 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 REF specificeres som parametre.

```
0010 PROC P(a(,))
```

**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 afsn 9.2).

```
WHILE i<=10 DO PRINT i
```

**ikke flere variable**
Systemet har ikke plads til flere variabelnavne.

**ikke implementeret**
Faciliteten er endnu 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 (FOR-NEXT,IF-ENDIF osv)**
Systemet kan ikke finde den tilhørende NEXT til FOR, ENDIF til IF osv.

**0097 : IKKE SAVE FIL**
LOAD-kommandoen er anvendt på en fil, der ikke indeholder et SAVEd program.

**0098 : IKKE IMPLEMENTERET**
Faciliteten er endnu ikke implementeret i systemet.

**0099 : SYSTEMFEJL**
I dette tilfælde bør man udfylde en RC FEJLMEDDELELSE, der beskriver fejlen.

**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) have værdien 100.

**0101 : ULOVLIG HELTALSVÆRDI**
Et sted, hvor systemet forventer et heltal, har en variabel en værdi større end 32768.

**0102 : LOG TIL NEGATIVT 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 COMAL80'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 : STAKOVERLØB**
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**
En etikette, en tekstvariabel og en numerisk variabel må ikke have det 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 PROC-ENDPROC,IF-ELSE-ENDIF, REPEAT-UNTIL, WHILE-ENDWHILE, 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**
Hvis man f.eks. indtaster tekst til en INPUT-sætning, der forventer et tal, og brugeren har ENABLE'd sin egen HANDLER, vil HANDLER'en blive kaldt, og SYS(0) 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 f.eks. 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 på Partner eller Piccoline hvis GSX ikke er loaded.

0124 : GSX DRIVER LOAD FEJL ELLER KUN INPUT DEVICE
Den refererede grafiske enhed findes ikke i systemet, eller den kan ikke benyttes til at tegne på.

## 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 det har forsøgt flere gange indenfor ca. 2 sekunder.

0207 : DISKETTE OFFLINE
Disketteenheden er f.eks. slukket.

0208 : DISKETTE ER BLEVET SKIFTET
Hvis man ønsker at skrive på en diskette, skal man først udføre en MOUNT kommando.

0209 : DISKETTE SKRIVEBESKYTTET
Disketten er blevet 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 : DISKETTE FULD**
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 (PRINTER ELLER PORT)**
Den ydre enhed 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 : FIL SKRIVEBESKYTTET (R/O eller PASSWORD)**
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).

/

## C. Ydre enheder

### C.1 Partner og Piccoline 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.

Udførslen af den ønskede kontrolfunktionen sker ved anvendelse af CHR$(x) i PRINT sætninger.

Følgende kontrolfunktioner er til rådighed:

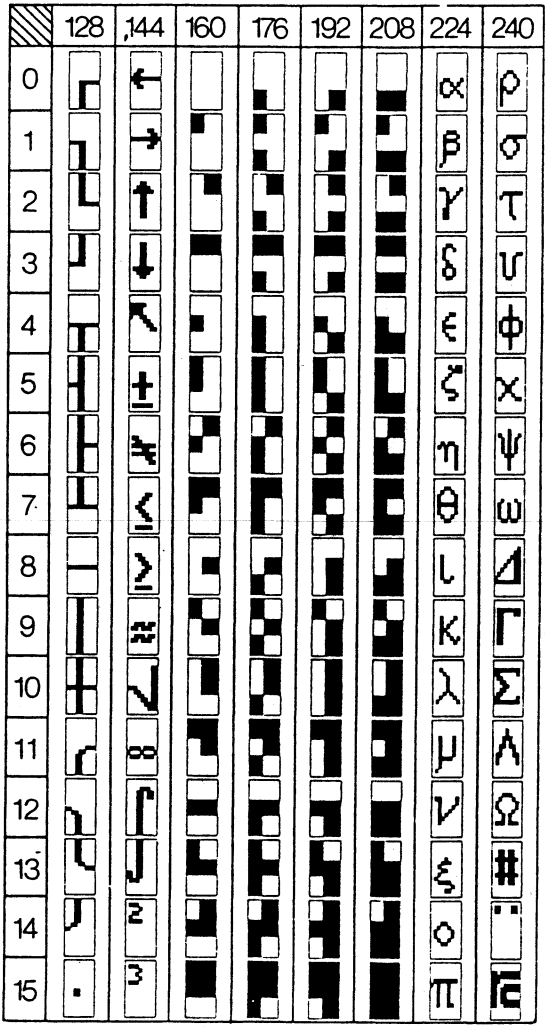
| X | CHR$(X) |
|---|---|
| 4 | Indsæt et tegn i linie |
| 5 | Slet et tegn i linie |
| 7 | 'BELL', dvs. hørbart signal |
| 8 | Cursor en position til venstre (backspace) |
| 9 | Tabulator |
| 10 | Linieskift (LF): cursor en position ned |
| 12 | Slet skærm (FF): cursor til position 1,1 |
| 13 | Vognretur (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 man sender 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 |

| | 0 | 16 | 32 | 48 | 64 | 80 | 96 | 112 |
|---|---|---|---|---|---|---|---|---|
| 0 | | § | | 0 | @ | P | ' | p |
| 1 | É | # | ! | 1 | A | Q | a | q |
| 2 | Ä | [ | " | 2 | B | R | b | r |
| 3 | Ö | \ | § | 3 | C | S | c | s |
| 4 | é | ] | $ | 4 | D | T | d | t |
| 5 | ä | ^ | % | 5 | E | U | e | u |
| 6 | ö | ' | & | 6 | F | V | f | v |
| 7 | | { | ' | 7 | G | W | g | w |
| 8 | | ¦ | ( | 8 | H | X | h | x |
| 9 | £ | } | ) | 9 | I | Y | i | y |
| 10 | | ~ | * | : | J | Z | j | z |
| 11 | Æ | | + | ; | K | Æ | k | æ |
| 12 | Ø | æ | , | < | L | Ø | l | ø |
| 13 | | ø | — | = | M | Å | m | å |
| 14 | Å | å | . | > | N | Ü | n | ü |
| 15 | Ü | ü | / | ? | O | _ | o | |

Figur C-1: Piccoline og Partner tegnsæt




Figur C-2 Piccoline og Partner tegnsæt

### C.1.1 Ændring af tegns udseende på skærmen

Tegn man skriver 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 |
| 115 | Start blink |
| 116 | Slut blink |
| 117 | Normal lysintensitet |
| 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 |
| 5 | Magenta (rød + blå) | |
| 6 | Gul (rød + grøn) | |
| 7 | Hvid (rød + blå + grøn) | |
| 8 | Grå | Lav intensitet |
| 9 | Kraftig lysende blå | |
| 10 | Kraftig lysende grøn | |
| 11 | Kraftig lysende cyan | |
| 12 | Kraftig lysende rød | Høj intensitet |
| 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

Ovenfor er i fig C-1 og C-2 vist standardtegnsættet til Partner og Piccoline

## C.2 Partner og Piccoline tastatur

Ved anslag leverer tastaturet et 7-bit tegn svarende til ASCII-tabellen i appendix E.

### C.2.1 Funktionstaster

Som beskrevet i kap 2.1 indeholder tastaturet en række taster, hvis funktion kan bestemmes af brugeren.

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 tasten 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 |
| Ins | 82 | CHR$(4) | Indsæt tegn |
| Del | 83 | CHR$(5) | Slet tegn |
| PRINT | 84 | | |
| SHIFT-A1 | 85 | | |
| SHIFT-A2 | 86 | | |
| SHIFT-A3 | 87 | | |
| SHIFT-A4 | 88 | | |
| 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 | | |
| SHIFT-F3 | 109 | | |

| Gravering | Kode | Standard-indhold | Forklaring |
|---|---|---|---|
| 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 på Partner og Piccoline

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
```

## C.4 IEEE-kort på Piccoline

iSBX 488 GPIB Interface Multimodule Board implementerer 1978 IEEE standard 488 instrumentbussen. Dette modul forbinder en Piccoline til en eller flere (op til 14) perifere enheder via GPIB.

For at få det fulde udbytte af dette modul fra et RcComa180 program, har brugeren direkte adgang til de enkelte registre på iSBX 488. Denne adgang foregår ved læsning og skrivning på nogle på forhånd udlagte HW-porte. Disse porte er beskrevet i nedenstående tabel. Læsning og/eller skrivning kan påbegyndes, efter man på sædvanlig måde har åbnet en strøm til den ønskede port.

Hvorledes selve styringen af iSBX 488 varetages, er beskrevet i Intel-manualen:

iSBX 488 General Purpose Interface Bus
(GPIB) Multimodule Board.
Hardware Reference Manual.
(Order number: 143154-001)

**HW-porte i Piccoline**

| Portnr | Læs | Skriv |
|---|---|---|
| 768 | Data In | Data Out |
| 770 | Interrupt Status 1 | Interrupt Mask 1 |
| 772 | Interrupt Status 2 | Interrupt Mask 2 |
| 774 | Serial Poll Status | Serial Poll Mode |
| 776 | Address Status | Address Mode |
| 778 | Command Pass Through | Aux Mode |
| 780 | Address 0 | Address 0/1 |
| 782 | Address 1 | EOS |
| 784 | Jumper Status | |
| 788 | Interrupt Mask | Interrupt Mask |
| | Error Flag | |
| | Error Mask | Error Mask |
| | Event Counter Status | Event Counter |
| | Time Out Status | Time Out |
| | Controller Status | |
| | GPIB Status | |
| 790 | Interrupt Status | Command Field (X) |

(X) Kommandoer til Command Field registret udpeger hvilken af de til port 788 hørende instruktioner, man ønsker udført.

## C.5 Piccoloens ydre enheder

### C.5.1 Piccolo Skærmstyring

Det er ved hjælp af kontroltegn muligt at udføre forskellige former for skærmstyring. Typiske eksempler kan være sletning af skærm, linieskift samt direkte styring af cursoren.

Udførelsen af den ønskede kontrolfunktion sker ved anvendelse af CHR$(X) i PRINT sætninger.

Følgende kontrolfunktioner er til rådighed:

X CHR$(X)
5 Slet et tegn i linie
7 'BELL', dvs. hørbart signal (ikke RC701)
8 Cursor en position til venstre (back space)
9 Indsæt et tegn i linie
10 Linieskift (LF): cursor en position ned og hen på første position på linien
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 skærmattribut
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 på skærmbilledet (EOF)

### C.5.1.1 Blink og invers skrift

Dele af skærmbilledet kan bringes til at blinke og/eller fremtræde som invers skrift (invers video). Dette gøres ved at sende CHR$(27) efterfulgt af bestemte kontroltegn større end 128, hvorefter uddata fra de efterfølgende PRINTsætninger vil se ud som specificeret ved de anvendte kontroltegn.

CHR$(27)+CHR$(128) sætter skærmstyringen tilbage til normal skrift

X CHR$(X)
128+ 2=130Blink
128+ 4=132Aktivering af semigrafisk tegnsæt
128+16=144Invers skrift
128+32=160Understregning
128 Normal skrift

**Eksempel**
0030 PRINT CHR$(27);CHR$(130);"RC700";CHR$(27);CHR$(128)

**Bemærkning**
Teksten RC700 vil blinke

De forskellige faciliteter kan anvendes sammen, således vil f.eks. X=128+2+16 bevirke blinkende, invers skrift.

Bemærk, at kontroltegnet i sig selv fylder en position (en blank) på skærmen. Kontroltegnet vil kun have virkning, så længe det befinder sig på skærmen, det vil sige, at virkningen f.eks. ophører, hvis billedet "ruller", og tegnet dermed forsvinder.

Hvis man benytter variable til at angive de forskellige værdier, vil det øge programmets overskuelighed, f.eks.

**Eksempel**

```
0010 blink:=2; invers:=16
0020 PRINT
CHR$(27);CHR$(128+blink+invers);"RC700";CHR$(27);CHR$(128)
```

### C.5.1.2 Semigrafisk tegnsæt

Skærmen indeholder mulighed for udskrivning af et semigrafisk tegnsæt. Denne mulighed kan f.eks. anvendes til kurvetegning, stolpediagrammer og simplere grafisk afbildning.

Det semigrafiske tegnsæt aktiveres ved brug af kontrolsekvensen CHR$(27)+CHR$(132). Returnering til normalt tegnsæt sker ved brug af kontrolsekvensen CHR$(27)+CHR$(128).

**Eksempel**

```
0010 PRINT CHR$(27);CHR$(132);"RC700";CHR$(27);CHR$(128)
```

Figur C.2 er en komplet fortegnelse over skærmens semigrafiske tegnsæt.

### C.5.1.3 Skærm - tegngenerering og konvertering

Ved udskrift til skærmen sendes et 8-bit tegn til en procedure, der skriver ud på skærmen. Tegnet konverteres her via CP/Ms outputkonverteringstabel.

Ønsker man ikke at få konverteret tegnet, skal man benytte CHR$-værdier over 128.

Man kan vælge mellem to tegnsæt:

- det almindelige alfanumeriske tegnsæt
- det semigrafiske tegnsæt

Man kan skifte mellem de to tegnsæt som beskrevet i afsnit C.4.1.2. De to tegnsæt fremgår af fig. C-3 og C-4.

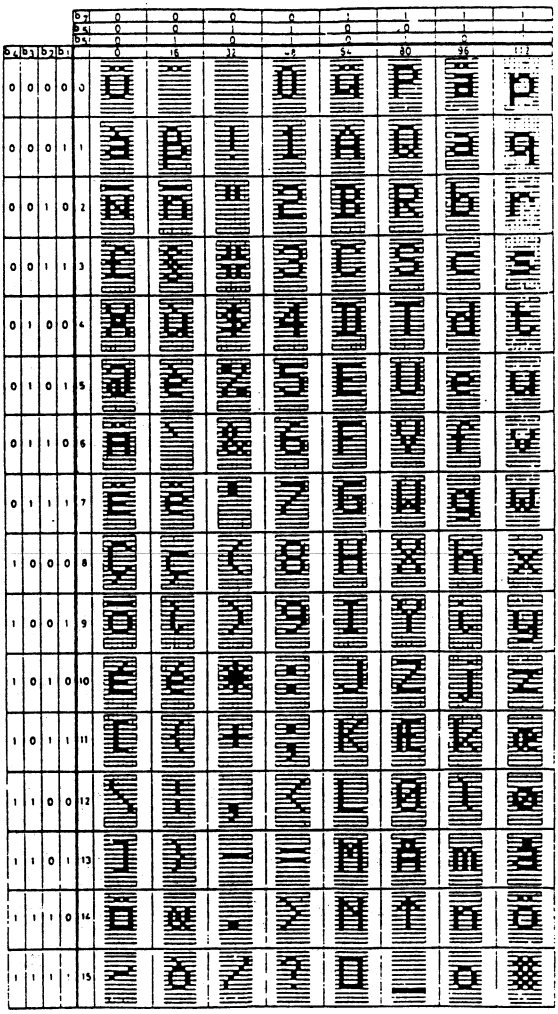
Bemærk at CHR$-værdier i intervallet 0-31 er forbeholdt styretegn til skærmen, hvorfor det tilsvarende værdiområde i tegnsættene må genereres uden konvertering.

**Eksempel**

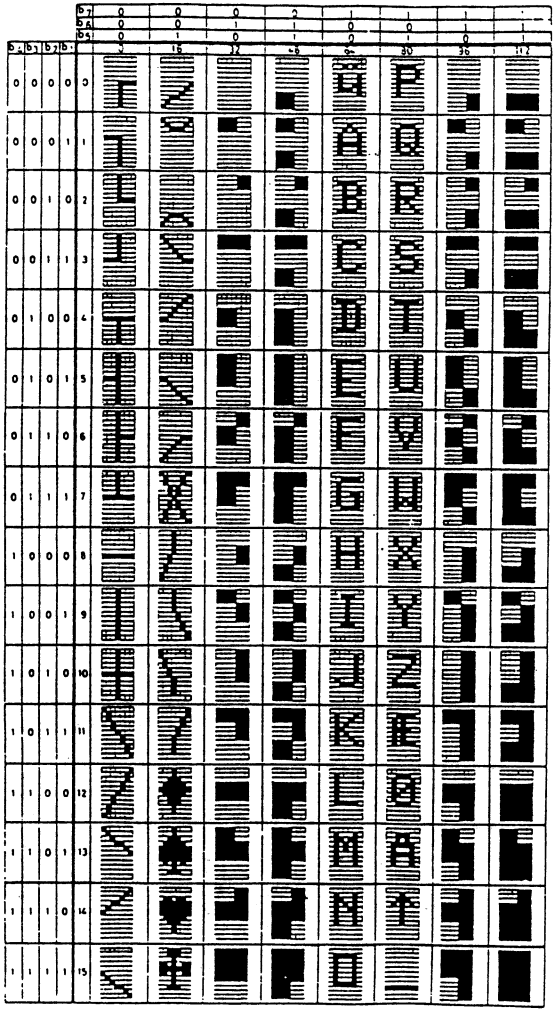
```
0010 PRINT CHR$(128+11);CHR$(128+13)
RUN

END
AT 0010
```

Det er specielt vigtigt at huske at lægge 128 til CHR$-værdien i forbindelse med semigrafisk udskrift.

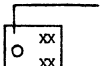
Benytter man SCREEN$, arbejder man direkte i selve skærmbufferen, og man får aldrig konverteret tegnene. I forbindelse med SCREEN$ fungerer CHR$-værdier over 128 som de kontroltegn, der er beskrevet i afsnit C.4.1.1.



Figur C-3: Piccolo alfanumerisk tegnsæt.



Figur C-4: Piccolo semigrafisk tegnsæt.




tastatur gravering.

- værdi fra tastatur , når 'SHIFT' anvendes
- værdi fra tastatur , når 'SHIFT' ikke anvendes
- værdi fra tastatur , når 'CTRL' anvendes (hvis defineret)

ændret.

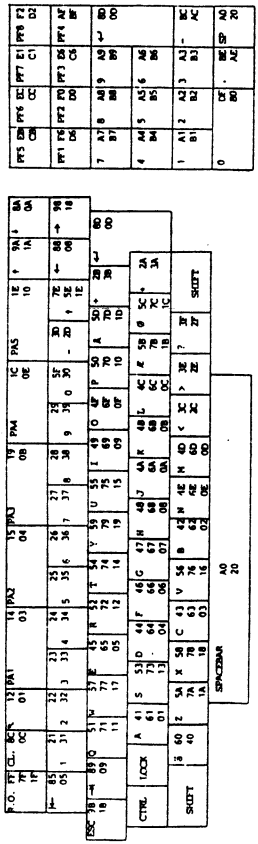


FIG.C.1: RC721/22 tastatur i dansk udgave (serienummer 0-50/383). Hexadecimal notation.

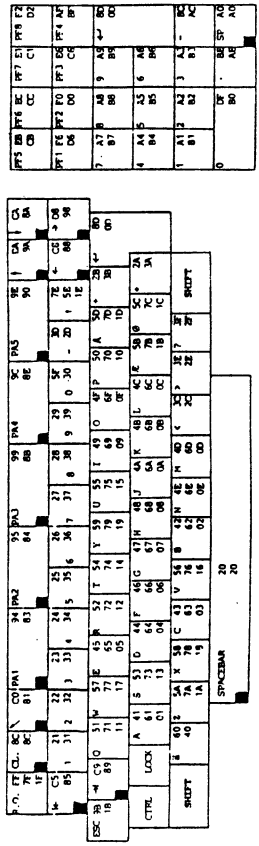


FIG.C.4: RC721/22 tastatur i dansk udgave (serienummer 51/384- ). Hexadecimal notation.

### C.5.2 Tastatur - tegngenerering og konvertering

Ved anslag leverer tastaturet et 8-bit tegn (256 forskellige muligheder) som vist i fig. C.5 og C.6. Dette tegn konverteres herefter til et nyt 8-bit tegn ved hjælp af inddata konverteringstabellen.

Konverteringstabellen for både inddata og uddata vælges med CP/M programmet CONFI. Hvorledes man kan ændre konverteringen for de enkelte tegn, er beskrevet i (3) og (4).

## C.6 Piccolo HW-porte

Nedenfor er angivet en oversigt over der fysiske portnumre for RC702. Det skal understreges, at disse porte skal anvendes med den allerstørste forsigtighed, da RcComa180-systemet ikke yder nogen form for beskyttelse.

Om brugen af portene henvises til leverandørernes tekniske manualer.

| Navn | Komponenttype | Port | Kommentar |
|---|---|---|---|
| Skærm | I8275 (Intel) | 0 | Data |
| | | 1 | Kontrol |
| Diskette | uPD765 (NEC) | 4 | Kontrol |
| | eller I8272(Intel) | 5 | Data |
| Seriel | Z80SIO2 (Zilog) | 8 | Data, Terminalport |
| I/O | | 9 | Data, Printerport |
| | | 10 | Kontrol, Terminalport |
| | | 11 | Kontrol, Printerport |
| CTC | Z80-CTC | 12 | Kanal 0, til SIO term |
| | | 13 | Kanal 1, til SIO printer |
| | | 14 | Kanal 2, int displ |
| | | 15 | Kanal 3, int flop |
| Parallel | Z80-PIO | 16 | Data, Tastatur |
| I/O | | 17 | Data, Parallel port |
| | | 18 | Kontrol, Tastatur |
| | | 19 | Kontrol, Parallel port |
| Intern switch | | 20 | 8 bit fra switchen |
| | | 22 | Motor kontrol floppy |
| DMA | AM9517A-4 | 240- | |
| | I3237-2 | 255 | |

**Eksempel**
Følgende to procedurer udskriver eller indlæser et tegn på/fra en port.

```
0010 PROC pout(port,d)
0020   OPEN FILE 5,"/"+STR$(port)+"/PORT", WRITE
0030   marg:=SYS(6) // Gem MARGINs værdi
0040   MARGIN 0
0050   PRINT FILE 5:CHR$(d);
0060   CLOSE FILE 5
0070   MARGIN marg // Reetabler MARGINs værdi
0080 ENDPROC pout
0090
0100 PROC pin(port,REF d)
0110   OPEN FILE 5,"/"+STR$(port)+"/PORT", READ
0120   d:=ORD(GET$(5,1))
0130   CLOSE FILE 5
0140 ENDPROC pin
```

## C.7 Styring af skrivere

Dette afsnit indeholder en introduktion til en række skriveres faciliteter. For en fuldstændig beskrivelse henvises til de enkelte skrivermanualer.

### C.7.1 RC861-skriveren

RC861-skriveren har 3 forskellige skrifttyper:

- komprimeret ( 132 tegn pr. linie)
- normal ( 80 tegn pr. linie)
- elongeret ( 40 tegn pr. linie)

Yderligere indeholder printeren mulighed for udskrift med et semigrafisk tegnsæt.

De forskellige faciliteter aktiveres ved hjælp af CHR$-funktionen anvendt i PRINT-sætninger:

| X | CHR$(X) |
|---|---|
| 14 | Aktivering af semigrafisk tegnsæt |
| 15 | Aktivering af almindeligt alfanumerisk tegnsæt |
| 29 | Udskrift med komprimeret skrift |
| 30 | Udskrift med normal skrift |
| 31 | Udskrift med elongeret skrift |

Når man har specificeret et tegnsæt og/eller en skriftstype, vil skriveren skrive som angivet, indtil brugeren skrifter tegnsæt og/eller skriftstype ved at udskrive nye kontroltegn. Derfor er det en god ide at afslutte et program med sætningen

```
PRINT CHR$(15);CHR$(30)
```

således, at skriveren "starter" rigtigt, når den næste gang benyttes.

### C.7.2 RC862,RC867 skriveren

RC862 og RC867 skriveren har en række faciliteter, der kan styres fra et program ved hjælp af CHR$-funktionen.

**Skriftstyper**

| | |
|---|---|
| CHR$(29) | Komprimeret skrift |
| CHR$(30) | Normal skrift |
| CHR$(29)+CHR$(31) | Lidt elongeret skrift |
| CHR$(30)+CHR$(31) | Elongeret skrift |

**Liniemellemrum**

| | |
|---|---|
| CHR$(27)+CHR$(54) | 6 linier pr tomme |
| CHR$(27)+CHR$(56) | 8 linier pr tomme |

**Tegnsæt**

CHR$(14) Semigrafisk tegnsæt
CHR$(15) Almindeligt alfanumerisk tegnsæt

**Formularkontrol**

| | |
|---|---|
| CHR$(12) | Fører papiret frem til første linie på ny side (TOF : Top Of Form) |
| CHR$(27)+CHR$(53) | Sætter TOF til nuværende position |
| CHR$(27)+CHR$(70)+"nn" | Sætter sidehøjden til nn linier |
| CHR$(27)+CHR$(11)+"nn" | Laver nn linieskift |

**Diverse**

CHR$(19) Afmelder skriveren (slukker SEL lampen)
CHR$(24) Sletter skriverbufferen

## D. Installering af RcComa180

### D.1 Partner og Piccoline installation

RcComa180-systemet distribueres på en enkelt 5 1/4 " diskette. På denne diskette ligger følgende:

| | |
|---|---|
| COMAL80.CMD | Selve RcComa180-fortolkeren |
| COMAL80.ERM | RcComa180 fejlmeddelelser. |
| GENERR.CSV | RcComa180 SAVE-program til oprettelse af fejlmeddelelsesfil. |
| INSTJOB.SUB | Job-fil til installering af RcComa180 |

Bemærk at disketten ikke indeholder styresystemet CCP/M. Før man kan benytte RcComa180, skal systemet kopieres over på et CCP/M system.

Fremgangsmåden er følgende:

1. Start Partner eller Piccoline op på sædvanlig måde
2. Tryk ESC til hovedmenuen.
   Spørgsmål: Ok at vende tilbage til TMP (j/n)
3. Tast: j
   Nederst på skærmen skrives f.eks.:
   A>

4a. Hvis Partner eller Piccoline har to diskettestationer, sættes RcComa180 distributionsdisketten i diskettestation 2.
   Skriv derefter: B:<retur>
   Svar: B>
   Skriv SUBMIT INSTJOB A

4b. Hvis Partner har en diskettestation og en winchesterdisk, sættes RcComa180 distributionsdisketten i diskettestationen.
   Skriv derefter: A:<retur>
   Svar: A>
   Skriv SUBMIT INSTJOB B

5. Herefter kopieres indholdet af RcComa180 disketten over på CCP/m systemdisken, og det er muligt at starte RcComa180 op.

### D.2 Piccolo installation

RcComa180-systemet distribueres på en enkelt 8" eller 5 1/4" diskette. På denne diskette ligger følgende:

| | |
|---|---|
| COMAL80.COM | Selve RcComa180-fortolkeren |
| COMALCNV.COM | Program til omdannelse af Stand-Alone COMAL80 disketter til CP/M disketter. |
| SYSTEXT.C80 | RcComa180 fejlmeddelelser |
| ERRMSG | RcComa180 SAVE-program til oprettelse af fejlmeddelelsesfil. |

Bemærk at disketten ikke indeholder styresystemet CP/M. Før man kan starte RcComa180 op, skal systemet kopieres over på et CP/M system.

På et RC700 system med to diskettestationer gøres det på følgende måde:

1. Sæt CP/M diskette i diskettestation 1
2. Sæt RcComa180 diskette i diskettestation 2
3. Tryk på RESET-knappen
4. CP/M startes op:...
   A>
5. Skriv PIP A:=B:*.*
6. Herefter kopieres indholdet af RcComa180 disketten over på CP/M disketten, og det er muligt at starte RcComa180 op.

På et RC700 system med een diskettestation flyttes de 4 filer enkeltvis ved hjælp af programmet TRANSFER (se 3 eller 4).

**Eksempel**

```
A> TRANSFER vognretur
TRANSFER UTILITY VERS 2.0 83.-01.05
SOURCE DISK TYPE (SS:=1, DD:=2): 2 vognretur
DESTINATION DISK TYPE (SS:=1, DD:=2): 2 vognretur
SOURCE FILENAME: COMAL80.COM vognretur
DESTINATION FILENAME: COMAL80.COM vognretur
INSERT SOURCE DISK AND TYPE RETURN vognretur
INSERT DESTINATION DISK AND TYPE RETURN vognretur
INSERT CP/M SYSTEM DISK AND TYPE RETURN vognretur
TRANSFER COMPLETE
A>
```

Det fremhævede tastes af brugeren.

### D.2.1 COMALCNV

For at muliggøre overflytning af programmer skrevet under Stand-Alone COMAL80 til RcComa180 ligger der på systemdisketten et program, der er i stand til at ændre katalogformatet.

Før man begynder at flytte filer, skal man være opmærksom på,

- at SAVE-formatet er ændret, så alle SAVE-programmer skal listes ud på stand-alone COMAL80-disketten
- at man overholder CP/Ms filnavnekonventioner (bl.a. kun store bogstaver i filnavne)

Når man har sikret sig dette, kan man overflytte disketteindholdet til en CP/M diskette.

**Eksempel**

A>COMALCNV vognretur
Coma1 til CP/M konvertering vers. 83.05.16

Coma180 format på enhed (A/B): B vognretur
CP/M format på enhed (A/B/C): A vognretur

Indsæt Coma180 diskette i disketteenhed B og tast <vognretur>
vognretur
Indsæt CP/M diskette og tast <vognretur> vognretur
...
Indsæt system diskette og tast <vognretur> vognretur
A>

Med dette programkald får man omdannet en Stand-Alone COMAL80 diskette i diskettestation 2 til en CP/M diskette i diskettestation 1.

**Bemærk**

- CP/M disketten skal være formatteret.
- Disketten der får CP/M format bliver overskrevet fuldt ud, hvorfor eventuelt indhold på den bliver slettet,
- Det er lovligt at lade den samme diskette optræde som COMAL80 diskette og CP/M diskette, men hvis der bliver fundet fejl under omdannelsen kan hele indholdet være gået tabt.

## E. 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 |

## F. Store programeksempler

Dette afsnit indeholder en række store programeksempler, der skulle illustrere nogle af de mange muligheder man har for programmering i RcComa180.

### F.1 Enarmet tyveknægt

Det første store programeksempel er et spil. Programmet fungerer ligesom en enarmet tyveknægt, bortset fra, at det er lidt for rundhåndet med gevinster.

**Bemærk** programmet kan kun udføres på Piccolo.

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; kløer:=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 kløer
1010     IF t2=kløer AND t3=kløer 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:=CRD(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.2 Tænker du på et dyr ?

Andet programeksempel er også 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$(100) OF 30//                   tekstabel
0030 DIM træ(100,3)//                              matrix
0040 ophav:= 3; venst:= 1; højre:= 2; max:= 2; dommedag:= FALSE
0050 spørgsmål$(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
```

## F.3 Tænker du på et dyr ? (filudgave)

Tredje programeksempel svarer til det andet, 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
```

## F.4 Data-indtastning

Her er et eksempel på en extern procedure til et procedure-bibliotek.

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, vognreturtasten og de almindelige taster. Man vender tilbage til hovedprogrammet ved at trykke på I<-- -tasten.

Hvis det lyder lidt indviklet så kør 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
```

## F.5 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
```

## F.6 Semigrafik

Piccolo har ikke altid fuld grafik, men dog altid de semigrafiske tegn, der er beskrevet i appendix C. Ved hjælp af disse kan man dog lave ganske pæne kurver og søjlediagrammer. Procedurerne i dette afsnit opdeler skærmen i 158 tegn på tværs (x-værdien) og 75 tegn i højden (y-værdien).

Procedurerne har følgende virkefelter:

PROC setdot(x,y) "tænder" punktet (x,y)

PROC drawline(x0,y0,x1,y1) tegner en "linie" fra (x0,y0) til (x1,y1)

PROC init sætter skærmen i semigrafisk mode, og initialiserer de nødvendige variable

PROC move,moveto,turnto,turn fungerer ligesom plotterkommandoer, idet man angiver hvorledes "pennen" skal bevæge sig i forhold til aktuel position.

PROC move(x) tegner en linie, x enheder lang i aktuel retning

PROC moveto(x0,y0) tegner en linie fra aktuel position til (x0,y0).

PROC turnto(d) ændrer aktuel retning til d grader

PROC turn(d) ændrer aktuel retning med d grader

```
0010 PROC setdot(x,y)
0020   y:=74-y
0030   iii:=(y DIV 3)*80+x div 2+1
0040   c$:=SCREEN$(iii:)
0050   bit:=tabel((y MOD 3)*2+x MOD 2+1)
0060   c$:=CHR$(ORD(c$) MOD bit+bit+(ORD(c$) DIV (bit*2))*bit*2)
0070   SCREEN$(iii:):=c$
0080 ENDPROC setdot
0090
0100 PROC drawline(x0,y0,x1,y1)
0110   EXEC setdot(x0,y0)
0120   CASE TRUE OF
0130   WHEN ABS(x1-x0)<ABS(y1-y0)
0140     dy:=(x1-x0)/ABS(y1-y0)
0150     x:=x0
0160     FOR y:=y0 TO y1 STEP SGN(y1-y0) DO
0170       x:=x+dy
0180       EXEC setdot(x,y)
0190     NEXT y
0200   WHEN ABS(x1-x0)>ABS(y1-y0)
0210     dx:=(y1-y0)/ABS(x1-x0)
0220     y:=y0
0230     FOR x:= x0 TO x1 STEP SGN(x1-x0) DO
0240       y:=y+dx
0250       EXEC setdot(x,y)
0260     NEXT x
0270   OTHERWISE
0280     y:=y0
0290     FOR x:=x0 TO x1 STEP SGN(x1-x0) DO
0300       y:=y+SGN(y1-y0)
0310       EXEC setdot(x,y)
0320     NEXT x
```

```
0330   ENDCASE
0340 ENDPROC drawline
0350
0360 PROC moveto(xkor,ykor)
0370   IF NOT penup THEN
0380     EXEC drawline(oldx,oldy,xkor,ykor)
0390   ENDIF
0400   oldx:=xkor; oldy:=ykor
0410 ENDPROC moveto
0420
0430 PROC move(x)
0440   xkor:=oldx+x*xangle; ykor:=oldy+x*yangle
0450   IF NOT penup THEN
0460     EXEC drawline(oldx,oldy,xkor,ykor)
0470   ENDIF
0480   oldx:=xkor; oldy:=ykor
0490 ENDPROC move
0500
0510 PROC turnto(d)
0520   deg:=d
0530   xangle:=COS(deg*pi/180); yangle:=SIN(deg*pi/180)
0540 ENDPROC turnto
0550
0560 PROC turn(d)
0570   deg:=(deg+d) MOD 360
0580   xangle:=COS(deg*pi/180); yangle:=SIN(deg*pi/180)
0590 ENDPROC turn
0600
0610 PROC init
0620   SCREEN$:=CHR$(132)
0630   DIM c$ OF 1,tabel(6)
0640   FOR i:=1 TO 6 DO READ tabel(i)
0650   DATA 1,2,4,8,16,64
0660   oldx:=1; oldy:=1; pi:=ATN(1)*4
0670   deg:=0; xangle:=1; yangle:=0; penup:=FALSE
0680   MARGIN 0
0690 ENDPROC init
0700
```

## F.7 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 denne næste tilstand (generation). Således fortsættes et antalgange, enten 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 mat1ligmat2
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 mat1ligmat2
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 mat1ligmat2
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
```

## F.8 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 CHRS(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
```

## F.9 Eksempel fra procedureafsnittet

Som lovet i afsnit 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 "elevoplyn"
0110 DELETE "elevdata"
0120
0130 postlængde:=klasselgd+2+navnlgd+2+adresselgd+2+postnrlgd+2
0140 CREATE "elevdata",postlængde*maxelev/1024
0150 OPEN FILE 1,"elevoplysn", 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,"elevoply", 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 "elevoply"
1190   RENAME "nyeoplys","elevoply"
1200   PRINT "Slut på kartoteksprogram"
1210 ENDPROC afslut
```

## F.10 Elektronisk orgel

Ved hjælp af Partneren og Piccolinens 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 CLOSE FILE sound_stream
0800 silence
```

## G. Automatisk opstart af programmer

Ved opstart er det muligt, at få et program indlæst og udført automatisk, hvis programmet gemmes (SAVE) under navnet "LOGON".

Når RcComa180 startes op ledes efter en fil med navn "LOGON" på den CP/M unit, der er startet op fra (dvs normalt unit A). På Partner og Piccoline ledes efter en fil med navnet "LOGON.CSV". Findes filen ikke, udskrives "RcComa180 rev x.xx" og systemet er klar til indtastning.

Findes filen derimod, indlæses programmet og startes umiddelbart.

Ønsker man at loade et andet program end "LOGON" kan man angive filnavnet i kaldet af RcComa180, f.eks.

COMAL80 MENU

I dette tilfælde vil RcComa180 forsøge at LOADe og udføre en fil med navnet "MENU".

## Stikordsregister


" 29
' 27
* 27
+ 27
+ (strenge) 31
- 27
/ 27
// 18
: (strenge) 31
:= 26,239
< 33
<= 33
<> 33
= 33
> 33
>= 33

A
ABS 27,87
Afbrydelse af programudførsel 22
Afslut grafik 104
Afvikling af programmer 222
AND 88
APPEND, OPEN-sætning 51,185
ASCII 285
AT 89
ATN 27,91
AUTO 20,93
Automatisk linienummerering 20

B
BASIC 11
Beregningspræcision 28
Beregningsudtryk 26
Betingede sætninger 37,160,161,162
Bordkalkulator 77
BYE 80,95

C
CASE 38
CASE-WHEN-ENDCASE 38,96
CHAIN 98
CHR$ 99

CIRCLE 100
Cirkel 100
CLEAR 102
CLOSE 103
CLOSE FILE 52
CLOSE GRAPHICS 104
COMAL80-kerne 11
COMALCNV 282
CON 22,105
CONTINUE 71,106
COPY 79,107
COS 27,108
CREATE 53,109
CSV 49
Cursor 13
Cursoradressering 89

D
DATA 111
Datafiler 50
Datastrømme 48
DATE$ 113
Dato, dags 113
Decimalangivelse 25
DEL 21,114
DELETE 115
Delstrenge 31
DIM 117
DIR 78,119
DISABLE 121
Disk 47
DIV 27,122
DRAW 123
DRAWTO 125

E
EDIT 75,126
Eksponentiel notation 25
ELSE 38
ENABLE 127
END 128
End-Of-File 131
ENDIF 37
Enheder, ydre 55
ENTER 49,129
EOD 130
EOF 131

Erklæring af strengvariable 117
Erklæring af taltabeller 57,117
Erklæring af teksttabeller 59,117
ERR 71,133
ERRTEXT$ 134
ESC 15,17,21
Etikette 135
EXEC 62,136
EXP 27,137
Externe procedurer 69

F

F1 - F12 15
FALSE 138
Fejl, Ind- og ud-læsning 260
Fejl, kommando 257
Fejl, programindtastning 254
Fejl, udførelses- 257
Fejlmeddelelser 18,253
Fejlnummer 71,133
Fejltekst 134
Fil 47
Filer, direkte tilgang 50,53
Filer, indlæsning 167,207
Filer, kopiering 107
Filer, lukning 103
Filer, omdøbning 56,208
Filer, oprettelse 109
Filer, oversigt over 119
Filer, sekventielle 50,51
Filer, sletning 56,115
Filer, tegnvis indlæsning 146
Filer, udskrivning 248
Filnavn 47
FOR 41,139
FOR-NEXT 42,141
Formatteret udskrift 197,198
FUNC-ENDFUNC 142
FUNC-EXTERNAL 144
Funktioner 61
Funktioner, brugerdefinerede 72,142
Funktioner, externe 144
Funktioner, indbyggede 27
Funktioner, returnering 219
Funktionstaster, Partner og Piccoline 15,267

G

GET$ 55,146
GLOBAL 67,148
Globale variable 67,148
GOTO 149
GPARM 150
Grafik 81
Grafik, afslut 104
Grafik, indlæsning 175
Grafik, opstart af 81,187
Grafik, Partner og Piccoline 13,81
Grafik, Piccolo 13,81
Grafik, polylinie 152
Grafik, teghøjde 152
Grafik, tekstudskrift 238
Grafik, udfyldt polygon 152
Grafik, valg af farve 191,192
Grafik, vindue 246
GRAPHICS 13
GSX 13,150,151
GSX, Direkte kald 151

H

HANDLER 70,203
HANDLER, aktivering 127
HANDLER, deaktivering 121
HANDLER, returnering 106,203,218,219
Heltalsdivision 122
HW-porte 55
HW-porte, Piccoline 271
HW-porte, Piccolo 278

I

IEEE-kort, Piccoline 270
IF, simpel 37
IF, udvidet 37
IF-THEN 37,160
IF-THEN-ELSE-ENDIF 38,162
IF-THEN-ENDIF 37,161
IMPORT 68,164
IN 34,165
Indentifikator 23
Indlæsning 45
Indlæsning, fil 167
Indlæsning, oversigt 56
Indlæsning, tastatur 45,166
INPUT 166

INPUT FILE 52,167
Installation, Piccolo 281
INT 27,169

K
Kald af procedurer 136
Kassettebånd 47
Katalogudskrift 119
KEY$ 170
Kommandoer 75,76
Kommandoer, datastrømme 78
Kommandoer, disk 78
Kommentar 18
Konstanter, streng- 29
Konstanter, tal- 26
Kontrolstrukturer 37
Kædning af programmer 98

L
Label 135
LEN 30,171
Linienummer 18
Linienummerering, automatisk 20
LIST 49,172
LOAD 49,174
LOCATE 175
LOG 27,177
Logiske udtryk 32
Lukkede procedurer 66
Lydkreds 269,310
Løkkestrukturer 41,139,141,211,243,244

M
MARGIN 178
MOD 27,179
MOUNT 78,180
MOVE 181
MOVETO 182
Multiforgreninger 38,96

N
Navn, fil 47
NEW 20,183
NOT 184
Notation, eksponentiel 25

Numeriske udtryk 34
Numeriske udtryk, prioritet 35
Numeriske variable 25
nvar 25

O
Omdøbning af filer 56
Omnummerering, programlinier 21
OPEN 51,54,185
OPEN GRAPHICS 187
Open Workstation 150
OR 188
ORD 189
OUT 190,225

P
PALETTE 191
Parameteroverførsel 63
PENCOLOR 192
PI 193
Poststørrelse 53
PREFIX 194
PRINT 195
PRINT FILE 52,196
PRINT FILE USING 197
PRINT USING 198
PROC-ENDPROC 62,200
PROC-EXTERNAL 202
PROC-HANDLER 70,203
Procedurebibliotek 70
Procedurer 61,200
Procedurer, externe 69,202
Procedurer, kald af 136
Procedurer, lukkede 66
Procedurer, oversigt 66
Procedurer, parametre 63
Procedurer, rekursive 63
Procedurer, returnering 219
Procedurer, simple 62
Program, indtastning 18
Programafslutning 128
Programhop 149
Programindtastning, kommandoer 75
Programindtastning, rettelser 75,126
Programlageret 18,229
Programlinier 18
Programlinier, omnummerering 21,209

Programlinier, Slet 21
Programlinier, sletning 114
Programmer, afvikling af 222
Programmer, fejlfinding 77
Programmer, indlæsning 49,129,174
Programmer, Kædning 98
Programmer, lagring 49
Programmer, sletning 183
Programmer, størrelse 76,229
Programmer, udskrivning 49,172,195,196,197,198,223
Programudførsel 22
Præcision, beregnings- 28

R
RANDOM, OPEN-sætning 185
RANDOMIZE 204
RC861 279
RC862 280
RC867 280
RcComa180, afslut 80,95
RcComa180, nøgleord 85
READ 206
READ FILE 52,54,207
READ, OPEN-sætning 51,185
REF 64
Regneregler 28
Regnetegn 27
Rekursive procedurer 63
RENAME 208
RENUMBER 21,209
REPEAT-UNTIL 43,211
Reserverede ord 23
RESTORE 216
RETRY 71,218
Retur 15
RETURN 71,219
RND 220
RUN 22,222

S
Sammenligninger 33
Sammenligningsoperatorer 33
SAVE 49,223
SCREEN$ 224
SELECT OUTPUT 190,225
SGN 27,226
Simple procedurer 62

SIN 27,227
SIZE 76,229
Skærm, slet 102
Skærmbillede 224
Skærmstyring, Partner og Piccoline 263
Skærmstyring, Piccolo 271
Slet programlinier 21
Sletning af filer 56
SOUND 269,310
SQR 27,230
STOP 231
STR$ 232
Strenge 25
Strenge, del- 31
Strenge, sammensætning 31
Strengfunktioner 73
Strengkonstanter 29
Strengvariable 29
Strengvariable, erklæring 29
Strengvariable, erklæring af 117
SYS 235
Sætninger, betingede 37,160,161

T
TAB 236
Tabulering 236
Tal, notation 26
Taltabelkomponenter 57
Taltabeller 57
Taltabeller, erklæring af 57,117
TAN 27,237
Tastatur, Partner og Piccoline 14,267
Tastatur, Piccolo 16,278
Tegnsæt, Partner og Piccoline 267
Tekster 25
Teksttabelkomponenter 58
Teksttabeller 57
Teksttabeller, erklæring af 59,117
TEXT 238
Tildeling 25,239
Tilfældige tal 204,220
TIME$ 240
TRUE 241
Tællesløjfer 41

U
Udskrivning 45

Udskrivning, oversigt 56
Udskrivning, skærm 46
Udtryk, beregnings- 26
Udtryk, logiske 32
Udtryk, numeriske 34

V
VAL 242
Variable, globale 148
Variable, importerede 164
Variable, streng- 29
Variable, tekst- 29

W
WHILE 44,243
WHILE-ENDWHILE 244
WINDOW 246
WRITE FILE 52,54,248
WRITE, OPEN-sætning 51,185

Y
Ydre enheder 55,263
Ydre enheder, navne 48

Z
ZONE 249

LÆSERBEMÆRKNINGER

Titel: RcComa180, Brugervejledning RCSL Nr.: 99-1 09997

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!

..... Fold her .....

..... Riv ikke - Fold her og hæft .....

Frankeres
som
brev

REGNECENTRALEN
af 1979
Informationsafdelingen
Lautrupbjerg 1
2750 Ballerup