# D D E - C O M A L - 8 0

Referencehåndbog

Dansk Data Elektronik ApS
1 juli 1982

Forfatter: Anders Ansted

Indholdsfortegnelse Side

## 1. Indledning

Denne referencehåndbog definerer sprogsyntaks og systemordrer for version 1 af DDE-COMAL-80 systemet implementeret på SPC/1 mikrodatamatsystemerne fra Dansk Data Elektronik ApS (DDE).

Håndbogen er skrevet for personer, der på forhånd har kendskab til COMAL-80 (fx gennem deltagelse i et DDE kursus eller gennem studier af lærebøger om COMAL-80), eller som har et grundigt forkendskab til programmering i andre højniveau sprog. Referencehåndbogen er uegnet som lærebog i programmeringssproget COMAL-80.

COMAL-80 er fremkommet som en standardisering af programmeringssproget COMAL, og er udarbejdet af en abejdsgruppe, der omfattede mikrodatamatfabrikanter, undervisere samt personer fra de højere læreanstalter. COMAL-80 er defineret i tre dele: et forslag til en kerne, et forslag til udvidelser samt et forslag til filsystem. DDE-COMAL-80 version 1 indeholder arbejdsgruppens forslag til en kerne for COMAL-80 tillige med mange kendte faciliteter fra DDE-COMAL og DDE-COMAL-B. Endelig benytter DDE-COMAL-80 det samme filsystem, som kendes fra DDE-COMAL-B.

Litteratur om COMAL-80:

COMAL 80 Programming Language - proposal, arbejdsgruppen, 25.03.1980

COMAL 80 Recommandation for Extensions, arbejdsgruppen, 10.04.1980

COMAL 80: File System Extension, arbejdsgruppen, 04.03.1980

Boldklubben, et tilfælde af administrativ databehandling, Børge R. Christensen, DATO

## 2. Forskelle mellem DDE-COMAL og DDE-COMAL-80

De vigtigste forskelle mellem DDE-COMAL og DDE-COMAL-80 er følgende:

- DDE-COMAL-80 arbejder med to typer tal - heltal og reelle tal. En implicit erklæret talvariabel er en reel variabel.

- alle sætningsreferencer til andre sætninger ved deres linienummer er fjernet. I stedet arbejdes der med etiketter, der opbygges på samme måde som identifikatorer.

- EDIT sætningen kan kun benyttes til redigering af en strengværdi

- CHR$ funktionen (i DDE-COMAL kaldet CHR) er udvidet til et tage tre argumenter og kan returnere en talværdi som en streng formatteret afhængig af argumenternes værdier.

- ASC funktionen er en standardfunktion , der returner talværdien af en streng, der indeholder et legalt tal

- specifikationen af formelle parametre i procedurer definerer hvorledes den aktuelle værdi skal overføres - enten som referenceparametre eller som værdiparametre. Samtidig er kontrollen af parametre ved kald af procedurer udvidet.

- sætningsfunktioner (DEF) er udgået og erstattet af en mulighed for at kalde en procedure som funktion.

- FOR og WHILE sætningerne findes i DDE-COMAL-80 som simple sætninger.

Herudover er der en række syntaktiske forskelle mellem DDE-COMAL og DDE-COMAL-80. Disse forskelle fremgår af beskrivelserne af de enkelte sætninger og kommandoer.

## 3. Indtastning

Når DDE-COMAL-80 systemet startes, slettes skærmbillederne på terminalen, og der udskrives en meddelelse med programidentifikation og programversion. Herefter kan terminalen anvendes til indtastning af programsætninger eller kommandoer.

Når en kommando er udført, og DDE-COMAL-80 systemet er klar til at behandle den næste kommando, udskrives en '*' i nederste, venstre hjørne af skærmbilledet. Markøren placeres umiddelbart til højre for '*', hvorefter næste kommando eller programsætning kan indtastes.

### 3.1 Programsætninger

Et DDE-COMAL-80 program består af et antal sætninger. En sætning er skrevet på en eller flere programlinier afhængig af sætningstypen.

En programlinie skal begynde med et linienummer og må afsluttes med en kommentar. Et linienummer skal være et heltal mellem 0 og 9999. Linienummerene bestemmer rækkefølgen for sætningerne i programmet. Et linienummer skal efterfølges af mindst eet blanktegn. En linie må bestå af linienummer uden nogen DDE-COMAL-80 sætning. Eksekvering af en sådan linie har ingen effekt.

En program linie må afsluttes af en kommentar. En kommentar indledes af to på hinanden følgende skråstreger (//), der kan følges af hvilken som helst sekvens af skrivbare ASCII tegn. En kommentar lagres i programmet men har ingen effekt på eksekveringen af programmet.

Når et program eksekveres sker dette normalt i linienummerorden. Denne rækkefølge kan dog brydes af visse sætninger.

Programsætninger må højst fylde 77 tegn, heri medregnet 4 tegn til et linienummer samt 1 tegn til mellemrum efter linienummeret, dvs højst 72 tegn i DDE-COMAL-80 sætningen.

Reglerne for opbygning af DDE-COMAL-80 programmer og sætninger er beskrevet i kapitel 4 - 7.

Når en programlinie er indtastet, foretages en syntaksanalyse. Er linien syntaktisk korrekt, lagres den i overensstemmelse med sit linienummer. Derefter flyttes alle linier på skærmbilledet en linie op, hvorved der opstår en blank linie nederst på skærmbilledet. Her kan brugeren indtaste

næste programlinie eller en kommando. Er linien ikke syntaktisk korrekt, udskriver DDE-COMAL-80 en fejlmelding og linien bliver stående i indtastningsfeltet med markøren placeret på det sted, hvor fejlen blev opdaget. Brugeren kan så rette linien og sende den til syntaksanalyse igen.

## 3.2 Kommandoer

En kommando er en instruktion til DDE-COMAL-80 systemet, som udføres øjeblikkeligt. En kommando består af et nøgleord, der evt. efterfølges af en eller flere parametre. DDE-COMAL-80 kommandoerne er beskrevet i kapitel 8.

En kommando adskiller sig fra en programlinie ved ikke at begynde med et linienummer.

## 3.3 Redigering

Ved indtastning af programsætninger, kommandoer eller inddata til et kørende program (INPUT eller EDIT sætning) har brugeren en række redigeringsfaciliteter til rådighed. Der gælder følgende konventioner:

- indtastning foregår på den nederste linie på skærmen (medmindre der anvendes markørflytning i forbindelse med INPUT eller EDIT, se afsnit 5.6, 5.10 og 5.19). Inddatalinien har en længde på 77 tegn og includerer således efterstillede blanktegn

- -> (cursor right) flytter markøren en plads mod højre uden at ændre noget tegn på linien

- <- (cursor left) flytter markøren en plads mod venstre uden at ændre noget tegn på linien

- ! (cursor up) bevirker, at alle tegn fra og med markørens placering flyttes en plads mod højre. På markørens plads indsættes et blanktegn. Tegnet yderst til højre på linien forsvinder. Operationen anvendes når man ønsker at indsætte et nyt tegn i inddatalinien

- ! (cursor down, line feed) bevirker, at alle tegn fra og med tegnet til højre for markørens placering flyttes en plads mod venstre. Tegnet på markørens plads forsvinder. Yderst til højre på linien indsættes et blanktegn. Operationen anvendes, når man ønsker at slette et tegn i inddatalinien

- TAB flytter markøren mod højre til den nærmeste af positionerne 2, 10, 18, 26, 34, 42, 50, 58 eller 66 uden at ændre noget tegn på linien

- HOME sletter alle tegn fra og med markørens placering og til liniens slutning; på visse dataskærme kan denne funktion være flyttet til en anden tast; dette oplyses ved leveringen

- RUBOUT sletter alle tegn på linien og flytter markøren til den første position på linien

- RETURN betyder afslutning på indtastning af en linie; hele den indtastede linie transmitteres til datamaten uafhængig af markørens placering på det tidspunkt hvor der trykkes på RETURN

## 3.4 Specialtaster

Foruden de i afsnit 3.3 omtalte taster anvender DDE-COMAL-80 følgende specialtaster på tastaturet:

- ESC anvendes til at afbryde et kørende program (se afsnit 8.19) eller en programudskrift (se afsnit 8.10). Hvis ESC nedtrykkes i stedet for RETURN efter indtastningen af en programlinie eller kommando, ignoreres den indtastede linie. Hvis ESC nedtrykkes efter indtastningen af en INPUT eller EDIT linie, afbrydes programeksekveringen. Se også afsnit 5.21.

Det er muligt at omstille de fleste dataskærme på en sådan måde, at ESC tasten kun virker hvis SHIFT tasten samtidig holdes nedtrykket. Dette vil normalt være tilfældet hvis systemet er leveret til administrative anvendelser.

I visse SPC/1 systemer kan ESC tastens funktion være flyttet til en anden tast. I visse af disse systemer ligger de to ovennævnte funktioner (afbrydelse af kørende program eller programudskrift, hhv. unormal afslutning af indtastet linie) på separate taster. Funktionsplaceringerne på tastaturet oplyses i disse tilfælde ved leveringen af systemet.

## 3.5 Øvrige taster

Ud over de i de foregående afsnit nævnte specialtaster, består dataskærmens tastatur af et almindeligt skrivemaskinetastatur, en talblok (højre del af tastaturet) og nogle få yderligere taster:

- talblokken indeholder en række taster, som er en gentagelse af de tilsvarende taster på skrivemaskinedelen af tastaturet

- CAPS LOCK kan benyttes til at sætte skærmen i en af to modi. Ved aktivering af tasten skifter skærmen til en modus, hvor der bogstavdelen af tastaturet kun genererer store bogstaver (CAPS LOCK indikatoren lyser). Når tasten aktiveres pånyskiftes til en modus, hvor der genereres store og små bogstaver afhængigt af om SHIFT tasten holdes nedtrykket eller ej (CAPS LOCK indikatoren lyser ikke)

- CONTROL, BREAK og CLEAR ALL tasterne anvendes ikke i forbindelse med DDE-COMAL-80

## 4. Beskrivelse af COMAL sprogets grundelementer

Dette og det følgende afsnit beskriver de syntaktiske regler for DDE-COMAL-80s grundelementer, suppleret med eksempler.

Medmindre andet udtrykkelig er anført i det følgende, kan blanktegn forekomme overalt i en sætning uden at ændre dennes betydning.

Den anvendte syntaksbeskrivelse er rimelig let at forstå, men ikke fuldstændig præcis. Nøgleord skrevet med store bogstaver og specialtegn skal skrives nøjagtig som vist, men gerne med små bogstaver. Ord skrevet med små bogstaver skal erstattes med en aktuel betegnelse, som forklaret i teksten. Notationen

abc , ... , abc

betyder een eller flere forkomster af abc. Hvis abc forekommer mere end een gang skal de enkelte forekomster være adskilte af det viste skilletegn, for eksempel

| | |
|---|---|
| | abc |
| eller | abc,abc |
| eller | abc,abc,abc |
| osv | |

### 4.1 Tal

Tal benyttes som konstanter i udtryk og i inddata.

Et tal skrives som et heltal eller som et decimalt tal (med decimalpunktum). Umiddelbart efter det sidste ciffer er det tilladt at angive en eksponent, som består af bogstavet E efterfulgt af et fortegn (kan udelades) og den aktuelle tierpotens (heltal), som decimaltallet skal ganges med.

Et tal må ikke indeholde blanktegn.

Husk at bogstavet O ikke kan anvendes i stedet for cifferet 0. Husk også, at bogstavet l (lille L) ikke kan anvendes i stedet for cifferet 1.

Værdiområdet for tal er anført i afsnit 4.5.

| Eksempler: Lovlige tal | Ulovlige tal | |
|---|---|---|
| 55 | 5o | bogstav o ulovlig i tal |
| 11231.0 | 11,231.0 | Positionskomma ulovlig |
| 50. | 50, | Komma ulovlig |
| 47E+06 | 47E+06. | Punktum ulovlig i eksponent |
| 2.4E0018 | 2.4E25 | Uden for værdiområde |
| .7E-10 | | |

Et tal kan forsynes med et fortegn, enten + eller -, der skal placeres til venstre for det første ciffer.

Eksempler: + 3.6
-477E-06

## 4.2 Identifikatorer

Identifikatorer benyttes i et program som navne på variable, procedurer mv.

En identifikator består af et eller flere symboler, hvoraf det første skal være et bogstav, og de efterfølgende skal være enten bogstaver, cifre eller symbolet _ .

En identifikator må højst bestå af 16 tegn, og må ikke indeholde blanktegn. En identifikator skal være forskellig fra de reserverede nøgleord (se afsnit 4.3), og umiddelbart før og umiddelbart efter en identifikator skal der stå et tegn, som ikke er et bogstav, et ciffer eller symbolet _.

Det er tilladt at benytte både store og små bogstaver i identifikatorer. DDE-COMAL-80 omsætter dog automatisk alle bogstaver i identifikatorer til store bogstaver, dvs. at ´Variabel1´ og ´VARIABEL1´ er den samme identifikator.

Identifikatorer kan være navne for følgende typer:

- simple talvariable
- indicerede talvariable (talsæt og matricer)
- simple strengvariable
- indicerede strengvariable
- standardfunktioner
- etiketter
- formelle parametre
- procedurer

En identifikator kan kun betegne een størrelse i et program. Dette betyder fx, at A og A$ ikke kan benyttes i det samme program, idet identifikatoren A da ville være navn på både en talvariabel og en strengvariabel.

| Eksempler: Lovlige identifikatorer | Ulovlige identifikatorer | |
|---|---|---|
| A123 | A1234$56789 | Ulovligt tegn ($) |
| abcdefGhi | ENDWHILE | Nøgleord |
| EnLangVariabel | 1VARIABEL | 1. tegn skal være et bogstav |
| COMAL | ENFORLANGVARIABEL | over 16 tegn |
| SIN | | |
| ANTAL_TIMER | | |
| LØN_ÅR_TIL_DATO | | |

## 4.3 Nøgleord

I DDE-COMAL-80 findes en række nøgleord, der har en fast betydning. Disse nøgleord må ikke benyttes som identifikatorer. Derimod må de gerne indgå som en del af en identifikator, fx er IF1 en tilladt identifikator.

De reserverede nøgleord er:

CALL, CASE, CHAIN, CLEAR, CLOSE, CREATE, CURSOR, DATA, DIM, EDIT, ELSE, END, ENDCASE, ENDFILE, ENDIF, ENDPROC, ENDWHILE, EXEC, EXIT, IF, IN, INPUT, INTEGER, FOR, GET, GO, GOTO, NEXT, ON, OPEN, OTHERWISE, OUT, OUTPUT, PRINT, PROC, PUT, READ, REAL, REPEAT, RESTORE, SELECT, STOP, WHEN, WHILE, UNTIL

Nøgleord kan indtastes med store eller små bogstaver. Ved udskrift vil de blive udskrevet med store bogstaver.

Umiddelbart før og umiddelbart efter et nøgleord skal stå et tegn, som hverken er et bogstav, et ciffer eller symbolet _. Nøgleord må ikke indeholde blanktegn.

## 4.4 Talvariable

I DDE-COMAL-80 findes to typer talvariable, nemlig heltalsvariable og reelle variable. Hver af disse to typer kan så igen være simple variable eller talsæt (indicerede variable).

En simpel variabel repræsenteres ved en identifikator. En simpel reel variabel skal ikke erklæres eksplicit, idet erklæringen automatisk sker første gang, DDE-COMAL-80 udfører en sætning, der tildeler en værdi til variabelen. Når en variabel oprettes, er dens værdi udefineret. Derimod skal en heltalsvariabel altid erklæres før den refereres - erklæringen sker i en INTEGER sætning.

En indiceret variabel består af en række elementer organiseret som en vektor (enkeltindiceret) eller en matrix (dobbeltindiceret) En reference til en indiceret variabel (et variabelelement) har følgende opbygning:

```
                identifikator ( indeks )
eller           identifikator ( indeks , indeks )
```

hvor identifikatoren betegner navnet på den indicerede variabel, og hvor positionen af det aktuelle element bestemmes af de anførte indices. Antallet af indices i referencen skal stemme overens med antallet af dimensioner i erklæringen af den indicerede variable.

En indiceret variabel skal erklæres før den anvendes. Ved en sådan erklæring skal dimensionen samt øvre grænse for hver enkelt dimension angives. Nedre grænse for indeks er altid 1. Erklæringen sker i en DIM, INTEGER eller REAL sætning.

Ved alle referencer til indicerede variable foretages en kontrol af indices, dvs. det undersøges om antallet af indices og deres værdier er tilladelige.

```
Eksempler på talvariable:          ANTAL
                                   Vektor(12)
og reference til indiceret variabel: MATRIX( 4+ANTAL, 5-VEKTOR(2) )
```

## 4.5 Aritmetiske udtryk

Aritmetiske udtryk indgår i en række forskellige sætninger.

Aritmetiske udtryk har følgende opbygning:

```
            operand
eller       + operand
eller       - operand
```

En operand i et aritmetisk udtryk har følgende opbygning:

```
            operand + operand
eller       operand - operand
eller       operand * operand
eller       operand / operand
eller       operand MOD operand
eller       operand DIV operand
eller       operand ^ operand
eller       operand ** operand
eller       ( aritmetisk-udtryk )
eller       ( logisk-udtryk )
eller       tal
eller       simpel talvariabel
eller       talvariabelelement
eller       standardfunktion
```

Standardfunktionerne er omtalt i kapitel 7.

De aritmetiske operatorer er

| | |
|---|---|
| + | monadisk plus |
| - | monadisk minus |
| ^ | potensopløftning |
| ** | potensopløftning |
| * | multiplikation |
| / | division |
| MOD | beregning af modulus |
| DIV | heltalsdivision |
| + | addition |
| - | subtraktion |

Værdien af X DIV Y (X heltalsdivideret med Y) er defineret som INT(X / Y) (se om INT i kapitel 7).

Værdien af X MOD Y (X modulo Y) er defineret som X-((X DIV ABS(Y))*ABS)Y) (se om ABS i kapitel 7).

Eksempler:

| | |
|---|---|
| 11 DIV 7 = 1 | 11 MOD 7 = 4 |
| -11 DIV 7 = -2 | -11 MOD 7 = 3 |
| -11 DIV (-7) = 1 | -11 MOD (-7) = 3 |
| 11 DIV (-7) = -2 | 11 MOD (-7) = 4 |

Beregningen af aritmetiske udtryk, herunder operatorprioriteterne, er omtalt i afsnit 4.10.

Aritmetiske udtryk kan antage værdien 0 og alle værdier i intervallerne +/-(1E-127 til 1E127). Beregningerne sker med 13 cifres nøjagtighed. Heltalsvariable kan dog kun repræsentere heltallige værdier i intervallet -32768 til 32767.

Typen af et aritmetisk udtryk bestemmes af de indgående størrelser, således at hvis der optræder blot een størrelse af typen reel (enten ved reference til en reel talvariabel, en reel konstant eller en heltalskonstant, der falder uden for det talområde, der kan repræsenteres af en heltalsvariabel) bliver værdien af udtrykket reelt, i alle andre tilfælde bliver udtrykket af typen heltal.

Ved tildeling af talværdier til en talvariabel konverteres værdien automatisk til variablens type. Ved tildeling af en værdi til en heltalsvariabel vil der blive givet en fejlmeddelelse, hvis talværdien falder uden for intervallet, der kan repræsenteres af en heltalsvariabel.

Det er vigtigt at bemærke, at denne implicitte konvertering belaster processen tidsmæssigt, og derfor bør søges undgået. Ligeledes skal det bemærkes at regning på heltal er væsentlig hurtigere på heltal end på reelle tal. Fx. bør den styrende variabel i en FOR sætning altid være af typen heltal. Generelt kan det siges, at heltal er at foretrække for reelle tal, hvor det heltallige talområde opfylder betingelserne.

Hvis den absolutte værdi af et aritmetisk udtryk beregnes til en værdi, der er mindre end 1E-127 uden at være 0, forekommer der et underløb (underflow). I tilfælde af underløb sættes værdien af det aritmetiske udtryk til 0, hvorefter beregningerne fortsættes.

Hvis den absolutte værdi af et aritmetisk udtryk beregnes til en værdi, der er større end 1E127, forekommer et overløb (overflow). I dette tilfælde afbrydes beregningerne med en fejlmeddelelse.

Den aritmetiske værdi af et sandt logisk udtryk er 1. Den aritmetiske værdi af et falsk logisk udtryk er 0. Logiske udtryk er omtalt i afsnit 4.9.

Eksempel (se også afsnit 4.10):

```
( A = B ) * 20 + 5  =  25    hvis A = B
                    =   5    hvis A <> B
```

## 4.6 Heltalsudtryk

Heltalsudtryk anvendes som indices, og anvendes desuden i en række andre sætningstyper (CURSOR, CREATE, GET, PUT).

Et aritmetisk udtryk accepteres som heltalsudtryk hvis dets værdi er større end eller lig 0.5 og mindre end 32767.5. Det aritmetiske udtryk konverteres til en heltalsværdi ved afrunding, dvs. ved addition af 0.5 med efterfølgende bortkastning af decimaler.

## 4.7 Strengvariable

DDE-COMAL-80 er i stand til at arbejde med tegnstrenge i form af strengvariable og strengkonstanter.

Indholdet af en strengvariabel er en sekvens af ASCII tegn.

DDE-COMAL-80 har to typer af strengvariable, nemlig simple strengvariable og vektorer af strengvariable (indicerede strengvariable).

Strengvariable adskiller sig fra talvariable ved, at der efter identifikatoren skal følge et $. Strengvariable skal erklæres i en DIM sætning og har følgende opbygning:

```
        identifikator$ OF max længde
eller   identifikator$ ( antal elementer ) OF max længde
```

hvor max længde og antal elementer er heltalsudtryk. Ved erklæringen afsættes der plads i lageret til strengvariablen, og alle tegn tildeles værdien 'slutmærke'.

En simpel strengvariabel kan refereres på følgende måder:

```
        identifikator $
eller   identifikator $ ( startposition )
eller   identifikator $ ( startposition : længde )
```

En reference til en indiceret strengvariabel - et variabelelement - skrives på følgende måde:

```
          identifikator $ ( indeks )
eller     identifikator $ ( indeks, startposition )
eller     identifikator $ ( indeks, startposition : længde )
```

Den evt. opgivne startposition, slutposition, længde og indeks skal være heltalsudtryk.

Når en simpel strengvariabel kun angives ved den tilhørende identifikator menes der hele strengen af ASCII tegn.

Er startposition og længde opgivet, menes der en delstreng med den opgivne længde fra og med startpositionen. Det første tegn i en streng er i position 1. Hvis kun startpositionen er opgivet, bliver resultatet en delstreng med eet tegn.

For en indiceret strengvariabel skal der angives et indeks, der udpeger det ønskede element i vektoren. Herudover kan der eventuelt være en delstrengsangivelse. Betydningen af denne er som for simple strengvariable.

Strenge og vektorer af strenge skal erklæres i en DIM sætning. Når en tegnstreng erklæres, afsættes lagerplads til strengvariablen og alle tegnene i strengvariablen tildeles værdien ´slutmærke´.

Ved den aktuelle længde af en tegnstreng forstås antallet af sammenhængende tegn regnet fra tegnstrengens start til det første slutmærke. Umiddelbart efter oprettelsen er den aktuelle længde af en tegnstreng derfor nul.

Ved programudførelsen foretages kontrol af indeks. I forbindelse med denne kontrol undersøges om indeks til en strengvektor, startposition samt evt. længde er tilladelige.

Eksempel: STRENGVAR$ er en simpel strengvariabel.

```
0010 DIM STRENGVAR$ OF 11
0020 STRENGVAR$ := "TEKSTSTRENG"
0030 PRINT STRENGVAR$
0040 PRINT STRENGVAR$(6)
0050 PRINT STRENGVAR$(6:6)
0060 END
0070
*RUN
TEKSTSTRENG
S
STRENG
```

Eksempel: STRENGINDVAR$ er en indiceret strengvariabel

```
0010 DIM STRENGINDVAR$(8) OF 11
0020 STRENGINDVAR$(4) := "TEKSTSTRENG"
0030 PRINT STRENGINDVAR$(4)
0040 PRINT STRENGINDVAR$(4,6)
0050 PRINT STRENGINDVAR$(4,6:6)
0060 END
0070
*RUN
TEKSTSTRENG
S
STRENG
```

## 4.8 Strengudtryk

Strengudtryk benyttes ved tildeling af værdier til strengvariable (i tildelingssætninger), ved udskrivning (i PRINT sætning) og i relationer.

Strengudtryk har følgende opbygning:

```
                simpel strengvariabel
eller           strengvariabelelement
eller           delstreng
eller           ""
eller           " et eller flere ASCII tegn "
eller           strengudtryk + strengudtryk
```

En strengkonstant består af et eller flere vilkårlige ASCII tegn omgivet af ". Tegnet " må dog ikke forekomme i sekvensen, da det markerer afslutningen på strengkonstanten. Hvis tegnet " skal forekomme i en tegnstreng, kan det indsættes ved hjælp af funktionen CHR$ (se afsnit 7.2).

Indgår en strengvariabel i et strengudtryk, vil længden af strengen være bestemt af af strengvariablens aktuelle længde, evt af længden i en delstrengsangivelse.

+ er en strengoperator, der angiver sammenkædning (konkatenering) af tegnstrenge. I udtrykket C$ := A$ + B$ fremkommer C$ ved at indholdet af B$ sættes umiddelbart efter indholdet af A$, og længden af C$ bliver summen af længderne af A$ og B$. Tildelinger af formen A$ := A$ + B$ er tilladte og ofte tilrådelige.

Eksempel:

```
0010 DIM NAVN$ OF 30
0020 NAVN$ := "SAM "
0030 PRINT NAVN$
0040 NAVN$ := NAVN$ + "FINKELSTEIN"
0050 PRINT NAVN$
0060 END
0070
*RUN
SAM
SAM FINKELSTEIN
```

## 4.9 Logiske udtryk

Logiske udtryk benyttes til at gøre sætningsudførelsen betinget. Logiske udtryk indgår i IF - THEN sætninger, WHILE sætninger og UNTIL sætninger. Logiske udtryk omgivet af parenteser kan desuden indgå i aritmetiske udtryk.

Logiske udtryk har følgende opbygning:

```
              logisk-operand
eller         NOT logisk-operand
eller         aritmetisk udtryk
```

En operand i et logisk udtryk har følgende opbygning:

```
              logisk-operand  OR  logisk-operand
eller         logisk-operand  AND  logisk-operand
```

eller aritmetisk udtryk relationsoperator aritmetisk udtryk
eller strengudtryk relationsoperator strengudtryk
eller ( logisk-udtryk )
eller strengudtryk IN strengudtryk

En relationsoperator kan være:

| | |
|---|---|
| < | mindre end |
| <= og =< | mindre end eller lig med |
| > | større end |
| => og >= | større end eller lig med |
| = | lig med |
| <> og >< | forskellig fra |

Desuden betyder de logiske operatorer

| | |
|---|---|
| NOT | negation |
| AND | logisk og |
| OR | logisk eller |

I de tilfælde, hvor der i ovenstående liste optræder to relationsoperatorer med samme betydning, kan begge anvendes ved indtastning, men systemet vil ved udskrivning altid vise den første.

Beregningen af logiske udtryk, herunder operatorprioriteterne, er omtalt i afsnit 4.10.

Den logiske værdi af et aritmetisk udtryk, med værdien nul, er ´falsk´. Den logiske værdi af et aritmetisk udtryk, med en værdi forskellig fra nul, er ´sand´.

Ved sammenligning af to forskellige taltyper foretages en implicit konvertering inden sammenligningen.

I en strengrelation sammenlignes strengudtrykkene tegn for tegn fra venstre mod højre (laveste indeks først), indtil der mødes en uoverensstemmelse (tegnstrengene er forskellige) eller begge tegnstrenge er slut (tegnstrengene er ens). Sammenligningerne er aritmetiske sammenligninger af ASCII værdierne for de sammenlignede tegn (se appendix A). Den aritmetiske værdi af det første tegn efter en tegnstreng er 0, dvs. at hvis to tegnstrenge er ens ind til det sted, hvor den ene tegnstreng slutter, vil den korteste tegnstreng være mindst.

IN operatoren tager som argument to strengudtryk og undersøger om det første strengudtryk er indeholdt i det andet. Værdien af IN operationen

er en heltalsværdi, der er index på den første forekomst af det første strengudtryk i det andet strengudtryk. Hvis det første strengudtryk ikke er indeholdt i det andet strengudtryk er værdien af IN 0. Hvis det første strengudtryk evalueres til den tomme streng er værdien af IN længden af det andet strengudtryk + 1.

Eksempler på sande strengrelationer:

```
"ABC"  = "ABC"
"ABCD" = "ABC" + "D"
"ABCD" < "Abcd"
"ABC"  < "ABCD"
"ABC"  < "ABC "
"ABD"  > "ABC"
```

## 4.10 Beregning af udtryk

Beregning af aritmetiske og logiske udtryk sker efter følgende regler:

1. når man møder en venstreparentes, beregnes udtrykket i parentesen altid før resten af det igangværende udtryk beregnes

2. når man møder et kald af en procedure som funktion beregnes denne før resten af udtrykket. Bemærk at en procedure kaldt på denne måde kan have utilsigtede sideeffekter

3. beregning af deludtryk sker i en rækkefølge, der afhænger af prioriteten for den anvendte operator. Operatorernes prioriteter er

| | |
|---|---|
| første (højst) | monadisk plus eller minus |
| anden | potensopløftning (^ eller **) |
| tredie | * / DIV MOD |
| fjerde | + - |
| femte | sammenkædning af tegnstrenge (+) |
| sjette | alle relationsoperatorer samt IN operatoren |
| syvende | logisk negation (NOT) |
| ottende | logisk og (AND) |
| niende (lavest) | logisk eller (OR) |

4. hvis to operatorer har samme prioritet sker beregningerne fra venstre mod højre

Eksempel: Understregningen viser i hvilken rækkefølge beregningerne udføres.

```
30 DIV 3 * (8 - 2 * 3) + 8 - 3 ** 2
   10    * (8 - 2 * 3) + 8 - 3 ** 2
   10    * (8 -   6  ) + 8 - 3 ** 2
   10    *       2     + 8 - 3 ** 2
         20              + 8 - 3 ** 2
               28          - 3 ** 2
               28          -   9
                     19
```

## 5. COMAL sætninger

Dette kapitel beskriver syntaksen for COMAL sætninger i DDE-COMAL-80 systemet. Sætninger, der bruges i forbindelse med filsystemet, er beskrevet i kapitel 6.

### 5.1 Programstruktur

Løsning af en opgave ved hjælp af DDECOMAL80 sker ved at skrive et program og udføre dette.

Et DDE-COMAL-80 program består af et antal programsætninger (se afsnit 3.1). De lovlige sætningstyper er beskrevet i afsnit 5.2 5.32 og i kapitel 6.

Det er vigtigt at bemærke, at COMAL i modsætning til de fleste andre programmeringssprog eksekverer erklæringer. Dette betyder, at erklæringer kan anbringes hvorsomhelst i et program, dog med den begrænsning, at en erklæring af en variabel skal eksekveres, før der første gang refereres til variablen.

Procedureerklæringer og DATA-sætninger eksekveres dog ikke. Disse sætninger behandles særskilt af systemet umiddelbart efter RUN kommandoen, og inden den egentlige programudførelse påbegyndes.

Visse DDE-COMAL-80 sætninger omfatter flere programsætninger. Disse sammensatte sætninger markeres ved en indledende og en afsluttende programsætning. Sætningslisten mellem den indledende og den aflsuttende programsætning kan indeholde en eller flere eller slet ingen DDE-COMAL-80 sætninger incl sammensatte sætninger, men det er vigtigt at bemærke, at hvis en sætningsliste indeholder en eller flere sammensatte sætninger, skal disse sætninger afsluttes inden for sætningslisten.

Det er ikke tilladt at benytte hopsætninger (GOTO) til indhop i en sætningsliste i en sammensat sætning. Ligeledes er det ikke tilladt at hoppe ud af en procedure.

Eksempler på DDE-COMAL-80 programmer findes i appendix C.

DDE-COMAL-80 udfører sætningerne i den rækkefølge, der bestemmes af linienumrene. Rækkefølgen kan dog ændres af bla EXEC, ENDPROC, EXIT og GOTO.

## 5.2 CALL - udfør assembler underprogram

CALL sætningen bruges til at aktivere et program eller underprogram, som er skrevet i SPC/1 assembler (maskinsprog).

CALL sætningen har følgende opbygning:

```
CALL " etikette : filnavn "
```

hvor etikette:filnavn er navnet på den fil, hvor det ønskede program er lagret. Se kapitel 6 om navngivning af filer.

Umiddelbart efter at RUN kommandoen er indtastet, gennemløber DDE-COMAL-80 hele det aktuelle program. Hver gang en CALL-sætning mødes, indlæses det kaldte program til dataområdet, og det disponible areal reduceres med den plads, som de indlæste programmer optager.

Når en CALL sætning udføres, aktiverer DDE-COMAL-80 det angivne program. Når programmet er afsluttet, returneres kontrollen til DDE-COMAL-80, som herefter genoptager eksekveringen af COMAL programmet med sætningen umiddelbart efter CALL-sætningen.

Assembler programmer kan læse og ændre indholdet af eksisterende DDE-COMAL-80 variable. Assembler programmer kan derimod ikke oprette nye DDE-COMAL-80 variable.

Fordelen ved at benytte programmer skrevet i SPC/1 assembler ligger i en væsentlig hurtigere eksekvering, og i at assembler programmer har adgang til en lang række faciliteter i datamaten, herunder specielt i/u systemet, som ikke er tilgængelige fra DDE-COMAL-80. Ulempen ved assemblerprogrammer er først og fremmest den, at de er langt mere besværlige at skrive og afprøve end DDE-COMAL-80 programmer. Desuden kan et fejlbehæftet assemblerprogram standse hele systemet, hvad et fejlbehæftet DDE-COMAL-80 program ikke kan.

Nærmere oplysninger om skrivning af assemblerprogrammer til DDE-COMAL-80 findes i manualen "DDE-COMAL ASSEMBLER INTERFACE", som kan købes hos Dansk Data Elektronik.

Eksempel: Programmet "DATO" fra pladen med etiketten "DDE" ønskes udført.

```
0010 CALL "DDE:DATO"
```

## 5.3 CASE - forgrening

CASE sætningen benyttes når een af flere sætningslister skal udføres afhængig af værdien af et udtryk.

CASE sætningen har følgende opbygning:

```
CASE  caseudtryk  OF
WHEN  udtryksliste
  sætningsliste

...

OTHERWISE
  alternativ sætningsliste
ENDCASE
```

Nøgleordet OTHERWISE og den efterfølgende alternative sætningsliste kan udelades.

Caseudtrykket og de enkelte udtryk i udtrykslisterne skal være enten strengudtryk eller aritmetiske udtryk af typen heltal, og caseudtrykket og alle whenudtryk skal være af samme type.

Sætningslisterne må indeholde vilkårlige sætninger. Hvis en sætningsliste indeholder en eller flere sammensatte sætninger, skal disse sætninger afsluttes inden for sætningslisten.

Udførelse af en CASE sætning starter med at caseudtrykket beregnes. Herefter beregnes udtrykkene i WHEN sætningerne en for en, indtil der findes en værdi, som er identisk med værdien af caseudtrykket. Hvis der opdages en identitet, udføres sætningslisten svarende til den pågældende WHEN sætning, og herefter fortsættes udførelsen efter ENDCASE. Selv om der er mulighed for flere identiteter, vil DDE-COMAL-80 kun tage den første i betragtning.

Hvis der ikke opdages nogen identitet, udføres den alternative sætningsliste, hvorefter udførelsen fortsættes efter ENDCASE. Sætningslister, der svarer til WHEN sætninger, hvor ingen af udtryksværdierne svarer til den værdi, der blev beregnet i CASE sætningen, udføres ikke.

Eksempel:

```
0010 DIM CIFFER$ OF 1
0020 INPUT "INDTAST ET CIFFER ":CIFFER$
0030 CASE CIFFER$ OF
0060 WHEN "0"
0070   PRINT "DU HAR INDTASTET ET NUL"
0080 WHEN "1", "2", "3", "5", "7"
0090   PRINT "DU HAR INDTASTET ET PRIMTAL"
0100 WHEN "4", "8"
0110   PRINT "DU HAR INDTASTET EN POTENS AF TO"
0120 WHEN "6"
0130   PRINT "DET VAR ET SEKSTAL"
0140 WHEN "9"
0150   PRINT "DET VAR ET NITAL"
0155   OTHERWISE // ALTERNATIV, DVS. IKKE NOGET TAL
0157   PRINT "JEG BAD DIG OM AT INDTASTE ET TAL!"
0160 ENDCASE
0170 END
```

## 5.4 CHAIN - udfør nyt program

CHAIN sætningen medfører, at udførelsen af det igangværende program standses, og et binært program indlæses fra en pladelagerfil. Herefter startes udførelsen af det nyindlæste program i det laveste linienummer.

CHAIN sætningen har følgende opbygning:

CHAIN filidentifikator

Filidentifikator er beskrevet i afsnit 6.1.2.

CHAIN sætningen anvendes typisk til at opdele et stort program i et antal mindre programmer, eller til at eksekvere uafhængige programmer fra et styreprogram på basis af brugerkommandoer.

Parameteroverførsel til det nyindlæste program er ikke direkte muligt, men kan fx ske ved hjælp af en fil, hvor det kaldende program gemmer de værdier, som ønskes overført til det nye program.

Det binære program, der skal udføres, skal tidligere være gemt i pladelagerfilen ved hjælp af en SAVE, SAVEOLD eller SAVEPROTECT kommando.

Eksempel: Programmet "PROGRAM2" hentes fra disken med etiketten "JENS" og opstartes.

```
0560 CHAIN "JENS:PROGRAM2"
```

CHAIN sætningen har samme funktion som en STOP eller END sætning, der efterfølges af kommandoerne NEW, LOAD etikette:filnavn, RUN. Det betyder, at alle datafiler i programmet med CHAIN sætningen lukkes og skal åbnes pånyi det indlæste program, og at fejl i referencen til det nye program resulterer i en fejlmeddelelse og et tomt arbejdslager.

## 5.5 CLEAR - slet dataskærm

En CLEAR sætning bevirker at dataskærmens indhold slettes, hvorefter markøren flyttes til øverste venstre hjørne af dataskærmen. Hvis brugeren ved hjælp af OUTPUT kommandoen/sætningen har dirigeret uddata til linieskriveren, tømmes uddatabufferen, og der skiftes side på linieskriveren. I dette tilfælde ændres skærmbilledet ikke.

CLEAR sætningen har følgende opbygning:

```
CLEAR
```

Eksempel: Se programmet i appendix C.1

## 5.6 CURSOR - flyt markør

En CURSOR sætning bevirker, at uddatabufferen tømmes, hvorefter markøren (cursor) flyttes til en bestemt position på dataskærmen.

CURSOR sætningen har følgende opbygning:

```
CURSOR  x position , y position
```

X position og y position skal begge være aritmetiske heltalsudtryk. X position angiver markørens nye x-koordinat ( 1 <= x <= 80 ). Y position angiver markørens nye y-koordinat ( 1 <= y <= 24 ). X-aksen regnes positiv fra venstre mod højre. Y-aksen regnes positiv fra oven og nedefter, dvs. at det øverste venstre hjørne har koordinaterne (x,y) = (1,1), mens det nederste højre hjørne har koordinaterne (x,y) = (80,24).

Hvis et af koordinatudtrykkene har en værdi, som falder uden for de angivne intervaller, standses udførelsen med en fejludskrift.

Hvis brugeren gennem en OUTPUT kommando/sætning har angivet, at uddata ønskes udskrevet på linieskriveren, ignoreres CURSOR sætningen.

Eksempel: Se programmet i appendix C.1

## 5.7 DATA - definer faste data

DATA sætningen benyttes til at definere værdier i datalisten. Disse værdier kan indlæses ved hjælp af READ sætninger.

DATA sætningen har følgende opbygning:

DATA dataelement , ... , dataelement

Et dataelement kan være

| | | |
|---|---|---|
| | tal | (se afsnit 4.1) |
| eller | + tal | |
| eller | - tal | |
| eller | strengkonstant | (se afsnit 4.8) |

Ved starten af programmets udførelse, dvs. umiddelbart efter indtastningen af RUN kommandoen, gennemgås programmet linie for linie, og alle DATA sætningerne kædes sammen i en dataliste ordnet efter linienummer. Den første variabel i den først udførte READ sætning vil få tildelt den første værdi fra den første DATA sætning i programmet osv.

Under selve programudførelsen ignoreres DATA sætningerne.

Eksempel:

```
0010 DATA 1, 2, 3, 5, 7, 11, 3.141592
0020 DATA "JENS ANDERSEN"
0030 DATA "OLE NIELSEN", -1675.37, "DEBET"
0040 DATA "KALLE JENSEN", "A"
0050 DATA +8.543E-12
```

Se i øvrigt afsnit 5.26 og 5.28

## 5.8 DIM - dimensioner variable

DIM sætningen benyttes til at erklære reelle talsæt (vektorer og matricer) samt strenge og vektorer af strenge. Erklæringer skal eksekveres inden de erklærede størrelser første gang benyttes i programmet.

DIM sætningen har følgende opbygning:

```
DIM erklæring , ... , erklæring
```

En erklæring kan være

```
                variabelnavn  ( heltalsudtryk )
eller           variabelnavn  ( heltalsudtryk, heltalsudtryk )
eller           variabelnavn$ OF max længde
eller           variabelnavn$ ( antal strenge ) OF max længde
```

De to øverste former for erklæringer anvendes til erklæring af reelle talsæt (vektorer, matricer), mens de to sidstnævnte former anvendes til erklæring af simple strengvariable og vektorer af strenge (indicerede strengvariable). Det er tilladt at blande de forskellige erklæringstyper i samme sætning.

I erklæringen af en streng angiver max længde det maksimale antal tegn strengen kan indeholde, mens en erklæring af en vektor af strenge angiver antallet af strenge i vektoren samt den maksimale længde for hver af disse strenge. Alle strenge i en vektor af strenge får samme maksimale længde.

Ved udførelse af DIM sætningen afsættes plads i dataområdet til de størrelser, der erklæres. Maksimumsstørrelsen for en indiceret variabel eller en strengvariabel er kun begrænset af arbejdslagerets størrelse.

En variabel må kun erklæres een gang.

Eksempler:

```
DIM ALFABET$ OF 29
DIM VEKTOR(15),MATRIX(3,3),BOGSTAV$ OF 1
DIM ELEVNAVNE$(50) OF 30
```

Den sidste erklæring definerer en strengvektor, som indeholder 50 strenge med hver 30 tegn.

## 5.9 EDIT - rediger tegnstreng eller variabel

EDIT sætningen benyttes til redigering af indholdet af en strengvariabel.

EDIT sætningen har følgende opbygning

```
            EDIT  strengvariabel
eller       EDIT  ledetekst : strengvariabel
```

Ledeteksten er en strengkonstant (se afsnit 4.8), og strengvariabel står for en reference til en simpel, et strengvariabelelement eller en delstreng.

Ved udførelsen af EDIT sætningen skrives den eventuelle ledetekst samt indholdet af den refererede variabel. Skærmoperatøren kan nu rette i det udskrevne ved hjælp af de sædvanlige redigeringsfunktioner (se afsnit 3.2). Endelig læses den redigerede værdi til den refererede variabel.

Hvis uddatabufferen er tom, og der ikke er angivet en ledetekst, udskriver DDE-COMAL-80 et ':' inden udskrivningen af det aktuelle variabelindhold.

Udskrivningen af det aktuelle variabelindhold foretages præcis som ved PRINT sætningen (se afsnit 5.23).

Der kan højst redigeres et antal tegn, der er bestemt af den maksimale længde af den referede streng eller længden af delstrengen.

Ved redigering af en strengvariabel anvendes " ikke til afgrænsning. Når den redigerede værdi skal tildeles strengvariablen efter redigeringen er afsluttet, anvendes reglerne for tildeling, som er beskrevet i afsnit 5.31. Det er dog vigtigt at bemærke, at evt efterstillede blanktegn ignoreres ved denne tildeling, dvs at strengen ved tildelingen kun omfatter tegnene fra og med indtastningens start og til og med sidste ikke-blanktegn.

EDIT sætningen opererer altid på dataskærmen, dvs. den er uafhængig af OUTPUT kommandoen/sætningen.

Eksempel:
```
0100 DIM NAVN$ OF 30,SVAR$ OF 1
0110 INPUT "INDTAST NAVN ":NAVN$
0120 REPEAT
0130   PRINT "ER DU TILFREDS MED DET DU NETOP INDTASTEDE (J/N) ? ";
0140   // FORESLÅ SVARET "J"
0150   // GENTAG SPØRGSMÅL INDTIL SVAR ER "J" ELLER "N"
0160   SVAR$:="J"
0170   EDIT "":SVAR$
0180 UNTIL SVAR$="J" OR SVAR$="N"
0190 IF SVAR$="N" THEN
0200   EDIT "RET NAVN ":NAVN$
0210 ENDIF
0220
*RUN
INDTAST NAVN A.OLSEN
ER DU TILFREDS MED DET DU NETOP INDTASTEDE (J/N) ? N
RET NAVN ANDERS OLSEN
```

I næstnederste uddatalinie har brugeren rettet J til N. I nederste linie har brugeren rettet "A.OLSEN" til "ANDERS OLSEN" ved hjælp af redigeringstasterne.

## 5.10 EXEC - udfør procedure

Procedurekald benyttes til at få udført en procedure, som er defineret i en procedureerklæring (se afsnit 5.25).

Procedurekald har følgende opbygning:

```
          EXEC  procedurenavn
eller     EXEC  procedurenavn ( parameter , ... , parameter )
```

En parameter i en EXEC sætning kaldes en aktuel parameter og er enten en simpel talvariabel, en indiceret talvariabel, en simpel strengvariabel, en indiceret strengvariabel, et aritmetisk udtryk eller et strengudtryk.

EXEC sætningen bevirker, at proceduren betegnet ved procedurenavn kaldes. Programudførelsen starter i sætningen umiddelbart efter PROC og fortsætter i proceduren indtil en EXIT eller ENDPROC sætning mødes. Derefter fortsætter programudførelsen med sætningen umiddelbart efter EXEC sætningen.

Hvis EXEC sætningen indeholder en parameterliste, skal antallet af (aktuelle) parametre stemme overens med antallet af (formelle) parametre i

procedureerklæringen, og reglerne for substitution af formelle parametre skal overholdes.

Reglerne for substitution af de formelle parametre med aktuelle parametre er følgende:

| Formel parameter erkl. | Tilladt aktuel param. | Overføres som |
|---|---|---|
| Simpel talvariabel | Aritmetisk udtryk | Værdi |
| REF simpel talvariabel | Simpel talvariabel | Reference |
| REF talvariabel () | Enkelt indiceret talvariabel | Reference |
| REF talvariabel (,) | Dobbelt indiceret talvariabel | Reference |
| Simpel strengvariabel | Strengudtryk | Værdi |
| REF simpel strengvar. | Simpel strengvariabel | Reference |
| REF strengvariabel () | Indiceret strengvar. | Reference |

Hvis en aktuel parameter er en indiceret talvariabel skal den have samme antal indices som den tilsvarende formelle parameter.

Eksempler: Overførsel af reelle parametre

```
0010 PROC REELTEST(REF P1, P2 )
0020   PRINT "PARAMETRE TIL REELTEST ";P1;P2
0030   P1:=P1+5
0040   P2:=P2+5
0050 ENDPROC REELTEST
0060 //*****************************************
0070 DIM C(9)
0080 A:=5;B:=7;C(5):=17
0090 EXEC REELTEST(A,1+B)
0100 PRINT "HILSEN FRA LINIE 100: A,B= ";A;B
0110 EXEC REELTEST(B,C(5))
0120 PRINT "HILSEN FRA LINIE 120: B,C(5)= ";B;C(5)
0130 END
0140
*RUN
PARAMETRE TIL REELTEST 5.000000000000 8.000000000000
HILSEN FRA LINIE 100: A,B= 10.00000000000 7.000000000000
PARAMETRE TIL REELTEST 7.000000000000 17.00000000000
HILSEN FRA LINIE 120: B,C(5)= 12.00000000000 17.00000000000
```

Overførsel af strengvariable:

```
0010 DIM STRENG$ OF 17, BOGSTAV$ OF 1
0020 STRENG$:="abcDEfghijKLm12"
0030 EXEC STORE_BOGSTAVER(STRENG$)
0040 PRINT STRENG$
0050 END
0060 //************************************************
0070 PROC STORE_BOGSTAVER(REF STRENGPARAMETER$)
0080   FOR INDEKS:=1 TO  LEN(STRENGPARAMETER$)
0090     BOGSTAV$:=STRENGPARAMETER$(INDEKS)
0100     IF BOGSTAV$=>"a" AND BOGSTAV$<="å" THEN
0110       BOGSTAV$:=CHR$(ORD(BOGSTAV$)+ORD("A")-ORD("a"))
0120       STRENGPARAMETER$(INDEKS):=BOGSTAV$
0130     ENDIF
0140   NEXT INDEKS
0150 ENDPROC STORE_BOGSTAVER
0160
*RUN
ABCDEFGHIJKLM12
```

## 5.11 Etikette erklæring

Etiketter benyttes til at ændre den sekventielle udførelse af programsætningerne. Etiketterne refereres af GOTO sætninger.

En etikette erklæring har følgende opbygning:

```
etikette:
```

En etikette er en identifikator.

En etikette sætning behøver ikke at være udført før der hoppes til etiketten. I den sekventielle udførelse af programsætningerne ignoreres en etikette sætning.

## 5.12 EXIT - udhop af sætningsliste

En EXIT sætning benyttes til udhop af en løkke eller til returhop fra en procedure.

EXIT sætningen har følgende opbygning:

```
EXIT
```

EXIT sætningen bevirker et udhop fra den sidst aktiverede sammensatte sætning, og programudførelsen fortsætter med sætningen umiddelbart efter afslutningen af den sammensatte sætning som angivet nedenfor:

| Sætningstype | Eksekveringen fortsætter med sætningen efter |
|---|---|
| procedure | EXEC |
| FOR-NEXT | NEXT |
| WHILE-ENDWHILE | ENDWHILE |
| REPEAT-UNTIL | UNTIL |

Bemærk at EXIT sætningen ikke kan anvendes til udhop fra IF eller CASE sætninger, men vil afslutte den omliggende iteration eller procedure. Udførelsen af en EXIT sætning kan således gøres betinget.

Hvis man udfører en EXIT sætning på et tidspunkt, hvor DDE-COMAL-80 ikke er inde i en sammensat sætning, bliver kørslen afbrudt med en fejlmeddelelse.

Eksempel:

```
0100 // UENDELIG LØKKE MED UDHOP
0110 FALSE:=0
0120 REPEAT
0130   INPUT "INDTAST TAL, 0 FOR UDHOP ":TAL
0140   IF TAL=0 THEN EXIT
0150   PRINT TAL
0160 UNTIL FALSE
0170 // VED EXIT HOPPES HERTIL
0180 PRINT "SLUT"
0190 END
0200
*RUN
INDTAST TAL, 0 FOR UDHOP 7
 7.000000000000
INDTAST TAL, 0 FOR UDHOP -5
-5.000000000000
```

```
INDTAST TAL, 0 FOR UDHOP 0
SLUT
```

## 5.13 FOR-NEXT - udfør en sætning et antal gange

FOR-NEXT sætningen benyttes, når en række sætninger skal udføres et bestemt antal gange.

FOR - NEXT sætningen har følgende opbygning:

```
  FOR kontrolvariabel:=startværdi TO slutværdi STEP trinværdi DO sætning
eller
  FOR kontrolvariabel:=startværdi TO slutværdi STEP trinværdi DO
    sætningsliste
  NEXT kontrolvariabel
```

I FOR-sætningen kan 'STEP trinværdi' udelades. Hvis trinværdien ikke er opgivet, sættes den til 1.

Kontrolvariablen skal være en simpel talvariabel, mens startværdien, slutværdien og trinværdien skal være aritmetiske udtryk.

Ved starten af en FOR sætning vil kontrolvariablen få tildelt startværdien. Derefter testes om de opgivne start-, slut- og trinværdier har nogen mening. Hvis startværdi <= slutværdi skal trinværdien være positiv, og hvis startværdi >= slutværdi skal trinværdien være negativ, eller kort: (slutværdi - startværdi) * fortegn(trinværdi) >= 0. Hvis denne betingelse ikke er opfyldt, overspringes FOR - NEXT sætningen, og man undgår således at udføre en uendelig løkke. Hvis betingelsen er opfyldt, udføres sætningen.

Sætningslisten må indeholde vilkårlige sætninger. Hvis sætningslisten indeholder sammensatte sætninger, skal disse sætninger være afsluttede inden NEXT sætningen.

Når en NEXT sætning udføres, testes først, om den simple variable angivet efter NEXT er identisk med styrevariablen i den tilhørende FOR sætning. Er dette ikke tilfældet, afsluttes kørslen med en fejlmeddelelse. Ellers udregnes trinværdien, der adderes til værdien af styrevariablen. Såfremt styrevariablens nye værdi ligger i området mellem startværdien og slutværdien fortsættes programudførelsen med første sætning efter FOR sætningen. I modsat tilfælde fortsættes med sætningen efter NEXT sætningen.

Det er tilladt at ændre værdien af styrevariablen såvel som slut- og

trinværdien i sætningslisten mellem FOR og NEXT. Dette må dog frarådes, da man let kan miste overblikket.

Efter afslutning af en FOR - NEXT sætning er værdien af kontrolvariablen udefineret.

Eksempel:

```
0090 INTEGER SUM
0100 SUM:=0
0110 FOR TAL:=1 TO 500 DO
0120   SUM:=SUM+TAL
0130 NEXT TAL
0140 PRINT "SUMMEN AF TALLENE FRA 1 TIL 500 ER";SUM
0150 END
0160
*RUN
SUMMEN AF TALLENE FRA 1 TIL 500 ER 125250.0000000
```

Eksempel:

```
0090 INTEGER X,Y,Z
0100 FOR X:=100 TO 200 STEP 100 DO
0110   FOR Y:=10 TO 35 STEP X/10 DO
0120     FOR Z:=1 TO 2 DO
0130       PRINT X,Y,Z
0140     NEXT Z
0150   NEXT Y
0160 NEXT X
0170 END
0180
*RUN
100.0000000000      10.00000000000      1.000000000000
100.0000000000      10.00000000000      2.000000000000
100.0000000000      10.00000000000      1.000000000000
100.0000000000      20.00000000000      2.000000000000
100.0000000000      30.00000000000      1.000000000000
100.0000000000      30.00000000000      2.000000000000
200.0000000000      10.00000000000      1.000000000000
200.0000000000      10.00000000000      2.000000000000
200.0000000000      30.00000000000      1.000000000000
200.0000000000      30.00000000000      2.000000000000
```

## 5.14 GOTO - hop til etikette.

GOTO sætningen benyttes til at bryde den normale sekventielle programudførelse for at fortsætte programudførelsen med en bestemt sætning.

GOTO sætningen har følgende opbygning:

```
GOTO etikette
```

En GOTO sætning bevirker, at programudførelsen fortsætter i sætningen der følger efter erklæringen af etikette. Findes etiketten ikke, afbrydes programudførelsen med en fejludskrift.

Det er ikke tilladt at anvende GOTO sætningen til indhop i en sammensat sætning eller udhop af en procedure. Forsøges dette afbrydes kørslen med en fejlmeddelelse. Det er derimod tilladt at benytte en GOTO sætning til udhop fra en sammensat sætning, og etiketten hvortil der hoppes må gerne være placeret i en omkringliggende sammensat sætning.

Eksempel:

```
0010 REAL BELØB(10)
0020 I:=1
0025 LAB:
0030 INPUT "INDTAST BELØB":BELØB(I)
0040 I:=I+1
0050 IF I<=10 THEN GOTO LAB
0060 PRINT "10 BELØB ER NU INDTASTET"
0070 END
```

De fleste GOTO konstruktioner kan erstattes af enten en FOR - NEXT (se afsnit 5.13), en REPEAT - UNTIL (se afsnit 5.27) eller en WHILE - ENDWHILE (se afsnit 5.32). Disse sætningstyper bør anvendes, fordi det gør programmerne mere overskuelige.

## 5.15 IF-THEN-ELSE - betinget sætningsudførelse

DDE-COMAL-80 indeholder muligheder for at gøre udførelsen af en række sætninger betingede af værdien af et logisk udtryk. I den simple udgave udføres en enkelt sætning betinget af værdien af et logisk udtryk. Ønskes en række sætninger udført når en betingelse er opfyldt, skrives disse efter THEN og afsluttes med ENDIF. Endelig findes der IF-THEN- ELSE, som kan benyttes til at udføre en række sætninger, når en betingelse er opfyldt, og en anden når betingelsen ikke er opfyldt.

En betinget sætning har følgende opbygning:

```
              IF logisk udtryk THEN sætning
eller
              IF  logisk udtryk  THEN
                sætningsliste
              ENDIF

eller         IF  logisk udtryk  THEN
                sætningsliste
              ELSE
                sætningsliste
              ENDIF
```

Sætningslisterne må indeholde vilkårlige sætninger. Hvis en sætningsliste indeholder sammensatte sætninger, skal disse sætninger afsluttes inden for sætningslisten.

Ved udførelse af en betinget sætning udregnes det logiske udtryk efter IF. Hvis værdien af dette er sand, udføres sætningen eller sætningslisten efter THEN indtil det eventuelt tilhørende ELSE. Herefter fortsætter programudførelsen med den første sætning efter det tilhørende ENDIF.

Hvis det logiske udtryk efter IF har værdien falsk, afhænger DDE-COMAL-80s reaktion af den betingede sætnings form. Ved en simpel IF-sætning fortsættes med næste sætning. Ved en IF-THEN sætning fortsættes med sætningen umiddelbart efter det tilhørende ENDIF. Ved en IF-THEN-ELSE sætning fortsættes med sætningslisten efter ELSE.

Eksempler: IF sætning

```
0010 DIM ELEVNAVN$(10) OF 30
0020 FOR I:=1 TO 10
0030   INPUT "INDTAST ELEVNAVN:":ELEVNAVN$(I)
0040   IF ELEVNAVN$(I)="SLUT" THEN EXIT
0050 NEXT I
0060 END
```

IF-THEN-ELSE sætning:

```
0010 DIM JANEJ$ OF 3
0020 REPEAT
0030   INPUT "INDTAST JA ELLER NEJ:":JANEJ$
```

```
0040 UNTIL JANEJ$="JA" OR JANEJ$="NEJ"
0050 IF JANEJ$="JA" THEN
0060   PRINT "DU ER ELLERS IKKE BANGE AF DIG"
0070   PRINT "DU SKULLE VIDE HVAD DU HAR SAGT JA TIL"
0080 ELSE
0090   PRINT "HVAD ER DU FOR EN SKEPTIKER"
0100   PRINT "AFVISER FORSLAG FØR DU KENDER DEM"
0110 ENDIF
0120 END
```

## 5.16 IN/OUT - læs/skriv INTEL 8080 ind/ud port

IN og OUT sætningerne anvendes til at indlæse, henholdsvis udskrive, en værdi mellem 0 og 255 til en i/u port . Disse sætninger anvendes særlig til proceskontrol o.lign.

IN sætningen har følgende opbygning:

```
IN  portnummer , maske , simpel talvariabel
```

Portnummer og maske skal være heltalsudtryk med en værdi mellem 0 og 255.

IN sætningen medfører at DDE-COMAL-80 udfører en IN assemblerinstruktion på i/u porten med det angivne portnummer. Herefter beregnes logisk OG (AND) mellem den indlæste værdi og masken opfattet som et 8-bits heltal. Den resulterende værdi returneres i den simple variabel.

OUT sætningen har følgende opbygning:

```
OUT  portnummer , udværdi
```

Her skal portnummeret og udværdien være heltalsudtryk med en værdi mellem 0 og 255.

OUT sætningen medfører at DDE-COMAL-80 udfører en OUT assemblerinstruktion på i/u porten med det angivne portnummer og den angivne udværdi.

Ukyndige brugere advares kraftigt imod brugen af begge disse sætninger, idet de modsat resten af DDE-COMAL-80 systemet ikke er fuldt beskyttede. Dette betyder bl.a. at forkert brug af disse sætninger, særlig i flerbrugersystemer, kan bevirke at systemet umiddelbart eller på et senere tidspunkt begynder at opføre sig mærkeligt, evt. standser helt.

## 5.17 INTEGER - erklæring af heltalsvariable.

INTEGER sætningen benyttes til at erklære simple heltalsvariable eller indicerede (vektorer eller matricer) heltalsvariable. Erklæringer skal eksekveres inden de erklærede størrelser første gang benyttes i programmet.

INTEGER sætningen har følgende opbygning:

```
INTEGER erklæring , ... , erklæring
```

En erklæring kan være

```
                variabelnavn
eller           variabelnavn  ( heltalsudtryk )
eller           variabelnavn  ( heltalsudtryk, heltalsudtryk )
```

Den første form anvendes til erklæring af en simpel heltalsvariabel, den anden til erklæring af en heltalsvektor, og den sidste til erklæring af en heltalsmatrix. Det er tilladt at blande erklæringerne i een og samme INTEGER sætning.

Ved udførelse af INTEGER sætningen afsættes plads i dataområdet til de størrelser, der erklæres. Maksimumsstørrelsen for en indiceret variabel er kun begrænset af arbejdslagerets størrelse.

En variabel må kun erklæres een gang.

Eksempler:
```
INTEGER NETTOUDGIFT
INTEGER SUM,VEKTOR(15),MATRIX(3,3)
```

## 5.18 INPUT - indlæs data

INPUT sætningen benyttes til at indlæse værdier fra dataskærmen og tildele disse til talvariable og strengvariable.

INPUT sætningen har følgende opbygning:

```
INPUT ledetekst : variabel , ... , variabel ,
      ... , ledetekst : variabel , ... , variabel ;
```

Hele INPUT sætningen skal stå på een programlinie.

Ledetekst skal være en strengkonstant, mens variabel kan være en talvariabel,talvariabelelement, strengvariabel, strengvariabelelement eller en delstreng.

Udførelse af en INPUT sætning bevirker først at et evt. indhold af uddatabufferen fra en PRINT eller PRINT USING sætning udskrives (uddatabufferen tømmes).

Herefter gennemgås elementerne i INPUT sætningen fra venstre mod højre. Ledetekster udskrives direkte. Når der mødes en variabel, startes en indlæsning, hvorunder brugeren kan indtaste værdier i form af tal eller strengkonstanter. En talværdi skal overholde den i afsnit 4.1 anførte syntaks. Flere talværdier indtastet på samme linie adskilles med blanktegn. Der kan kun skrives een strengkonstant af gangen, dvs indtastningen af en strengkonstant afsluttes med tryk på RETURN. Strengkonstanter skal skrives uden omgivende '"'.

Der må højst skrives så mange værdier, som der er variable i listen indtil næste ledetekst.

Den indtastede linie sammenlignes med de variable i INPUT sætningen, og værdierne tildeles en for en. Hvis der efter tildelingen er værdier i inddatalisten, som ikke har fået tildelt en værdi, udskrives en evt. ny ledetekst hvorefter en ny indlæsning startes.

Hvis der hverken udskrives uddatabufferindhold eller ledetekst inden systemet starter en indlæsning, udskriver DDE-COMAL-80 et ':', når en ny indlæsning påbegyndes.

Inddatalinien kan højst rumme 70 tegn. Inddatalinien kan placeres hvorsomhelst på skærmen ved hjælp af CURSOR kommandoen (se afsnit 5.6).

Ledetekster og ':' udskrives altid på dataskærmen, også hvis uddata udskrives på linieskriveren.

Indtastede strengkonstanter omgives ikke af ". En indtastet strengkonstant starter enten umiddelbart efter ':' , eller umiddelbart efter det blanktegn, der følger efter en evt. foregående reel konstant. En strengkonstant omfatter altid resten af inddatalinien. Det er derfor ikke muligt at indtaste flere strengkonstanter på samme linie. Ved tildeling af en strengkonstant til den tilsvarende strengvariabel i variabellisten anvendes reglerne for tildelingssætninger, som er beskrevet i afsnit 5.18. Det er dog vigtigt at bemærke, at evt efterstillede blanktegn ignoreres ved denne tildeling, dvs at strengkonstanten kun omfatter tegnene fra indtastningens start til og med sidste ikke-blanktegn.

Hvis brugeren afslutter inddatalinien, inden der er indtastet en værdi for hver variabel i inddatalisten, vil systemet på næste linie udskrive et nyt ':' som tegn på at der forventes mere inddata.

Ved indtastning kan brugeren anvende de sædvanlige redigeringsfaciliteter. Indtastningen afsluttes ved at trykke på RETURN. Hvis indtastningen afsluttes ved at trykke på ESC afbrydes kørslen af programmet (se også afsnit 5.20).

Hvis der ikke indtastes et syntaktisk korrekt tal, når inddatalisten kræver et tal, vil skærmen udsende en hyletone. Markøren placeres på det fejlbehæftede tal, og brugeren kan rette fejlen og afsende de rettede data til systemet ved påny at trykke RETURN.

Eksempel:

```
0010 INPUT "STYKTAL ":STYKTAL," STYKPRIS ":STYKPRIS
0020 PRINT "TOTALPRIS:";STYKTAL*STYKPRIS
0030 END
0040
*RUN
STYKTAL 25 STYKPRIS 7.00
TOTALPRIS: 175.000000000
```

I dette eksempel er indtastningen sket ved at systemet først har udskrevet "STYKTAL ", hvorefter brugeren har indtastet 25 og trykket RETURN. Herefter har systemet umiddelbart til højre for 25 tilføjet " STYKPRIS ", hvorefter brugeren har indtastet 7.00 og trykket RETURN

Sluttegnet ';' kan udelades. Hvis det forekommer vil efterfølgende skrivning fortsætte på samme linie, ellers skiftes til første position på næste linie.

## 5.19 Kommentar

Kommentarsætningen bruges til at kommentere et program. Kommentarer er også tilladte efter alle DDE-COMAL-80 sætninger.

En kommentar har følgende opbygning:

```
// kommentar
```

En kommentar er en vilkårlig sekvens af ASCII tegn.

Kommentarer tjener alene til at gøre programmet mere forståeligt for brugeren. Kommentarer ignoreres under programudførelsen.

## 5.20 ON ESC - hop hvis ESC

ON ESC giver brugeren mulighed for at styre programudførelsen efter aktivering af ESC tasten.

```
ON  ESC  GOTO  etikette
```

Når ESC aktiveres, overføres kontrollen til linien efter den anførte etikette. Enhver aktivering af ESC tasten vil overføre kontrollen til denne linie, indtil en ny ON ESC sætning er udført. Herefter overføres kontrollen ved aktiverering af ESC tasten til linien udpeget i den nye ON ESC sætning.

Eksempel:
```
0005 INTEGER I
0010 ON ESC GO TO OUTLABEL
0020 I:=0
0030 WHILE I<100 DO
0040   I:=I+1
0050   PRINT I
0080 ENDWHILE
0085 OUTLABEL:
0090 END
```

Ved aktivering ESC tasten under eksekvering af WHILE løkken overføres kontrollen til linien efter OUTLABEL (linie 90).

I forbindelse med brugen af ON ESC er det vigtigt, at man respekterer strukturerne i programmet, idet der gælder de samme regler for ON ESC som for GOTO sætningen.

Sætningen ON ESC GOTO DUMMY sætter ESC tasten ud af funktion indtil en ny ON ESC sætning eksekveres eller indtil programudførelsen stopper. Aktiveringen af ESC tasten gemmes dog, således at den bliver behandlet når der åbnes for ESC tasten ved udførelsen af en ON ESC sætning.

Eksempel:
```
0005 INTEGER I
0010 ON ESC GOTO DUMMY
0020 FOR I:=0 TO 10 DO
0030   PRINT "A",I
0040 NEXT I
```

```
0050 ON ESC GOTO PRC
0060 PRINT "B"
0065 PRC:
0070 PRINT "C"
0080 END
```

Hvis man aktiverer ESC tasten under udskrivningen af A´erne vil linie 60 overspringes.

Hvis man efter en ON ESC GOTO DUMMY aktiverer ESC tasten som svar på en INPUT eller EDIT sætning gentages INPUT eller EDIT sætningen. Hvis ESC er deaktiveret kan INPUT og EDIT kun afsluttes ved at trykke på RETURN.

## 5.21 OUTPUT - vælg uddataenhed

OUTPUT sætningen anvendes til at bestemme, om udskriften fra et program (PRINT eller PRINT USING sætninger) skal sendes til dataskærmen eller skriveren.

OUTPUT sætningen har følgende opbygning:

OUTPUT strengudtryk

Værdien af strengudtrykket er bestemmende for hvor efterfølgende udskrift skrives. Hvis værdien er "T" eller "t" dirigeres udskriften til terminalen. Hvis værdien er "P1", "P2", "P3", "p1", "p2" eller "p3" dirigeres udskriften til henholdsvis skriver 1, 2 eller 3. Alle andre værdier af strengudtrykket vil resultere i en fejlmeddelelse.

DDE-COMAL-80 vil efter afslutningen af enhver eksekvering af et program udføre en implicit OUTPUT T kommando, dvs. at uddataenheden automatisk bliver dataskærmen. Hvis uddataenheden var skriveren, udskrives automatisk et sideskift på denne.

Skriveren reserveres først, når der er brug for den. Det betyder, at et brugerprogram, som skal anvende skriveren, først reserverer den, når uddatabufferen skal tømmes for første gang. Hvis skriveren er reserveret af en anden bruger, stoppes udførelsen, og der udskrives en fejlmeddelelse. I dette tilfælde kan brugeren genoptage kørslen ved at indtaste OUTPUT Px (OUTPUT som kommando se afsnit 8.14) efterfulgt af CONTINUE xxx, hvor xxx er linienummeret på den PRINT sætning, som forårsagede fejlen. Det kan ikke garanteres at uddatalinien i dette tilfælde udskrives korrekt.

## 5.22 PRINT - udskriv data

PRINT sætningen benyttes til udskrivning af resultater i form af tal eller tekststrenge i et fast format på dataskærm eller skriver (se afsnit 5.20 og 8.14).

PRINT sætningen har følgende opbygning:

```
                PRINT
eller           PRINT printelement skilletegn ... skilletegn
                      printelement sluttegn
```

Hele PRINT sætningen skal stå på een programlinie.

Skilletegn og sluttegn kan være et , (komma) eller et ; (semikolon), og sluttegnet kan udelades.

Et printelement kan være:

```
                aritmetisk udtryk
eller           strengudtryk
eller           TAB ( position )
```

Position skal være et heltalsudtryk.

PRINT sætningen konverterer data fra intern (binær) repræsentation til ASCII og overfører disse ASCII data til en uddatabuffer. Uddatabufferen udskrives (tømmes) på dataskærmen henholdsvis skriveren i følgende tilfælde:

- et printelement forsøges udskrevet, men uddatabufferen er fuld

- alle printelementer er behandlet

En PRINT sætning uden printelementer bevirker, at der udskrives et linieskift.

Skilletegnet mellem printelementerne har indflydelse på udskriftens placering på linien.

Anvendes skilletegnet ´,´ , opdeles linien i 4 søjler, der starter i position 1, 21, 41 og 61. Et printelement starter i næste frie søjle. Hvis der ikke er flere frie søjler på den aktuelle linie, tømmes uddatabufferen, hvorefter udskriften fortsætter i den første søjle på den næste linie. Printelementerne udskrives venstrestillede i søjlen.

Anvendes skilletegnet ´;´ vil udskriften fortsættes i den første frie position, dog således at der udskrives et blanktegn efter tal, mens tegnstrenge udskrives uden efterfølgende blanktegn.

For begge skilletegn gælder, at før et printelement udskrives, sammenlignes dets længde med længden af den resterende plads på den aktuelle uddatalinie. Hvis printelementet er et tal og ikke kan være på den aktuelle linie, tømmes uddatabufferen, og printelementet udskrives på næste linie. Hvis printelementet er en tegnstreng udskrives den del af elementet, der kan rummes på linien, og resten skrives på næste linie. Hvis printelementet er længere end 79 tegn (den maksimale linielængde), udskrives uddata over flere linier med 79 tegn pr. linie til alt er udskrevet.

Ved udskrift af tal gælder følgende regler:

- reelle tal med en absolut værdi mindre end 1E13 skrives med 13 cifre samt et decimalpunkt

- reelle tal med en absolut værdi større end 1E13 skrives med eksponentangivelse og mantissen med 13 betydende cifre, således at at der skrives eet ciffer forskellig fra 0 til venstre for decimalpunktet

- heltal skrives uden foranstillede nuller

- eksponenten udskrives som fx E+005 eller E-010, dvs med fortegn og tre cifre i eksponenten

Eksempler på udskrift af reelle tal og deres placering i en søjle:

```
Position    12345678901234567890
            25
            25.30000000000
            -2500
            0.000000000250
```

Sluttegnet i printlisten bruges til at afgøre, hvad der skal ske efter at alle printelementer er behandlede. Hvis der ikke er noget sluttegn vil næste PRINT eller PRINT USING sætning starte i første position på næste linie. Hvis sluttegnet er ´,´ eller ´;´ vil det første printelement i den næste PRINT sætning blive udskrevet på samme linie i overensstemmelse med reglerne for skilletegn mellem printelementer.

Funktionen TAB anvendes til at ændre startpositionen i uddatabufferen for de følgende printelementer til værdien af den opgivne position. Printpositionerne nummereres fra 1 til 79. Baglæns tabulering er tilladt, og kan

benyttes til overskrivning af tidligere udskrevne data, inden for samme PRINT sætning. Hvis positionen er større end 79, afbrydes programudførelsen med en fejlmeddelelse.

Eksempel:
```
0010 I:=10
0020 PRINT "12345678901234567890123456789012345678901234567"
0030 PRINT "TALLET",I
0040 PRINT "TALLET ";I;
0050 PRINT "vi kan fortsætte på samme linie"
0060 PRINT TAB(20);"TALLET ";I
0070 PRINT SIN(3.14),-3.0,2+I,"LINIESKIFT HVIS LINIEN ER FULD"
0080 END
0090
*RUN
12345678901234567890123456789012345678901234567
TALLET               10.00000000000
TALLET 10.00000000000 vi kan fortsætte på samme linie
                    TALLET 10.00000000000
0.001592652917      -3.000000000000     12.00000000000
LINIESKIFT HVIS LINIEN ER FULD
```

I visse situationer kan det være nødvendigt at udsende tegn til en dataskærm eller skriver, som ikke kan udtrykkes i strengkonstanter, fx styresekvenser, der får en dataskærm til at udsende en hyletone, eller får en skriver, som er indrettet hertil, til at ændre skrifttype. Sådanne styretegn udsendes lettest ved hjælp af CHR$ funktionen.

Eksempel: udsend hyletone fra dataskærm. Af manualen for skærmen fremgår at hyletonen har kodeværdien 7 (BELL). PRINT-sætningen har derfor følgende udseende:

```
0100 PRINT CHR$(7)
```

Bemærk at `0100 PRINT 7` giver et helt andet resultat.

I visse situationer kan det være ønskeligt at få data udskrevet umiddelbart efter PRINT sætningen uden at få udskrevet det afsluttende vognretur/linieskift som DDE-COMAL-80 normalt tilføjer. Dette kan gøres ved først på uddatalinien at udskrive tekststrengen "<S>", som dog ikke vil figurere i udskriften, men som fylder 3 tegn i udskriftsbufferen.

Eksempel: tegn en lodret streg på skærmen

```
0080 INTEGER I
0090 CLEAR
0100 FOR I:=1 TO 24 DO
0110   CURSOR 40,I
0120   PRINT "<S>!"
0130 NEXT I
```

Udskrift på enten skærm eller skriver kan styres af andre kommandoer til den respektive skærm eller skriver. Styringsmuligheder og kommandoerne er beskrevet i MIKADOS Operating System and Utility Guide. Det aktuelle styresystem vil automatisk omsætte disse kommandoer til en sekvens af tegn som skærmen eller skriveren vil reagere på, subsidiært ignorere kommandoen hvis skærmen eller skriveren ikke kan udføre den beordrede funktion. 36

## 5.23 PRINT USING - udskriv data formatteret

PRINT USING benyttes til udskrivning af resultater i form af tal eller tekststrenge i de tilfælde, hvor brugeren selv ønsker at styre formatet for udskriften. Udskriften sker på dataskærm eller skriver.

PRINT USING sætningen har følgende opbygning:

```
PRINT USING format : aritmetisk-udtryk skilletegn
            ... skilletegn aritmetisk-udtryk sluttegn
```

En PRINT USING sætning skal stå på een programlinie.

Skilletegn og sluttegn kan være , (komma) eller ; (semikolen), og sluttegnet kan udelades.

Skilletegnene har ingen indflydelse på formatteringen, mens sluttegnet fungerer på samme måde som i en PRINT sætning.

Format skal være et strengudtryk og må således gerne være en strengkonstant.

Formatet skal indeholde en beskrivelse af hvoreledes værdierne af de aritmetiske udtryk skal anbringes på linien. Tegnene i formatspecifikationen fortolkes på følgende måde:

```
£  cifferposition og fortegn
.  decimalpunktum (kun hvis omgivet af )
```

Alle andre tegn overføres direkte til udskriften

Hvis tallet inklusive fortegn er for stort til at være inden for det opgivne format, skrives *'er i de respektive positioner. Hvis tallet har flere decimaler end angivet i udskriftsformatet, foretages en afrunding inden udskrivningen. Bemærk, at der altid skal afsættes plads til et fortegn, også hvis tallet er ikke-negativt (i dette tilfælde efterlades fortegnspositionen blank).

PRINT USING sætningen udføres ved, at formatstrengen gennemløbes tegn for tegn fra venstre mod højre (lave mod høje indeks). Alle tegn undtagen £ og . efter £ overføres direkte til uddatabufferen. Hver gang DDE-COMAL-80 møder en sekvens, som indledes med et £, konverteres og udskrives det næste aritmetiske udtryk i listen i overensstemmelse med det opgivne format. Sekvensen afsluttes ved første efterfølgende tegn, der ikke er £ eller . . Hvis der ikke er flere udtryk, standses udførelsen af PRINT USING sætningen, dvs. resten af formatstrengen ignoreres.

Udtryk, som ikke er udskrevet, når hele formatet er gennemløbet, ignoreres. Hvis et eller flere udtryk ikke udskrives, anvendes skilletegnet efter det sidst udskrevne aritmetiske udtryk som afslutningstegn.

Eksempel:
```
0010 LET A:=0.5
0020 PRINT USING "1:£££.££ **2 ER ££.££":A,A^2
0050 LET A:=11
0060 PRINT USING "2:£££.££ ** 2 ER ££.££":A,A^2
0070 END
0080
*RUN
1:  0.50 **2 ER  0.25
2: 11.00 **2 ER *****
```

## 5.24 PROC-ENDPROC - definer procedure

PROC og ENDPROC sætningerne benyttes til at erklære en procedure.

Erklæringen af en procedure har følgende opbygning:

```
PROC procedurenavn ( parameter , ... , parameter )
  sætningsliste
ENDPROC procedurenavn
```

Procedurenavn er en identifikator.

Procedurenavnet efter ENDPROC skal være det samme som i PROCEDURE sætningen. Hvis navnene er forskellige afbrydes kørslen med en fejlmeddelelse ved eksekveringen af ENDPROC.

Parameterlisten inklusive de omgivende parenteser kan udelades. En parameter (betegnes i forbindelse med procedureerklæringer en formel parameter) kan være:

talvariabel
REF talvariabel
REF talvariabel ()
REF talvariabel (,)
strengvariabel
REF strengvariabel
REF strengvariabel ()

Procedureerklæringen fastlægger, at den opgivne sætningsliste skal opfattes som en procedure med det navn, der er angivet umiddelbart efter PROC.

En procedureerklæring kan placeres et vilkårligt sted i programmet. Når DDE-COMAL-80 møder en PROC sætning, overspringes de følgende sætninger til en ENDPROC sætning mødes, hvorefter programudførelsen fortsættes.

Det er tilladt, at der forekommer procedurekald i sætningslisten, og rekursive kald er tilladte.

En procedure aktiveres ved at udføre en EXEC sætning. Data overføres mellem det kaldende program og proceduren gennem parametre eller via globale varaible.

De aktuelle parametre specificeres i EXEC sætningen. Antallet af formelle og aktuelle parametre skal være ens, og typen af de aktuelle og formelle parametre skal stemme overens parameter for parameter efter følgende regler:

| Formel parameter | Aktuel parameter | Overføres som |
|---|---|---|
| simpel talvariabel | aritmetisk udtryk | værdi |
| REF simpel talvariabel | simpel talvariabel | reference |
| REF talvariabel () | Enkelt indiceret talvariabel | reference |
| REF talvariabel (,) | dobbelt indiceret talvariabel | reference |
| simpel strengvariabel | strengudtryk | værdi |
| REF simpel strengvariabel | simpel strengvariabel | reference |
| REF strengvariabel () | indiceret strengvariabel | reference |

Når EXEC udføres får de formelle parametre værdier (overføres som værdi) eller bliver erstattet af de aktuelle parametre (overføres som reference) afhængig af typen af den korresponderende formelle parameter, som beskrevet ovenfor.

Formelle parameternavne i en procedure erklæring må ikke benyttes som variable uden for proceduren og må ikke benyttes som formelle parameternavne i andre procedurer.

Returhop fra en procedure sker når DDE-COMAL-80 eksekverer en EXIT eller en ENDPROC sætning. Herved overføres kontrollen til sætningen. der følger efter EXEC.

Aktivering af en procedure kan også foretages som et funktionskald i et aritmetisk udtryk. Benyttes en procedure på denne måde bør proceduren indeholde mindst een tildeling, hvor procedurenavnet optræder på venstresiden af tildelingsoperatoren ´:=´. Ved afslutning af en procedure kaldt som en funktion fortsættes med evalueringen af det aritmetisk udtryk, hvori kaldet indgår. Bemærk at hvis man benytter globale variable i proceduren kan det medføre utilsigtede sideeffekter.

Eksempel:

```
0010 DIM NAVN$ OF 20, NNAAVVNN$ OF 40
0020 NAVN$:="ERIK RASMUSSEN"
0030 EXEC UDVIDETSKRIFT
0040 NAVN$:="RASMUS ERIKSEN"
0050 EXEC UDVIDETSKRIFT
0060 // PROCEDURE STARTER HER
0070 PROC UDVIDETSKRIFT
0080   // BLANKE EFTERSTILLES NAVNET
0090   NAVN$:= NAVN$+"                    "
0100   FOR I:=1 TO 20 DO
0110     NNAAVVNN$(2*I-1):=" "
0120     NNAAVVNN$(2*I):=NAVN$(I)
0130   NEXT I
0140   PRINT NNAAVVNN$
0150 ENDPROC UDVIDETSKRIFT
0160 END
0170
*RUN
 E R I K   R A S M U S S E N
 R A S M U S   E R I K S E N
```

Yderligere eksempler på procedurekald findes i afsnit 5.10.

## 5.25 READ - indlæs faste data

READ sætningen benyttes til at tildele startværdier til talvariable og strengvariable.

READ sætningen har følgende opbygning:

```
READ variabel , ... , variabel
```

De ovenfor anførte variable er enten talvariable eller strengvariable.

Ved udførelsen af READ sætningen vil de variable i variabellisten få tildelt værdier fra de næste værdier i datalisten. Opbygningen af datalisten er beskrevet i afsnit 5.7.

Hvis typen af den læste værdi ikke stemmer med den angivne variabels type, eller hvis datalisten er udtømt, stoppes programudførelsen med en fejludskrift.

Eksempel:

```
0010 DIM TEKST$ OF 6
0015 INTEGER I,A
0020 DATA "DET ","VIRKER",7,9,13
0030 READ TEKST$
0040 PRINT TEKST$;
0050 READ TEKST$
0060 PRINT TEKST$
0070 FOR I:=1 TO 3 DO
0080   READ A
0090   PRINT A;
0100 NEXT I
0110 END
0120
*RUN
DET VIRKER
7.000000000000 9.000000000000 13.00000000000
```

## 5.26 REAL - erklæring af reelle variable

REAL sætningen benyttes til at erklære simple reelle variable eller indicerede (vektorer eller matricer) reelle variable. Erklæringer skal eksekveres inden de erklærede størrelser første gang benyttes i programmet.

REAL sætningen har følgende opbygning:

```
REAL erklæring , ... , erklæring
```

En erklæring kan være

```
                variabelnavn
eller           variabelnavn  ( heltalsudtryk )
eller           variabelnavn  ( heltalsudtryk, heltalsudtryk )
```

Den første form anvendes til erklæring af en simpel reel variable, den anden til erklæring af en reel vektor, og den sidste til erklæring af en reel matrix. Det er tilladt at blande erklæringerne i een og samme REAL sætning.

Ved udførelse af REAL sætningen afsættes plads i dataområdet til de størrelser, der erklæres. Maksimumsstørrelsen for en indiceret variabel er kun begrænset af arbejdslagerets størrelse.

En variabel må kun erklæres een gang.

Eksempler:
```
REAL NETTOUDGIFT
REAL SUM,VEKTOR(15),MATRIX(3,3)
```

## 5.27 REPEAT-UNTIL - udfør løkke indtil betingelse opfyldt

REPEAT - UNTIL sætningen benyttes, når en række sætninger skal udføres indtil en betingelse er opfyldt.

REPEAT - UNTIL sætningen har følgende opbygning:

```
REPEAT
  sætningsliste
UNTIL logisk udtryk
```

Sætningslisten må indeholde vilkårlige sætninger. Hvis sætningslisten indeholder en eller flere sammensatte sætninger, skal disse sætninger være afsluttede inden UNTIL sætningen.

En REPEAT sætning medfører ingen særlig aktion set fra programmørens synspunkt.

Når en UNTIL sætning udføres, udregnes det logiske udtryk efter UNTIL. Hvis det logiske udtryk er sandt, fortsætter udførelsen med den næste sætning efter UNTIL sætningen. Hvis det logiske udtryk er falsk, fortsættes med sætningen umiddelbart efter den REPEAT sætning, der svarer til UNTIL sætningen.

Eksempel:

```
0100 // BEREGNING AF KVADRATRØDDER
0110 X:=10
0130 SQRT:=X/2
0140 // ITERATION
0150 REPEAT
0160   DELTA:=(X/SQRT-SQRT)/2
0170   SQRT:=SQRT+DELTA
0180 UNTIL ABS(DELTA)<0.001
0190 PRINT SQRT
0200 END
0210
*RUN
3.162277660444
```

Sammenlign dette eksempel med eksemplet i afsnit 5.33

## 5.28 RESTORE - genstart indlæsning af faste data

RESTORE sætningen benyttes til at genstarte indlæsningen af data fra DATA sætningerne. Efter udførelsen af en RESTORE sætning vil den første variabel i den første READ sætning efter RESTORE sætningen få tildelt den første værdi fra den første DATA sætning i programmet osv.

RESTORE sætningen har følgende opbygning:

```
RESTORE
```

## 5.29 SELECT OUTPUT - vælg uddataenhed

SELECT OUTPUT sætningen anvendes til at bestemme om udskriften skal sendes til skærmen eller skriveren.

SELECT OUTPUT sætningen har følgende opbygning:

```
SELECT OUTPUT strengudtryk
```

SELECT OUTPUT svarer funktionsmæssigt til OUTPUT sætningen.

## 5.30 STOP/END - stands eksekvering af program

STOP og END sætningerne anvendes til at stoppe udførelsen af et program.

STOP og END sætningerne har følgende opbygning:

```
              STOP
henholdsvis   END
```

Udførelse af en STOP eller END sætning bevirker, at DDE-COMAL-80 systemet stopper programudførelsen, og man kan herefter indtaste kommandoer eller programlinier. Når der udføres en STOP sætning, udskrives desuden følgende meddelelse på terminalen:

```
STOP EFTER LINIE xxxx
```

hvor xxxx erstattes med linienummeret for den udførte STOP sætning.

DDE-COMAL-80 kræver ikke, at brugeren eksplicit angiver den fysiske afslutning på et program ved hjælp af en END sætning. DDE-COMAL-80 placerer automatisk en implicit END sætning efter den sidste programlinie i et program.

Bortset fra udskrivningen af STOP meddelelsen, har STOP og END sætningerne nøjagtig samme effekt i DDE-COMAL-80.

## 5.31 Tildeling

Tildelingssætninger benyttes til at tildele værdier til talvariable og strengvariable.

En tildelingssætning har følgende opbygning:

```
tildeling ; ... ; tildeling
```

En tildeling har følgende opbygning:

```
      talvariabel , ... , talvariabel := aritmetisk udtryk
eller strengvariabel := strengudtryk
eller procedurenavn := aritmetisk udtryk
```

Hvis der optræder flere talvariable på venstresiden af ':=' skal de alle være af samme type - enten reelle eller heltalsvariable.

De fleste tildelingssætninger i et program er simple tildelingssætninger af formen:

```
      talvariabel     := aritmetisk udtryk
eller strengvariabel := strengudtryk
```

I aritmetiske tildelingssætninger vil alle variable på venstre side af ':=' få tildelt værdien af det aritmetiske udtryk.

Eksempel:
```
VAR1,VAR2,VAR3 := 0
VAR1:=0;VAR2:=0;VAR3:=0
```

De to sætninger udfører det samme.

Tildelinger hvor venstresiden er et procedurenavn benyttes i procedurer kaldt som funktioner.

For strengtildelinger gælder følgende regler (som også gælder ved tildelinger i EDIT, INPUT, READ og GET sætningerne):

Såfremt strengudtrykket er længere end strengvariablen vil strengudtrykket blive afkortet ved at tilpas mange tegn, regnet fra højre, bortkastes. Kun det antal tegn, der passer ind i strengvariablen, vil blive overført.

Såfremt strengudtrykket har samme længde som strengvariablen sker overførslen direkte uden sideeffekter.

Såfremt strengudtrykket er kortere end strengvariablen, overføres strengudtrykket direkte og venstrestillet. Hvis strengvariablen er forsynet med en delstrengsangivelse udfyldes den resterende del af delstrengen med blanktegn. Hvis strengvariablen ikke er forsynet med en delstrengsangivelse, vil der blive placeret et slutmærke umiddelbart efter det sidste tegn, som overføres til strengvariablen. De resterende tegn i strengvariablen vil i dette tilfælde være udefinerede.

Eksempel:
```
DIM A$ OF 5, B$ OF 5, C$ OF 5
A$:="ABCDEF"
B$:=A$
C$:=A$
A$:="GH"
B$(1:2):="GH"
C$(1:4):="GHI"
```

Herefter er A$ = "GH", B$ = "GHCDE" og C$ = "GH E".
Indholdet af tegn 4-5 i A$ er udefineret.

I en strengtildeling må strengvariablen gerne indgå i strengudtrykket. Følgende tildeling er ikke alene tilladelig, men også hensigtsmæssig:

```
LET STRING$:=STRING$+"ER"
```

Sætningen har den virkning, at "ER" bliver placeret efter de tegn, der tidligere er blevet placeret i STRING$.

Hvis strengvariablen er en delstreng skal startpositionen være mindre end eller lig strengens maksimale længde og startpositionen + delstrengslængde - 1 skal være mindre end eller lig strengens maksimale længde.

## 5.32 WHILE-ENDWHILE - udfør løkke sålænge betingelse opfyldt

WHILE - ENDWHILE sætningen benyttes, når en række sætninger skal udføres sålænge en betingelse er opfyldt.

WHILE - ENDWHILE sætningen har følgende opbygning:

```
        WHILE logisk udtryk DO sætning
eller   WHILE logisk udtryk DO
          sætningsliste
        ENDWHILE
```

Sætningslisten må indeholde vilkårlige sætninger. Hvis sætningslisten indeholder en eller flere sammensatte sætninger, skal disse sætninger være afsluttede inden ENDWHILE sætningen.

Når en WHILE sætning udføres, udregnes værdien af det logiske udtryk efter WHILE. Hvis det logiske udtryk er sandt, fortsættes programudførelsen med sætningen efter WHILE. Hvis det logiske udtryk er falsk, fortsættes udførelsen med sætningen efter den tilhørende ENDWHILE sætning subsidiært sætningen efter WHILE sætningen.

Udførelsen af en ENDWHILE sætning bevirker udregning af det logiske udtryk i den tilhørende WHILE sætning, hvorefter programudførelsen fortsætter som beskrevet ovenfor.

Eksempel:
```
0100 // BEREGNING AF KVADRATRØDDER
0110 X:=10
```

```
0120 DELTA:=X
0130 SQRT:=X/2
0140 // ITERATION
0150 WHILE ABS(DELTA)>0.001 DO
0160   DELTA:=(X/SQRT-SQRT)/2
0170   SQRT:=SQRT+DELTA
0180 ENDWHILE
0190 PRINT SQRT
0200 END
0210
*RUN
3.162277660444
```

# 6. DDE-COMAL-80 filsystem

Dette kapitel beskriver syntaksen for de DDE-COMAL-80 sætninger, som bruges til at oprette, åbne, skrive, læse, teste, afslutte og lukke sekventielle og direkte tilgængelige filer.

Andre filsystem funktioner kan udføres ved hjælp af et antal særlige anvendelsesprogrammer, som er beskrevet i kapitel 9.

Overalt i dette kapitel benyttes termen ´etikette´ for ´pladeetikette´, hvis intet andet udtrykkeligt er nævnt.

## 6.1 Oversigt

Filinformationen, som kan behandles af DDE-COMAL-80, er lagret på pladelagerenheder. Der kan være en eller flere pladelagerenheder i et system, og pladelagerenhederne behøver ikke at være af samme type. Den samlede pladelagerkapacitet i et DDE-COMAL-80 system kan være på op til 80 millioner tegn.

Hver pladelagerenhed kan indeholde en eller flere plader (disketter). I forbindelse med DDE-COMAL-80 systemet anvendes normalt kun pladelagerenheder, som indeholder netop een plade hver. Pladerne kan være faste eller udskiftelige. Pladekapaciteten i et DDE-COMAL-80 system kan beregnes ud fra oplysningerne i tabel 6.1.

### 6.1.1 Pladelagre

Set fra DDE-COMAL-80 systemet består en plade af en pladeetikette, som indeholder pladens navn, et katalog, som indeholder en oversigt over filerne på pladen, samt et filområde. Katalogets omtrentlige størrelse for en række pladetyper fremgår af tabel 6.1. Etiketten og kataloget optager så lidt plads på pladen, at man normalt kan regne med, at filområdets størrelse er identisk med pladens størrelse.

| Pladelagertype | antal plader | antal sektorer | kapacitet 1000 tegn | katalog størrelse max. antal filer |
|---|---|---|---|---|
| 10 Mb Winchester | 2 | 19200 | 4784 | 9728 |
| 5 Mb Winchester | 2 | 10304 | 2637 | 4864 |
| 1 Mb diskette | 1 | 4004 | 959 | 4864 |

| | | | | |
|---|---|---|---|---|
| 560 Kb mini disk (dobb.sid.dobb.tæth.dobb.sportæth.) | 1 | 2112 | 540 | 2432 |
| 280 Kb mini disk (dobb.sid.dobb.tæth.) | 1 | 1120 | 270 | 1216 |

Antal sektorer, kapacitet og katalogstørrelse gælder pr. plade. Kapaciteten er beregnet eksklusiv den plads kataloget optager.

Tabel 6.1

### 6.1.2 Filidentifikation

En fil identificeres ved etiketten på den plade, hvor filen er lagret, samt filnavnet. Identifikatoren skal opbygges som

etikette : filnavn

Både etiketten og filnavnet skal være identifikatorer. Identifikatorerne svarende til etiketter og filnavne må højst være på 8 tegn. Der må ikke forekomme blanktegn i en filidentifikation.

Filens identifikation skal være indeholdt i en filidentifikator, der kan være:

- en strengkonstant, som altså skal opbygges som:
  " etikette : filnavn "
- en simpel eller indiceret strengvariabel, som skal indeholde en tekst på formen:
  etikette : filnavn

Et eksempel på et lovligt filnavn er "JENS:TESTPROG", som henviser til filen med navnet TESTPROG på pladen med etiketten JENS.

Hvis der i et DDE-COMAL-80 system med to eller flere pladelagerenheder monteres to eller flere plader med samme etikette, vil DDE-COMAL-80 kun kunne anvende pladen i pladelagerenheden med det laveste nummer. Pladelagernumrene opgives ved leveringen, men normalt har den pladelagerenhed længst til venstre / øverst nummer 1, osv.

### 6.1.3 Filtyper

DDE-COMAL-80 arbejder med følgende 3 typer af filer:

Type K: programtekstfiler, sekventielle datafiler, direkte datafiler
Type B: binære programfiler
Type O: filer indeholdende programmer der ikke er skrevet i DDE-COMAL-80 (se beskrivelsen af CALL sætningen, afsnit 5.2)

DDE-COMAL-80 programmer kan arbejde med datafiler og programtekstfiler.

Det er tilladt på en plade at have flere filer af samme navn, blot de har forskellig type.

### 6.1.4 Initialisering af plader

Inden en ny plade kan anvendes af DDE-COMAL-80, skal den formatteres og initialiseres. Formatteringen foretages af DDE, inden disketten leveres, mens brugeren normalt selv må sørge for at initialisere pladen.

Programmet INITLIZE anvendes til at initialisere en plade, se afsnit 9.2, og kan tillige anvendes til at slette indholdet af en plade og/eller forsyne den med en ny etikette. Etikettebetegnelsen for en plade, hvis etikette man har glemt, kan vises ved hjælp af CATALOG kommandoen, se afsnit 8.2.

## 6.2 Datafiler

Datafiler kan være organiseret sekventielt eller direkte. Organisationsformen for en fil fastlægges ved oprettelsen af filen.

I en sekventiel datafil kan data kun læses eller skrives i en bestemt rækkefølge. Når filen åbnes, har brugeren direkte mulighed for at læse eller skrive den første post i filen. Når den første post er læst eller skrevet, kan brugeren læse eller skrive den næste post osv. For at få adgang til post nummer n kræves således (n-1) læse- eller skriveoperationer. En post i en sekventiel fil fylder det samme som den information, der er i posten, dvs. poster i sekventielle filer kan have variabel længde. Fordelen ved en sekventiel fil er således god pladsudnyttelse, og ulempen er langsom tilgang til de enkelte poster. Ajourføring af poster i en sekventiel fil er kun mulig, hvis den nye post har nøjagtig den samme

længde som den gamle post. Hvis postlængderne ikke er ens, ødelægges informationen efter den ajourførte post. Derimod er det altid muligt at tilføje nye poster til en eksisterende sekventiel fil. Dette kræver blot, at man læser frem til slutmærket, hvorefter skrivningen påbegyndes (se afsnit 6.8).

En sekventiel fil kan udvides indtil 60 gange, dersom pladsen på pladen tillader dette. Filudvidelserne er alle af samme størrelse som den originale fil.

I en direkte datafil kan man umiddelbart læse eller skrive en vilkårlig post. Alle poster i filen har samme længde, og en direkte fil kan derfor sammenlignes med en kartoteksæske, hvor hver post svarer til et kartotekskort. Fordelen ved en direkte fil er en meget hurtig tilgang til informationen i en bestemt post, mens ulempen er, at alle poster fylder det samme, dvs. at der ofte vil være et vist pladsspild.

Antallet af poster i en direkte fil fastlægges, når filen oprettes. Det er ikke muligt siden at udvide filen.

Den maksimalt tilladte postlængde er 77 tegn for sekventielle filer og 188 tegn for direkte filer.

De første 32 oktetter i den første sektor i hver fil og i hver efterfølgende filudvidelse benyttes af systemet og er ikke tilgængelige for brugeren.

## 6.3 CREATE sætning

CREATE sætningen anvendes til at oprette en datafil.

CREATE sætningen har følgende opbygning:

```
          CREATE  filidentifikator
eller     CREATE  filidentifikator , antal-sektorer
eller     CREATE  filidentifikator , antal-poster , postlængde
```

CREATE kommandoen opretter datafilen på den plade og under det navn, som specificeres af filidentifikatoren.

Antal-sektorer, antal-poster og postlængde skal være heltalsudtryk.

Den første og anden form af CREATE kommandoen anvendes til at oprette en sekventiel fil. Filstørrelsen er i det første tilfælde altid 2528 oktet-

ter (10 sektorer a 256 oktetter - 32 oktetter). I det andet tilfælde angiver antal-sektorer, hvor mange sektorer (1 sektor = 256 oktetter) filen skal fylde. Man skal dog være opmærksom på, at systemet benytter 32 oktetter af den afsatte plads. Sekventielle filer udvides automatisk når behovet opstår, se afsnit 6.2.

Den tredie form af CREATE kommandoen anvendes til at oprette en direkte fil. Filen kan rumme mindst det antal poster, som angives af antal-poster (i visse tilfælde vil filen kunne rumme nogle få poster mere end angivet). Det største antal poster i en direkte fil er 32767. Antallet af oktetter i hver post angives af postlængde. Den ønskede postlængde kan beregnes, idet et reelt tal fylder 8 oktetter, et heltal 2 tegn og en tegnstreng fylder strengens længde plus 1 oktet. Den mindste tilladte postlængde er 4 tegn og den største er 188 tegn. Direkte filer kan ikke udvides.

Efter en CREATE operation bør brugeren ved hjælp af STATUS funktionen (se afsnit 6.7) kontrollere om filen er blevet oprettet.

## 6.4 OPEN sætning

OPEN sætningen anvendes til at åbne en forbindelseslinie mellem DDE-COMAL-80 og en bestemt fil på en bestemt plade. En fil skal åbnes, inden poster fra den kan læses eller skrives.

OPEN sætningen har følgende opbygning:

```
              OPEN  filidentifikator , R
eller         OPEN  filidentifikator , W
```

OPEN sætningen bevirker, at DDE-COMAL-80 undersøger om der findes en plade med den opgivne etikette, samt om denne plade indeholder en fil med det opgivne filnavn. Hvis dette er tilfældet, etableres den ønskede forbindelse, hvorefter brugeren kan udføre GET og/eller PUT ordrer på filen.

De tilladte tilgangsmåder er

R - Read, kun læsning tilladt
W - Write, læsning og skrivning tilladt

Flere brugere kan benytte den samme fil på en gang, hvis alle anvender tilgangsmåde R. Derimod har kun een bruger adgang til filen, hvis den åbnes med tilgangsmåde W. Hvis en bruger søger at åbne en fil på en sådan måde, at der vil opstå en konflikt med en anden brugers aktive tilgang

til filen, afslås OPEN operationen, og der returneres en særlig resultatkode, som kan inspiceres ved hjælp af STATUS.

Et DDE-COMAL-80 program kan have flere filer åbnet samtidig. Det maksimale antal filer, som kan være åbne samtidig, afhænger af det aktuelle system og fastlægges på systemgenereringstidspunktet.

Når en sekventiel fil åbnes, er den positioneret til den første post i filen. En åben sekventiel fil kan repositioneres til post 1 ved at lukke og derpå åbne filen.

Efter en OPEN operation bør brugeren ved hjælp af STATUS funktionen (se afsnit 6.7) kontrollere, om filen er blevet åbnet.

## 6.5 GET sætning

GET sætningen bruges til at indlæse en post fra en fil og overføre indholdet til en eller flere DDE-COMAL-80 variable.

GET sætningen har følgende opbygning:

```
GET filidentifikator : variabel , ... , variabel
eller
GET filidentifikator , postnummer : variabel , ... , variabel
```

Postnummeret skal være et heltalsudtryk. De anførte variable skal være enten talvariable eller strengvariable.

De to former for GET sætningen svarer til sekventielle hhv. direkte filer. Ved en sekventiel fil indlæses den næste post i filen, ved en direkte fil indlæses posten med det opgivne postnummer.

Når DDE-COMAL-80 har indlæst posten, overføres data fra posten til de variable, som er anført i variabellisten. Hver variabel i variabellisten bør være af den samme type som den tilsvarende variabel i den variabelliste, der blev anvendt ved skrivning af posten. Manglende overensstemmelse mellem variabellisterne, anvendt ved indlæsning og udskrift, opdages ikke umiddelbart af DDE-COMAL-80, men kan give anledning til fejl senere i programudførelsen.

Ved tildeling af en strengkonstant til den tilsvarende strengvariabel i variabellisten anvendes reglerne for tildelingssætninger, som er beskrevet i afsnit 5.31.

Hvis variabellisten kræver flere data, end der er i posten, afbrydes udførelsen med en fejlmeddelelse. Hvis variabellisten kræver færre data, end der er i posten, ignoreres overskydende data.

Efter en GET operation bør brugeren ved hjælp af STATUS funktionen kontrollere, om der er indlæst en post, om slutmærket er nået (kun ved sekventielle filer), eller om der er konstateret andre fejl.

Eksempel:

```
0010 // PROGRAMMET LÆSER DATA FRA DEN SEKVENTIELLE FIL
0020 // "DATAFIL1" LIGGENDE PÅ DISKETTEN MED ETIKETTEN
0030 // "DISK1" OG UDSKRIVER DEM.
0040 DIM DATAVAR$ OF 40,FILNAVN$ OF 17
0045 INTEGER FÆRDIG,ENDOFFILE
0050 FILNAVN$:="DISK1:DATAFIL1";ENDOFFILE:=19
0060 FÆRDIG:=0
0070 OPEN FILNAVN$,R
0080 REPEAT
0090   GET FILNAVN$:DATAVAR$
0100   IF STATUS(FILNAVN$)=ENDOFFILE THEN
0110     FÆRDIG:=1
0120   ELSE
0130     IF STATUS(FILNAVN$) THEN
0140       PRINT "FEJL NUMMER ",STATUS(FILNAVN$)
0150       FÆRDIG:=1
0160     ELSE
0170       PRINT DATAVAR$
0180     ENDIF
0190   ENDIF
0200 UNTIL FÆRDIG
0210 CLOSE FILNAVN$
0220 END
```

## 6.6 PUT sætning

PUT sætningen bruges til at udskrive værdien af et eller flere udtryk i en post på en fil.

PUT sætningen har følgende opbygning:

```
      PUT  filidentifikator : udtryk , ... , udtryk
eller PUT  filidentifikator , postnummer : udtryk , ... , udtryk
```

De angivne udtryk skal enten være aritmetiske udtryk eller strengudtryk.

De to former for PUT sætningen svarer til sekventielle hhv. direkte filer. Ved en sekventiel fil udskrives data til den næste post i filen, ved en direkte fil udskrives data til posten med det opgivne postnummer.

I en sekventiel fil bestemmes postlængden af antallet og typen af elementer i udtrykslisten. Et reelt tal optager 8 tegn, et heltal 2 tegn og en streng optager lige så mange tegn som dens dynamiske længde angiver plus 1 stoptegn efter strengen. Hvis posten ikke kan rummes i filen, udvides denne (se afsnit 6.2).

Postlængden i en sekventiel fil må højst være 77 oktetter.

I en direkte fil er postlængden fastlagt ved oprettelsen af filen. Hvis udtrykslisten kræver flere oktetter, end der er i posten, afbrydes udførelsen med en fejlmeddelelse. Hvis udtrykslisten kræver færre oktetter, end der er i posten, er resten af postindholdet udefineret.

Forsøg på at udskrive en post til en fil, der er åbnet med tilgangsmåde R, resulterer umiddelbart i en fejlkode (status 13). Forsøg på at udskrive en post til en fil, som er åbnet med tilgangsmåde W, men som er beliggende på et skrivebeskyttet pladelager, vil resultere i en fejlkode (status 44). I det sidstnævnte tilfælde vil fejlkoden ikke nødvendigvis komme efter den første PUT operation på filen, men kan også komme som et resultat af en af de efterfølgende operationer på filen (evt. først i forbindelse med CLOSE).

Efter en PUT operation bør brugeren ved hjælp af STATUS funktionen kontrollere, om de ønskede data er udskrevet på filen.

## 6.7 STATUS funktion

STATUS funktionen anvendes til at undersøge status for en bestemt fil efter en filoperation.

En reference til STATUS funktionen, som er en systemfunktion, har følgende udseende:

STATUS ( filidentifikator )

Værdien af STATUS funktionen angiver resultatet af den nærmest foregående filoperation på den angivne fil. Følgende værdier er mulige:

0 - operationen afsluttet normalt
1 - en fil med det opgivne navn eksisterer ikke (OPEN)

2 - en fil med det opgivne navn eksisterer allerede (CREATE)
3 - pladen er fuld, der er ikke plads til posten (PUT, sekventiel) eller filen (CREATE)
4 - ulovlig postlængde (CREATE)
5 - filen benyttes af en anden bruger (OPEN)
6 - filen er ikke åben
8 - filen er fuld, der er ikke plads til posten, og filen er allerede udvidet 60 gange (PUT, sekventiel)
9 - fejl i syntaksen for filnavn
10 - fejl i syntaksen for etikette
11 - forsøg på at positionere filen til en ikke-eksisterende post
12 - (samme som 11)
13 - skrivning ikke tilladt, idet filen er åbnet med tilgangsmåde R
14 - kataloget er fuldt
18 - forsøg på at benytte direkte fil som om den var sekventiel eller omvendt; forsøg på at læse eller skrive ud over grænsen på en direkte fil
19 - slutmærke for sekventiel fil nået (GET, sekventiel)
20 - forsøg på at kopiere fra DDE-beskyttet diskette til ikke-DDE-beskyttet diskette
40 - det valgte pladelager er ikke klar (klappen er åben eller der er ikke indsat en diskette)
42 - plade ikke initialiseret eller pladelager defekt
44 - det valgte pladelager er skrivebeskyttet (CREATE, PUT, CLOSE)
46 - en plade med den opgivne etikette findes ikke (OPEN)

Eksempel: Se afsnit 6.5

## 6.8 ENDFILE sætning

ENDFILE sætningen bruges til at udskrive et slutmærke i en sekventiel fil.

ENDFILE sætningen har følgende opbygning:

ENDFILE filidentifikator

ENDFILE sætningen placerer et slutmærke efter den sidst udskrevne post i den angivne sekventielle fil. Ved en påfølgende indlæsning af filen kan brugeren ved hjælp af STATUS funktionen teste på dette slutmærke, og anvende det som tegn på at den meningsfyldte information i filen er slut.

Bemærk at DDE-COMAL-80 ikke selv generer et slutmærke i en sekventiel fil. Således er fx. indholdet af en nyoprettet sekventiel fil udefineret.

Efter en ENDFILE operation bør brugeren ved hjælp af STATUS funktionen kontrollere, at der er udskrevet et slutmærke på filen.

## 6.9 CLOSE sætning

CLOSE sætningen anvendes til at lukke en fil, som tidligere er blevet åbnet af programmet. Når en fil lukkes, ophæves enhver forbindelse mellem programmet og filen.

CLOSE sætningen har følgende opbygning:

```
                CLOSE  filidentifikator
eller           CLOSE
```

Den første sætningsform bevirker, at den angivne fil lukkes. Den anden sætningsform bevirker, at alle filer åbnet af dette program lukkes.

Hvis en eller flere filer er åbne når udførelsen af et program stoppes, lukkes filerne af DDE-COMAL-80.

Efter en CLOSE operation på en fil bør brugeren ved hjælp af STATUS funktionen kontrollere at filen er blevet lukket.

# 7. Standardfunktioner

## 7.1 Aritmetiske standardfunktioner

Aritmetiske standardfunktioner kan indgå i aritmetiske udtryk.

De eksisterende aritmetiske standardfunktioner er:

| | |
|---|---|
| ABS(X) | den absolutte værdi af X. |
| INT(X) | heltalsdelen af X, dvs. det største heltal, der er mindre end eller lig X. INT(1.5) = 1, INT(-1.5) = -2 . |
| SGN(X) | 0 hvis X=0, 1 hvis X>0, -1 hvis X<0 . |
| SQR(X) | kvadratroden af X, X >= 0 . |
| SIN(X) | sinus til X, X i radianer. |
| COS(X) | cosinus til X, X i radianer. |
| TAN(X) | tangens til X, X i radianer. |
| ATN(X) | arcus tangens til X udtrykt i radianer. |
| EXP(X) | eksponentialfunktionen taget af X, -32 < X < 32 . |
| LN(X) | den naturlige logaritme af X, X > 0 . |
| LOG(X) | titalslogaritmen af X, X > 0 . |

## 7.2 Tegnorienterede standardfunktioner

Tegnorienterede standardfunktioner anvendes til behandling af tegnstrenge. Funktionsværdien kan være et tegn eller et tal.

De eksisterende tegnorienterede standardfunktioner er:

CHR$ ( heltalsudtryk ) returnerer det tegn, der svarer til den opgivne heltalsværdi modulo 128. Sammenhængen mellem talværdier og ASCII tegn kan ses af appendix A. NB: tegnene hvis binære værdi ligger mellem 0 og 31 ( CHR(0) - CHR(31) ) er kontroltegn og bør benyttes med forsigtighed. Specielt bør man bemærke, at tegnet CHR(31) angiver afslutning på en streng - et forsøg på at konstruere en streng, som indeholder et CHR(31), vil derfor give mærkelige resultater.

CHR$ ( aritmetisk udtryk , heltalsudtryk , heltaludtryk )
returnerer værdien af det aritmetiske udtryk som en streng. Det første heltalsudtryk angiver hvor

mange cifre der skal være til venstre for decimalpunktet (evt indsættes indledende blanktegn) det andet hvor mange cifre der skal være efter decimalpunktet. Der foretages ingen afrunding og det kontrolleres ikke om tallet kan repræsenteres i det angivne format. Hvis formatet er for lille afskæres cifre til både højre og venstre for decimalpunktet, og der kan således forekomme tab af betydende cifre. Strengen vil således fylde (heltalsudtryk + 1 + heltalsudtryk) tegn. Det andet heltalsudtryk kan udelades og strengen fylder da (heltalsudtryk) tegn.

ASC( strengudtryk ) returnerer talværdien af strengen. Strengen skal indeholde et legalt tal på tegnform. Foranstillede blanktegn ignoreres. Konverteringen slutter med det første tegn som ikke kan indgå i et legalt tal. Hvis tegnstrengen ikke indledes med et tal afbrydes kørslen med en fejlmeddelelse.

LEN( strengudtryk ) returnerer et heltal, som er den aktuelle længde af det opgivne strengudtryk (se afsnit 4.7).

ORD( strengudtryk ) returnerer et heltal, som er talværdien for det første tegn i det opgivne strengudtryk. Sammenhængen mellem talværdier og ASCII tegn kan ses af appendix A.

## 7.3 Andre standardfunktioner

Ud over de i afsnit 7.1 og 7.2 beskrevne standardfunktioner, indeholder DDE-COMAL-80 yderligere et antal standardfunktioner, som er beskrevet i andre afsnit i denne håndbog.

Funktionerne er:

TAB(X) som er beskrevet i afsnit 5.23 (PRINT sætning).
STATUS(X) som er beskrevet i afsnit 6.7.

## 8.Kommandosystemet

Dette kapitel beskriver syntaks for og funktion af DDE-COMAL-80 kommandoerne.

En kommando er en instruktion til DDE-COMAL-80 systemet, som udføres øjeblikkeligt.

En kommando består af et nøgleord, der evt. efterfølges af en eller flere parametre. Nøgleordet kan forkortes. Den forkortede form af hvert nøgleord er anført i parentes efter nøgleordet i kommandobeskrivelsen. De angivne forkortelser kan ikke anvendes i DDE-COMAL-80 programsætninger, heller ikke hvis et nøgleord både kan anvendes som kommando og som sætningsnøgleord.

En kommando adskiller sig fra en programsætning ved ikke at begynde med et sætningsnummer.

Programmer kan lagres på to former: som interne programmer på binær form (LOAD, SAVE, SAVEOLD kommandoer), eller på tekstform (GET, PUT, PUTOLD, MERGE kommandoer). Programmer på binær form er hurtige at udskrive eller indlæse, men kan til gengæld være umulige at overføre fra en version af DDE-COMAL-80 systemet til en anden. Programmer på tekstform er langsomme at indlæse eller udskrive, bl. a. fordi de bliver syntaksanalyseret, men kan til gengæld umiddelbart overføres fra en DDE-COMAL-80 version til en anden.

Når en programlinie slettes, bliver den plads, som programlinien optager i datamatens lager, ikke umiddelbart tilgængelig. Da specielt redigering af en linie (herunder indtastning af en linie med samme nummer som en allerede eksisterende) medfører, at den originale linie slettes, og den rettede linie tilføjes efter programmet, kan den disponible plads reduceres betragteligt, hvis man foretager mange rettelser.

Den plads, som slettede programlinier optager, kan gøres tilgængelig ved at lagre programmet på tekstform (PUT/PUTOLD) og hente det igen (GET). Den frigjorte plads kan ikke genvindes ved hjælp af SAVE/LOAD.

Bemærk at der ikke må være blanktegn i en filidentifikation i filsystemkommandoer. Man kan til gengæld undlade " omkring en filidentifikation angivet som strengkonstant.

## 8.1 AUTO - automatisk udskrivning af linienumre

AUTO (AU) kommandoen benyttes, når systemet automatisk skal generere voksende linienumre ved programindtastning.

AUTO kommandoen har følgende opbygning:

```
          AUTO
eller     AUTO  startlinienummer
eller     AUTO  startlinienummer , interval
```

Når denne kommando er indtastet, vil DDE-COMAL-80 automatisk udskrive passende linienumre i starten af hver linie, indtil AUTO tilstanden forlades.

Det første linienummer, der genereres, er startlinienummer, og de følgende genereres med en tilvækst bestemt af interval. Hvis startlinienummer ikke angives, startes med 100, og hvis interval ikke angives benyttes tilvæksten 10.

Autolinieringstilstanden forlades ved at trykke på ESC, eller ved at overskrive linienummeret med en kommando.

Det er tilladt at ændre det af systemet automatisk udskrevne linienummer. Dette har ingen indflydelse på de følgende linienumre. Beregning af næste linienummer sker modulo 10000.

## 8.2 CATALOG - udskrivning af pladekatalog

CATALOG (CAT) kommandoen benyttes til at udskrive indholdet af et pladekatalog på dataskærm eller linieskriver.

CATALOG kommandoen har følgende opbygning:

```
          CATALOG  etikette
eller     CATALOG  £ pladenummer
```

hvor pladenummer er et heltal.

Kommandoen udskriver indholdet af kataloget for den angivne plade. Kommandoformen med pladenummer kan anvendes, hvis man har glemt etiketten for en plade. Pladenummer er nummeret på den pladelagerenhed, hvor pladen er monteret.

Kataloget udskrives normalt på dataskærmen, men kan også udskrives på linieskriveren (se afsnit 8.14).

Formatet for udskriften er:

```
KATALOG - PLADE "VICTOR  "

FILNAVN         TYPE                SEKTORER IALT

VICTOR1         DIREKTE FIL.12          0080
TESTPROG        PROGRAM                 0025
DATAFIL         SEKV. FIL               0020
TESTPROG        BINÆR                   0008
ASMPROG         0                       0004

SEKTORER IALT I BRUG   00154 -  TILBAGE    00206
```

De mulige filtyper er

- PROGRAM, program på kildetekstform (PUT kommando)
- BINÆR, program på binær form (SAVE kommando)
- SEKV. FIL, sekventielle DDE-COMAL-80 programdata
- DIREKTE FIL, direkte DDE-COMAL-80 programdata hvor tallet efter '.' er postlængden i oktetter
- 0 (eller andet), underprogram som ikke er skrevet i DDE-COMAL-80

Den sidste linie i katalogudskriften angiver det totale antal benyttede sektorer på pladen (inklusive etikette, katalog, aktive og slettede filer) og antal disponible sektorer. En sektor indeholder 256 oktetter.

## 8.3 CONTINUE - fortsæt afbrudt programeksekvering

CONTINUE (CON) kommandoen benyttes til at genstarte et program, som enten er blevet afbrudt af en fejl, eller fordi ESC tasten er blevet nedtrykket, eller som følge af at en STOP eller END sætning er blevet udført.

CONTINUE kommandoen har følgende opbygning:

```
            CONTINUE
eller       CONTINUE  linienummer
```

Kommandoen bevirker, at programudførelsen starter i det opgivne linienummer. Hvis der ikke er opgivet noget linienummer, fortsættes med sætningen

umiddelbart efter den sidst udførte. Når udførelsen genoptages, er den del af dataområdet, som indeholder variabelværdier samt oplysninger om procedurekald og løkkesætninger, uændret, medmindre man via LET kommandoen har ændret variables indhold.

En CONTINUE kommando accepteres kun af DDE-COMAL-80, såfremt der hverken er indtastet nye programlinier eller andre kommandoer end HELP, LET, LIST, OUTPUT, PRINT eller SIZE siden programmet stoppede.

Ved hjælp af OUTPUT kommandoen kan brugeren bestemme, om evt. uddata fra den fortsatte programudførsel skal udskrives på dataskærmen (ingen OUTPUT kommando) eller på linieskriveren (OUTPUT kommando indtastes inden CONTINUE).

## 8.4 CREATE - opret fil

CREATE (CR) kommandoen bruges til at oprette en datafil.

CREATE kommandoen har følgende opbygning:

```
      CREATE  filidentifikation
eller CREATE  filidentifikation , antal-sektorer
eller CREATE  filidentifikation , antal-poster , postlængde
```

CREATE kommandoen svarer syntaktisk og funktionsmæssigt til CREATE sætningen, som er beskrevet i afsnit 6.3.

## 8.5 DELETE - slet en eller flere programsætninger

DELETE (D) kommandoen benyttes til at slette en eller flere programsætninger.

DELETE kommandoen har følgende opbygning:

```
           DELETE  linienummer
eller      DELETE  startlinienummer , slutlinienummer
```

Den første form af kommandoen anvendes, når en enkelt programsætning ønskes fjernet fra programmet. Den anden form af kommandoen anvendes når alle programsætninger fra og med startlinienummer og til og med slutlinienummer ønskes fjernet fra programmet. Ved at angive slutlinienummer som * slettes alle linier fra og med startlinienummer til og med programmets sidste linie.

## 8.6 EDIT - rediger en eller flere programsætninger

EDIT (E) kommandoen benyttes til at redigere en eller flere programsætninger.

EDIT kommandoen har følgende opbygning:

```
          EDIT  linienummer
eller     EDIT  startlinienummer , slutlinienummer
```

Den første form af kommandoen anvendes, når en enkelt programsætning ønskes redigeret. Den anden form af kommandoen anvendes, når alle programsætninger fra og med startlinienummer og til og med slutlinienummer ønskes redigeret. Ved at angive slutlinienummer som * redigeres fra og med startlinienummer til og med programmets sidste linie.

Redigeringen sker ved, at programsætningen inklusive linienummer udskrives nederst på skærmen, og markøren placeres efter linienummeret. Brugeren kan nu ved hjælp af DDE-COMAL-80s redigeringsfaciliteter (se afsnit 3.3) rette den pågældende sætning. Når sætningen er rettet, tastes RETURN, hvorefter programsætningen syntaksanalyseres og lagres på normal vis. Derefter udskrives evt. den næste programsætning osv.

EDIT kommandoen afsluttes, når den sidste sætning i det opgivne interval er blevet rettet, når sætningen overskrives med en kommando, eller når der trykkes på ESC.

Det er tilladt at ændre linienummeret. I dette tilfælde placeres den redigerede linie svarende til det ændrede linienummer, og den oprindelige linie efterlades uændret.

## 8.7 GET - hent en programfil fra en plade

GET (GET) kommandoen benyttes til indlæsning af et program på tekstform fra en plade.

GET kommandoen har følgende opbygning:

```
GET  filidentifikation
```

Programmet skal være lagret ved hjælp af en PUT/PUTOLD kommando.

Inden programmet indlæses, udføres en implicit NEW kommando, som bevirker, at tidligere programmer og dataområder slettes.

Hvis DDE-COMAL-80 under indlæsningen af et program møder en programlinie, som ikke er syntaktisk korrekt (fx pga. en pladelagerfejl eller fordi programfilen ikke er genereret af DDE-COMAL-80), vil den indlæste linie blive omdannet til en kommentar, ved at '//' indføjes mellem linienummeret og den fejlbehæftede programsætning.

## 8.8 HELP - udskriv en oversigt over systemkommandoer

HELP (H) kommandoen udskriver navnene på de kommandoer, som er lovlige under DDE-COMAL-80, på skærmen.

HELP kommandoen har følgende opbygning:

```
HELP
```

## 8.9 LET - tildel variabel nyt indhold

LET (LET) kommandoen benyttes til at ændre indholdet af en eller flere variable direkte fra dataskærmen, fx inden programudførelsen genoptages ved hjælp af en CONTINUE kommando.

LET kommandoen har følgende opbygning:

```
LET tildeling ; ... ; tildeling
```

LET kommandoen svarer syntaktisk og funktionsmæssigt til tildelingssætningen som er beskrevet i afsnit 5.31.

## 8.10 LIST - udskriv programsætninger

LIST (LI) kommandoen benyttes til udskrivning af en eller flere linier i det indtastede program på dataskærm eller linieskriver.

LIST kommandoen har følgende opbygning:

```
          LIST
eller     LIST  linienummer
eller     LIST  startlinienummer , slutlinienummer
```

Den første form af kommandoen anvendes, når hele programmet ønskes udskrevet. Den anden form anvendes, når en enkelt linie ønskes udskrevet. Den tredie form anvendes, når alle programsætninger fra og med start li-

nienummer og til og med slutlinienummer ønskes udskrevet. I den tredie form kan slutlinienummeret angives som *. I dette tilfælde udskrives fra og med startlinienummer til og med programmets sidste linie.

Ved udskriften vil alle variable og nøgleord blive udskrevet med store bogstaver. Kommentarer og tekstkonstanter udskrives som de er blevet indtastet. Sætningslisten i sammensatte sætninger vil blive indrykket 2 positioner, dog max 20 positioner. Hvis indrykningen medfører, at en linie bliver længere end 79 tegn, vises kun de første 79 tegn af linien.

Udskrivningen sker normalt på dataskærmen, men kan også dirigeres til linieskriveren (se afsnit 8.14).

Udskrivningen kan afbrydes nårsomhelst ved tryk på ESC tasten.

## 8.11 LOAD - indlæs et binært program

LOAD (LOAD) kommandoen indlæser et binært program til brugerens programområde.

LOAD kommandoen har følgende opbygning:

```
LOAD filidentifikation
```

Det binære program skal være gemt ved hjælp af en SAVE/SAVEOLD kommando.

Indlæsning af binære programmer er hurtigere end indlæsning af programmer på tekstform (GET kommandoen), bl. a. fordi syntaksanalysen bortfalder.

Inden programmet indlæses, udføres en implicit NEW kommando, som bevirker, at tidligere programmer og dataområder slettes.

Efter indlæsningen er der ingen forskel på om et program er indlæst ved hjælp af GET eller LOAD.

## 8.12 MERGE - indflet en programfil fra en plade

MERGE (ME) kommandoen benyttes til at indflette en programtekst i brugerens programområde.

MERGE kommandoen har følgende opbygning:

```
MERGE filidentifikation
```

Programteksten skal være gemt ved hjælp af en PUT/PUTOLD kommando.

De indlæste programlinier placeres efter deres linienummer. Hvis der indlæses en programlinie med et linienummer, der findes i forvejen, slettes den eksisterende programlinie, og den nye programlinie indtager dens plads.

Hvis DDE-COMAL-80 under indlæsningen møder en programlinie, som ikke er syntaktisk korrekt (fx pga. en pladelagerfejl eller fordi den indflettede programfil ikke er genereret af DDE-COMAL-80), vil den indlæste linie blive omdannet til en kommentar, ved at '//' indføjes mellem linienummeret og den fejlbehæftede programsætning.

## 8.13 NEW - reinitialiser terminal og arbejdsareal

NEW (NEW) kommandoen benyttes til at fjerne et eksisterende program og de tilhørende data fra DDE-COMAL-80s programlagerområde, således at man kan indtaste et nyt program.

NEW kommandoen har følgende opbygning:

```
NEW
```

NEW kommandoen bør altid benyttes inden indtastning af et nyt program, for at sikre at linier fra tidligere programmer ikke bliver inkluderet i det nye program.

## 8.14 OUTPUT - diriger udskrift til linieskriver/dataskærm

OUTPUT (OUT) kommandoen anvendes til at bestemme om, udskriften fra et program (PRINT eller PRINT USING sætninger) eller en CATALOG eller LIST kommando skal sendes til dataskærmen eller linieskriveren.

OUTPUT kommandoen har følgende opbygning:

```
          OUTPUT Px
eller     OUTPUT T
```

OUTPUT kommandoen svarer funktionsmæssigt til OUTPUT sætningen, som er beskrevet i afsnit 5.21.

## 8.15 PRINT - vis indholdet af en variabel

PRINT (PR) kommandoen bruges til at udskrive indholdet af en eller flere variable eller udtryk på dataskærmen, fx efter at et program er blevet afbrudt.

PRINT kommandoen har følgende opbygning:

PRINT printelement skilletegn ... skilletegn printelement

PRINT kommandoen svarer syntaktisk og funktionsmæssigt til PRINT sætningen, som er beskrevet i afsnit 5.22.

PRINT kommandoen kan også anvendes til direkte beregning af aritmetiske udtryk.

```
Eksempel:   *PRINT SIN(1.05)^2 + COS(1.05)^2, SQR(49)
            1.000000000000      6.999999999999

            *
```

## 8.16 PUT - gem et program i en ny fil

PUT (PUT) kommandoen opretter en ny fil på en plade. Herefter gemmes indholdet af brugerens programområde på tekstform i den nyoprettede fil. Indholdet af programområdet ændres ikke.

PUT kommandoen har følgende opbygning:

PUT filidentifikation

## 8.17 PUTOLD - gem et program i en eksisterende fil

PUTOLD (PUTO) kommandoen gemmer indholdet af brugerens programområde på tekstform i en eksisterende fil. Indholdet af programområdet ændres ikke.

PUTOLD kommandoen har følgende opbygning:

PUTOLD filidentifikation

## 8.18 RENUMBER - omnummerering af programsætninger

RENUMBER (REN) kommandoen benyttes til at omnummerere sætningerne i et program.

RENUMBER kommandoen har følgende opbygning:

```
          RENUMBER
eller     RENUMBER  startlinienummer
eller     RENUMBER  startlinienummer , interval
```

Denne kommando bevirker, at programmet omnummereres således, at den første linie i programmet får det opgivne startnummer, og de følgende får en tilvækst bestemt ved interval. Hvis et eller flere af de beregnede linienumre overstiger 10000, genereres en fejlmeddelelse, og RENUMBER kommandoen udføres ikke.

Hvis startlinienummeret ikke er opgivet sættes startværdien til 100. Hvis interval ikke er opgivet, sættes tilvæksten til 10.

## 8.19 RUN - udfør et program

RUN (RUN) kommandoen benyttes til at starte udførelsen af et program.

RUN kommandoen har følgende opbygning:

```
          RUN
eller     RUN  linienummer
eller     RUN  filidentifikation
```

Den første form af kommandoen anvendes, når programmet ønskes udført startende med den første sætning. Den anden form af kommandoen anvendes, når programmet ønskes udført startende med en bestemt sætning. Den sidste form er en kort skrivemåde for kommandosekvensen

```
LOAD filidentifikation
RUN
```

Ved denne kommandosekvens indlæses et binært program (gemt ved hjælp af SAVE/SAVEOLD/SAVEPROTECT) til brugerens programområde, og eksekveringen starter herefter umiddelbart i programmets første linie.

Før programudførelsen slettes dataområdet, og stakken med information om underprogramkald og løkkesætninger initialiseres.

Inden RUN kommandoen kan brugeren ved hjælp af en OUTPUT kommando angive, om programuddata ønskes vist på dataskærmen eller udskrevet på linieskriveren.

Et kørende program kan nårsomhelst afbrydes ved tryk på ESC tasten (se dog afsnit 5.20).

## 8.20 SAVE - gem et binært program i en ny fil

SAVE (SAVE) kommandoen opretter en ny fil på en plade. Herefter gemmes indholdet af brugerens programområde på binær form i den nyoprettede fil. Indholdet af programområdet ændres ikke.

SAVE kommandoen har følgende opbygning:

```
SAVE filidentifikation
```

## 8.21 SAVEOLD - gem et binært program i en eksisterende fil

SAVEOLD (SAVEO) kommandoen gemmer indholdet af brugerens programområde på binær form i en eksisterende fil. Indholdet af programområdet ændres ikke.

SAVEOLD kommandoen har følgende opbygning:

```
SAVEOLD filidentifikation
```

## 8.22 SAVEPROTECT - gem et binært program i en fil på beskyttet form

SAVEPROTECT (SAVEP) kommandoen gemmer indholdet af brugerens programområde på binær form i en ikkeeksisterende fil. Indholdet af programområdet ændres ikke.

SAVEPROTECT kommandoen har følgende opbygning:

```
SAVEPROTECT filidentifikation
```

Ved LOAD´ning af en fil der er lagret med SAVEPROTECT bringes fortolkeren i en bekyttelsesmodus, hvor man alene kan udføre kommandoerne CATALOG, LOAD, RUN og NEW. Brugeren har således ingen mulighed for at se en udskrift af programteksten. Beskyttelsesmekanismen bevares ved kopiering af den binære programfil.

## 8.23 SIZE - udskriv størrelse af program- og dataområde

SIZE (SI) kommandoen udskriver oplysninger på dataskærmen om den benyttede og disponible længde af program- og dataområde.

SIZE kommandoen har følgende opbygning:

```
SIZE
```

Den udskrevne information er:

```
TOTAL xxxxx; I BRUG: PROGRAM yyyyy, DATA zzzzz
```

Her angiver xxxxx størrelsen af det kombinerede program- og dataområde i oktetter (bytes), yyyyy størrelsen af den del af området, der i øjeblikket anvendes til opbevaring af DDE-COMAL-80 programsætninger, og zzzzz størrelsen af den del af området, der i øjeblikket anvendes til programdata. Den samlede plads, som er til rådighed for yderligere programsætninger eller data, er således xxxxx-(yyyyy+zzzzz).

Når en linie slettes, mistes den plads som linien optog. Når en linie redigeres, udvides programområdet, idet den plads, som den oprindelige linie optog, mistes. Pladsen kan i begge tilfælde genvindes ved at udskrive programmet på kildetekstform (ikke binær form) på baggrundslageret (PUT/ PUTOLD) og herefter indlæse det igen.

Størrelsen af det til rådighed værende område bør aldrig falde under ca 350 oktetter, idet dette område benyttes af DDE-COMAL-80 til lagring af mellemresultater under programudførslen.

## 8.24 STOP - stop DDE-COMAL-80 systemet

STOP (STOP) kommandoen medfører, at DDE-COMAL-80 programmet standser ved at udføre en MIKADOS EXIT ordre. Herefter vil terminalen opføre sig som en normal MIKADOS terminal.

STOP kommandoen har følgende opbygning:

```
STOP
```

Denne kommando ignoreres i DDE-COMAL-80 systemer, som ikke eksekveres under et MIKADOS programudviklingssystem.

## 9. Særlige anvendelsesprogammer

Dette kapitel omhandler et antal anvendelsesprogrammer, som er udviklet af Dansk Data Elektronik ApS til hjælp ved vedligeholdelse af DDE-COMAL-80 filsystemet.

Filsystem anvendelsesprogrammerne er DDE-COMAL-80 programmer, der anvender et eller flere underprogrammer. Underprogrammerne er skrevet i assembler, og kaldes ved hjælp af en CALL sætning. Brugeren af DDE-COMAL-80 advares kraftigt mod at ændre i disse programmer. Brugeren advares ligeledes kraftigt imod at kalde de assemblerprogrammer, som kaldes af anvendelsesprogrammerne, fra sine egne programmer.

Anvendelsesprogrammerne ligger på pladen mærket "COMAL SYSTEM PROGRAMS", som leveres med hvert alenestående (dedikeret) DDE-COMAL-80 programsystem. Brugere, der anvender COMAL under MIKADOS, henvises til at anvende MIKADOS anvendelsesprogrammer beskrevet i "MIKADOS Utility Programs and Subroutines" manualen.

Et anvendelsesprogram udføres ved at montere pladen mærket "COMAL SYSTEM PROGRAMS" i en af pladelagerenhederne. Herefter kan programmet indlæses ved hjælp af en

```
GET SYSPROG:xxxxxx
```

kommando (xxxxxx erstattes med navnet på det ønskede program). Det indlæste program startes ved hjælp af en RUN kommando. Alle programmerne er konverserende (interaktive).

Under udførelsen af disse anvendelsesprogrammer, vil ESC tasten kunne være inaktiv.

I beskrivelserne af konversationen mellem et anvendelsesprogram og en bruger, er oplysninger indtastet af brugeren fremhævet med understregning. Den af anvendelsesprogrammerne udskrevne tekst kan afvige fra det i eksemplerne anførte.

I flere af anvendelsesprogrammerne kræves en drevbetegnelse til at udpege et bestemt pladelager. En drevbetegnelse er altid bogstavet 'P' efterfulgt af et ciffer. Drevbetegnelserne fremgår af den konfigurationsbeskrivelse, der leveres med systemet fra Dansk Data Elektronik ApS.

## 9.1 PCOPY80 - kopier en plade

PCOPY programmet kopierer alle filer på en plade til en anden plade af samme art. Programmet kan kun anvendes på systemer med mere end een pladelagerenhed. PCOPY kopierer også kataloget, og man sletter således alt tidligere indhold på den plade, man kopierer til.

Programmet kommunikerer med brugeren på følgende måde:

```
PCOPY - KOPIERING AF DISKETTER
PROGRAMMET STOPPES VED AT TRYKKE PAA ESC

INDTAST DREVBETEGNELSE (P1, P2, ...) FOR DEN DISKETTE

SOM DER KOPIERES FRA: P1

SOM DER KOPIERES TIL: P2
```

I det ovennævnte eksempel kopieres hele indholdet af pladen i drev P1 til pladen i drev P2.

Kopiering kan kun ske mellem to drev af samme type, dvs. af samme fysiske type og med samme spor- og sektorantal. Ved forsøg på at overtræde denne regel vil kopiering blive afslået med en fejlkode på mellem 46 og 52.

PCOPY programmet kan anvendes til sikkerhedskopiering af plader. Sikkerhedskopiering bør foretages mindst een gang dagligt af alle plader, som indeholder bevaringsværdige data.

## 9.2 INITLI80 - initialiser en plade

INITLIZE programmet sletter (initialiserer) en plade. Hvis det ønskes, kan programmet samtidig kontrollere om pladen er fysisk i orden. Programmet kan også anvendes til at ændre etiketten på en plade uden at ændre dennes indhold.

Programmet kommunikerer med brugeren på følgende måde:

```
INITLIZE - INITIALISERING AF DISKETTER
         - ELLER ÆNDRING AF ETIKETTER

INDTAST ETIKETTE: MINPLADE
INDTAST DREVBETEGNELSE (P1, P2, .. ): P2

INDSÆT DISKETTE I DREV NR. 2
```

```
TAST DEREFTER   I FOR INITIALISERING
                E FOR ÆNDRING AF ETIKETTE
              ESC FOR AT STOPPE PROGRAMMET : I
                J FOR INDLÆSNINGSKONTROL   : J
```

I det ovennævnte eksempel gennemføres en initialisering med kontrol af pladen monteret i drev P2. Samtidig får pladen tildelt etiketten MINPLADE (det gamle etiketteindhold er uden betydning).

Spørgsmålet om indlæsningskontrol stilles kun i forbindelse med en initialisering. Hvis der svares 'J' til dette spørgmål, vil programmet læse al den udskrevne initialiseringsinformation tilbage i lageret, og kontrollere at den udskrevne information rent faktisk er blevet registreret på disketten. Denne mulighed tager ca. dobbelt så lang tid som en initialisering uden tilbagelæsning, men hjælper til gengæld med at finde fejl (defekt diskettebelægning etc.).

INITLIZE programmet kan ikke anvendes til at formattere en fabriksny diskette. Fabriksny disketter kan kun formatteres af Dansk Data Elektronik ApS.

## 9.3 PURGE80 - slet en fil

PURGE programmet sletter en fil på en plade. Den plads, som filen optog, kan genvindes ved at komprimere pladen ved hjælp af COMPR programmet.

Programmet kommunikerer med brugeren på følgende måde:

```
PURGE - SLET EN FIL
PROGRAMMET STOPPES VED AT TRYKKE PÅ ESC

INDTAST 'ETIKETTE:FILNAVN' FOR DEN FIL, DER ØNSKES SLETTET: TEST:SLET
INDTAST FILTYPE (K FOR PROGRAM-/DATAFILER, B FOR BINÆRFILER): K
```

I det ovennævnte eksempel vil programfilen 'SLET' på disketten med etiketten 'TEST' blive slettet.

## 9.4 RENAME80 - omdøb en fil

RENAME programmet ændrer navnet på en fil. Filens indhold ændres ikke. Hvis etiketten på en plade ønskes ændret, anvendes programmet INITLIZE.

Programmet kommunikerer med brugeren på følgende måde:

RENAME - ÆNDRING AF FILNAVN
PROGRAMMET STOPPES VED AT TRYKKE PÅ ESC

INDTAST DET NUVÆRENDE FILNAVN (MED ETIKETTE): PLADE:GAMLNAVN
INDTAST FILTYPE (K FOR PROGRAM-/DATAFILER, B FOR BINÆRFILER): K
INDTAST DET NYE FILNAVN (UDEN ETIKETTE): NYTNAVN

I det ovennævnte eksempel vil programfilen GAMLNAVN på pladen med etiketten PLADE blive omdøbt til NYTNAVN. Filens indhold berøres ikke.

## 9.5 FCOPY80 - kopier en fil

FCOPY programmet kopierer en fil fra en plade til en anden. Den modtagende fil behøver ikke at have samme navn som originalfilen.

Programmet kommunikerer med brugeren på følgende måde:

FCOPY - KOPIER FIL.

BETEGNELSE PÅ DET DREV, SOM DER KOPIERES FRA (P1, P2, .. ): P2
BETEGNELSE PÅ DET DREV, SOM DER KOPIERES TIL (P1, P2, .. ): P3
INDTAST FILNAVN: FLYTFIL
FILTYPE (K FOR PROGRAM-/DATAFIL, B FOR BINÆRFIL): B
EVT. NYT FILNAVN: _

I det ovennævnte eksempel vil den binære programfil FLYTFIL blive kopieret fra drev P2 til drev P3. Kopifilen vil have samme navn som originalfilen.

Hvis den modtagende fil eksisterer i forvejen, overskriver FCOPY dens indhold med indholdet af originalfilen. Hvis den modtagende fil ikke eksisterer i forvejen, oprettes den af FCOPY som en tro kopi af originalfilen.

Det er tilladt, at kopiere fra og til den samme plade, man skal blot huske at omdøbe sin fil.

## 9.6 COMPR80 - komprimer en plade

COMPR programmet anvendes til at genvinde den plads på en diskette, som optages af evt. slettede filer.

Programmet kommunikerer med brugeren på følgende måde:

COMPR - KOMPRIMERING AF EN PLADE
PROGRAMMET STOPPES VED AT TRYKKE PÅ ESC

INDTAST BETEGNELSE PÅ DET DREV HVOR DER SKAL KOMPRIMERES (P1, P2, .. ) P2

I det ovennævnte eksempel vil disketten, som er anbragt i drev P2, blive komprimeret.

Komprimeringen er relativt langsom (flere minutter). Under komprimeringen bør der ikke foregå andet i systemet, dvs. i flerbrugersystemer bør der ikke være nogen aktivitet ved de andre terminaler.

## Appendix A. Aritmetiske værdier af ASCII tegn

| tegn | værdi | tegn | værdi | tegn | værdi | tegn | værdi |
|---|---|---|---|---|---|---|---|
| NUL | 0 | blank | 32 | § | 64 | ` | 96 |
| SOH | 1 | ! | 33 | A | 65 | a | 97 |
| STX | 2 | " | 34 | B | 66 | b | 98 |
| ETX | 3 | £ | 35 | C | 67 | c | 99 |
| EOT | 4 | $ | 36 | D | 68 | d | 100 |
| ENQ | 5 | % | 37 | E | 69 | e | 101 |
| ACK | 6 | & | 38 | F | 70 | f | 102 |
| BELL | 7 | ´ | 39 | G | 71 | g | 103 |
| BSL | 8 | ( | 40 | H | 72 | h | 104 |
| HT | 9 | ) | 41 | I | 73 | i | 105 |
| LF | 10 | * | 42 | J | 74 | j | 106 |
| VT | 11 | + | 43 | K | 75 | k | 107 |
| FF | 12 | , | 44 | L | 76 | l | 108 |
| CR | 13 | - | 45 | M | 77 | m | 109 |
| SO | 14 | . | 46 | N | 78 | n | 110 |
| SI | 15 | / | 47 | O | 79 | o | 111 |
| DLE | 16 | 0 | 48 | P | 80 | p | 112 |
| DC1 | 17 | 1 | 49 | Q | 81 | q | 113 |
| DC2 | 18 | 2 | 50 | R | 82 | r | 114 |
| DC3 | 19 | 3 | 51 | S | 83 | s | 115 |
| DC4 | 20 | 4 | 52 | T | 84 | t | 116 |
| NAK | 21 | 5 | 53 | o | 85 | u | 117 |
| SYN | 22 | 6 | 54 | V | 86 | v | 118 |
| ETB | 23 | 7 | 55 | W | 87 | w | 119 |
| CAN | 24 | 8 | 56 | X | 88 | x | 120 |
| EM | 25 | 9 | 57 | Y | 89 | y | 121 |
| SUB | 26 | : | 58 | Z | 90 | z | 122 |
| ESC | 27 | ; | 59 | Æ | 91 | æ | 123 |
| FS | 28 | < | 60 | Ø | 92 | ø | 124 |
| GS | 29 | = | 61 | Å | 93 | å | 125 |
| RS | 30 | > | 62 | ^ | 94 | ¨ | 126 |
| US | 31 | ? | 63 | - | 95 | DEL | 127 |

## Appendix B. Fejlmeddelelser

Dette afsnit indeholder en oversigt over alle fejlmeddelelser, som DDE-COMAL-80 systemet kan frembringe, samt en kort forklaring af fejlmeddelelsens betydning. Ved fejludskrifter i forbindelse med indtastning af kommandoer eller programsætninger vil markøren blive placeret i nærheden af det sted, hvor fejlen er konstateret. Ved fejludskrifter i forbindelse med kørsel af programmer vil DDE-COMAL-80 udskrive nummeret på den programsætning, som blev udført, da fejlen blev opdaget. COMAL findes i versioner med enten danske eller engelske fejlmeddelelser. De engelske fejlmeddelelser er anført i parantes.

ARITMETISK OVERLØB (ARITHMETIC OVERFLOW)
: et resultat eller et mellemresultat i en aritmetisk beregning ligger uden for datamatens talområde (se afsnit 4.5)

CASE/ENDCASE (CASE/ENDCASE MISMATCH)
: CASE og ENDCASE er anbragt forkert i forhold til hinanden eller optræder ikke parvis

CONTINUE ULOVLIG (CONTINUE ILLEGAL)
: CONTINUE kommandoen er kun lovlig i visse tilfælde, se afsnit 8.3

DATA MANGLER (DATA EXHAUSTED)
: den angivne READ sætning forsøger at læse ud over de definerede DATA (se afsnittene 5.7, 5.25 og 5.28)

DATA OVERLØB (DATA OVERFLOW)
: arbejdsområdet er fuldt - der er ikke plads til at oprette flere variable, til at åbne flere filer eller til nødvendige mellemresultater (ca. 350 oktetter skal konstant være disponible til mellemresultater); måske er en eller flere variable dimensioneret større end nødvendigt - ellers kan programmet opdeles i flere dele, der kædes sammen med CHAIN sætningen. Se også vinkene i afsnit 8.22 (SIZE kommandoen)

DIVISION MED 0 (DIVISION BY 0)
: forsøg på at dividere med nul

ETIKETTE FINDES IKKE (LABEL NOT FOUND)
: der er ikke indsat en plade med den i filnavnet opgivne etikette (benyt evt. CATALOG kommandoen til at se hvad etiketten er) eller der forsøges at hoppe til en ikk-erklæret etikette

FILEN ER I BRUG (FILE IN USE)

filen benyttes af en anden bruger, eller filen er allerede åbnet af denne bruger

FILEN FINDES (FILE EXISTS)

forsøg på at oprette en fil med samme navn som en allerede eksisterende fil

FILEN FINDES IKKE (FILE NOT FOUND)

en fil med det opgivne navn findes ikke på pladen med den angivne etikette

IKKE-DIMENSIONERET VARIABEL (UNDIMENSIONED VARIABLE)

i den fejlbehæftede sætning optræder en variabel, der ikke er erklæret i en DIM sætning, som om den var en vektor eller matrix

INDEX FEJL (INDEX ERROR)

i den anførte sætning optræder et indeks, som har en værdi, der er negativ, nul eller større end det maksimale antal elementer erklæret ved DIMensioneringen af den pågældende variabel. Bemærk at ved strengvariable med selektor skal startindeks+længde være mindre end eller lig med den erklærede længde. Indeksværdien på fejltidspunktet kan inspiceres med PRINT kommandoen

INDLEJRING FOR DYB (NESTING TOO DEEP)

der er for mange sammensatte sætninger (FOR-NEXT, REPEAT-UNTIL, PROC-ENDPROC osv.) inden i hinanden

INTET PROGRAM (NO PROGRAM)

kommandoen kan ikke udføres, da der ikke er indtastet nogen programsætninger

KATALOGET ER FULDT (CATALOG FULL)

kataloget på pladen er fuldt; der kan ikke oprettes flere filer på pladen. Der kan skaffes plads i kataloget ved at slette en eller flere filer; komprimering hjælper ikke

MATERIALEFEJL (PLADEN INITIALISERET ?)

(HARD ERROR (DISK INITIALIZED?)) forsøg på at læse eller skrive på en plade som er helt eller delvist uinitialiseret. Hvis pladen er korrekt initialiseret, er der fejl på pladelagerenheden, og der bør tilkaldes kvalificeret teknisk bistand

MANGLER ) (MISSING ))

der mangler en eller flere højreparenteser ´)´ i sætningen

MANGLER ( (MISSING ()
: på det sted, hvor markøren står, skal der ifølge syntaksreglerne stå en venstreparentes '('

MANGLER " (MISSING ")
: der mangler et afsluttende anførelsestegn " i en strengkonstant

MANGLER FUNKTIONSNAVN (MISSING FUNCTION ID)
: funktionsnavnet mangler

MANGLER RELATION (MISSING RELATION)
: et strengudtryk optræder i en sammenhæng, hvor det kun er lovligt hvis det efterfølges af en relationsoperator (se afsnit 4.9)

PLADEN ER FYLDT (DISK FULL)
: der er ikke plads til den nye fil eller filudvidelse på pladen; indsæt en ny plade eller komprimer den fulde plade

PLADEN IKKE KLAR (DISK NOT READY)
: Der er ikke monteret en diskette i det refererede drev

PLADEN SKRIVEBESKYTTET (WRITE PROTECT)
: forsøg på at skrive på en skrivebeskyttet plade

PROGRAM OVERLØB (PROGRAM OVERFLOW)
: der er ikke plads til flere programsætninger i arbejdsområdet; se DATA OVERFLOW og vinkene i afsnittet om SIZE kommandoen (afsnit 8.23)

SKRIVER OPTAGET (PRINTER BUSY)
: (kun i systemer med flere samtidige brugere) forsøg på at udskrive på printeren mens denne benyttes af en anden bruger

TYPE KONFLIKT (DATA MISMATCH)
: typen på en variabel i en READ sætning passer ikke med typen på den tilsvarende dataværdi i DATA listen

TYPE KONFLIKT (TYPE CONFLIKT)
: forskellige typer af udtryk i samme WHEN liste, eller forskellige typer af udtryk i CASE sætning og tilhørende WHEN sætning

UDEFINERET PROCEDURE (UNDEFINED PROCEDURE)
: i den fejlbehæftede sætning optræder en ikke tidligere erklæret funktion eller procedure

UDEFINERET VARIABEL (UNDEFINED VARIABLE)
en variabel refereres uden at den er erklæret / har fået tildelt en værdi

UDTRYK FOR KOMPLEKS (EXPRESSION TOO COMPLEX)
det aritmetiske udtryk er for kompliceret (indeholder for mange parentesniveauer) til at det kan beregnes; beregningen af udtrykket bør ske over flere sætninger

UKENDT KOMMANDO (PRØV 'HELP') (UNKNOWN COMMAND (ENTER 'HELP'))
ukendt kommando; brugeren kan indtaste HELP for at få en oversigt

ULOVLIG ARGUMENT (ILLEGAL ARGUMENT)
ulovlig parameter til CREATE; en systemfunktion har et ulovligt argument; værdiområderne for argumenter til disse funktioner er anført i kap. 7

ULOVLIG ARGUMENT (ILLEGAL ARGUMENT LIST)
antallet af aktuelle parametre i EXEC sætningen og antallet af formelle parametre i den tilsvarende PROC sætning er ikke det samme

ULOVLIG ETIKETTE (ILLEGAL DISK ID)
pladelagerbetegnelsen i en CATALOG kommando er forkert, se afsnit 8.2

ULOVLIG EXIT (ILLEGAL EXIT)
en EXIT sætning udføres når programmet ikke er inde i en sammensat sætning

ULOVLIG FILNAVN (ILLEGAL FILE NAME)
det opgivne strengudtryk indeholder ikke et korrekt filnavn (etikette:fil), se afsnit 6.1.2

ULOVLIG FORMEL PARAMETER (ILLEGAL FORMAL PARAMETER)
parametrene i argumentlisten i en PROC sætning; en formel parameter må ikke være brugt som identifikator i andre sammenhænge.

ULOVLIG FUNKTIONSNAVN (ILLEGAL FUNCTION NAME)
ulovligt navn på funktion eller procedure - navnets betydning er tidligere defineret i programmet

ULOVLIG GET (ILLEGAL GET)
forsøger at læse en binær programfil med GET

ULOVLIG GOTO (ILLEGAL GOTO)
hop ud af en procedure eller ind i en sammensat sætning

ULOVLIG KONSTANT (ILLEGAL CONSTANT)
fejl i et reelt tal (syntaksfejl eller overløb, se afsnit 4.1)

ULOVLIG LOAD (ILLEGAL LOAD)
forsøger at læse en programtekstfil med LOAD

ULOVLIG NØGLEORD (ILLEGAL KEYWORD)
markøren udpeger et ord i sætningen som er ulovligt ifølge syntaksen (stavet galt ? - ellers gennemlæs syntaksreglerne endnu en gang)

ULOVLIG OPERAND (ILLEGAL OPERAND)
forkert variabeltype i denne sammenhæng, evt. syntaksfejl

ULOVLIG SÆTNINGSNUMMER (ILLEGAL SEQUENCE NUMBER)
der mangler et linienummer; linienummeret er syntaktisk ukorrekt; der er opgivet to linienumre i en kommando, men det andet er mindre end det første

ULOVLIG TEGN (ILLEGAL CHARACTER)
næste tegn i sætningen svarer ikke til syntaks

ULOVLIG TILDELING (ILLEGAL ASSIGNMENT)
forsøg på at tildele en værdi til en størrelse, som ikke kan tildeles en værdi, fx en strengkonstant eller en funktion

ULOVLIG VARIABEL (ILLEGAL VARIABLE)
variabel eller udtryk har ikke den forventede type

UVENTET RETURN ELLER NEXT (UNEXPECTED RETURN OR NEXT)
en eller flere sammensatte sætninger afsluttes ikke i omvendt rækkefølge af den hvori de påbegyndtes (eksempel: en NEXT sætning refererer ikke til samme variabel som den sidst udførte FOR sætning). Bemærk at denne fejlmeddelelse omfatter alle typer af sammensatte sætninger, dvs. også fx REPEAT-UNTIL og PROC-ENDPROC

WHEN UDEN CASE (WHEN WITHOUT CASE)
der optræder en WHEN uden et indledende CASE

# Appendix C. Programeksempler

## C.1 Dæmpet svingning.

Programmet tegner en dæmpet svingning på dataskærmen. Dæmpningsfaktoren mindskes for hvert billede. Ialt 5 billeder tegnes.

```
0010 PROC DÆMPETSV(X)
0012   DÆMPETSV:=11*EXP(-X/P)*COS(X/2.5)+12
0014 ENDPROC DÆMPETSV
0020 DIM AKSE$ OF 80
0025 INTEGER P,I,VENT
0030 FOR P:=5 TO 25 STEP 5 DO
0040   CLEAR
0050   // OPBYG OG UDSKRIV X-AKSEN.
0060   FOR I:=1 TO 71 STEP 10 DO
0070     AKSE$(I:10):="----------"
0080   NEXT I
0090   CURSOR 1,12
0100   PRINT AKSE$(1:79)
0110   // BEREGN OG UDSKRIV 79 PUNKTER
0120   FOR I:=1 TO 79 DO
0130     Y:=DÆMPETSV(I)
0140     // FLYT MARKØREN TIL BEREGNEDE PLADS
0150     CURSOR I,Y
0160     PRINT "<S>*"
0170   NEXT I
0180   // FLYT MARKØR TIL NEDERSTE VENSTRE HJØRNE
0190   CURSOR 1,24
0200   // VENT MENS KURVEN BEUNDRES
0210   FOR VENT:=1 TO 1000 DO
0220   NEXT VENT
0230 NEXT P
0240 STOP
```

Linieskiftet efter udskriften af hver * undgås ved at benytte kommandoen <S> til skærmen i linie 160.

## C.3 Fletning af to kildetekstfiler

Program til fletning af to kildetekstfiler (fx DDE-COMAL-80 programmer gemt ved hjælp af PUT kommandoen).

Programmet henter poster fra hver af de to inddatafiler og placerer posterne i uddatafilen efter deres linienummer. Hvis der optræder poster med samme linienummer i filerne, udskrives kun posten fra den første fil. Programmet kræver, at linierne alle er forsynet med et linienummer, og at linienumrene optræder i strengt voksende orden.

```
0100 // * * PROGRAM TIL FLETNING AF TO KILDETEKSTFILER * *
0130 DIM PRIMÆRFIL$ OF 20,SEKUNDÆRFIL$ OF 20,RESULTATFIL$ OF 20
0140 DIM POST1$ OF 80,POST2$ OF 80,NUMMER1$ OF 4,NUMMER2$ OF 4
0144 INTEGER TRUE,FALSE,EOF
0150 TRUE:=1;FALSE:=0;EOF:=19
0160 INPUT "NAVNET PÅ UDDATAFILEN ":RESULTATFIL$
0170 OPEN RESULTATFIL$,W
0180 EXEC FEJL(STATUS(RESULTATFIL$))
0190 INPUT "NAVNET PÅ DEN FØRSTE PROGRAMFIL ":PRIMÆRFIL$
0200 OPEN PRIMÆRFIL$,R
0210 EXEC FEJL(STATUS(PRIMÆRFIL$))
0220 INPUT "NAVNET PÅ DEN ANDEN PROGRAMFIL ":SEKUNDÆRFIL$
0230 OPEN SEKUNDÆRFIL$,R
0240 EXEC FEJL(STATUS(SEKUNDÆRFIL$))
0250 EXEC GET1
0260 EXEC GET2
0270 REPEAT
0280   CASE TRUE OF
0290   WHEN NUMMER1$>NUMMER2$
0300     PUT RESULTATFIL$:POST2$
0305     EXEC FEJL(STATUS(RESULTATFIL$))
0310     EXEC GET2
0320   WHEN NUMMER1$=NUMMER2$
0330     PUT RESULTATFIL$:POST $
0335     EXEC FEJL(STATUS(RESULTATFIL$))
0340     EXEC GET1
0350     EXEC GET2
0360   WHEN NUMMER1$<NUMMER2$
0370     PUT RESULTATFIL$:POST1$
0375     EXEC FEJL(STATUS(RESULTATFIL$))
0380     EXEC GET1
0390   ENDCASE
0400 UNTIL NUMMER1$>"9999" AND NUMMER2$>"9999"
```

```
0410 ENDFILE RESULTATFIL$
0415 EXEC FEJL(STATUS(RESULTATFIL$))
0420 CLOSE
0425 EXEC FEJL(STATUS(PRIMÆRFIL$))
0426 EXEC FEJL(STATUS(SEKUNDÆRFIL$))
0427 EXEC FEJL(STATUS(RESULTATFIL$))
0430 PRINT "FLETNING AFSLUTTET"
0440 END
0450 PROC GET1
0460   GET PRIMÆRFIL$:POST1$
0470   IF STATUS(PRIMÆRFIL$)=EOF THEN
0480     NUMMER1$:="SLUT"
0490   ELSE
0500     EXEC FEJL(STATUS(PRIMÆRFIL$))
0510     NUMMER1$:=POST1$(1:4)
0520   ENDIF
0530 ENDPROC GET1
0540 PROC GET2
0550   GET SEKUNDÆRFIL$:POST2$
0560   IF STATUS(SEKUNDÆRFIL$)=EOF THEN
0570     NUMMER2$:="SLUT"
0580   ELSE
0590     EXEC FEJL(STATUS(SEKUNDÆRFIL$))
0600     NUMMER2$:=POST2$(1:4)
0610   ENDIF
0620 ENDPROC GET2
0630 PROC FEJL(FILSTATUS)
0640   IF FILSTATUS<>0 THEN
0650     PRINT "FILSYSTEMFEJL NR.";FILSTATUS
0660     PRINT "MERGE AFSLUTTET"
0670     STOP
0680   ENDIF
0690 ENDPROC FEJL
```

18 okt 1983

Comal80 version 1.5

I DDE's comal80 version 1.5 er mekanismen omkring reservation og frigivelse af skrivere ændret, så det fra comal programmet kan undersøges om reservationen eller frigivelsen af en skriver er forløbet tilfredsstillende. I modsat fald er det muligt fra programmets side at tage en fornuftig aktion på grundlag af en fejlkode.

Ændringen bevirker, at reservationen af skriveren foretages ved udførelsen af en OUTPUT "P" sætning, mens frigivelsen udføres ved en OUTPUT "T" sætning. Resultatet af reservationen eller frigivelsen kan aflæses ved at kald af standardfunktionen STATUS med argument "P" eller "T", fx I := STATUS("P").

Statuskoderne, der benyttes er:

0 - reservationen eller frigivelsen er foretaget
1 - skriveren er reserveret af en anden bruger (reservation)
2 - skriveren er ikke reserveret (frigivelse)
3 - reservationen kan ikke foretages pga overløb i ressourcetabel (reservation)
4 - benyttes ikke
5 - brugeren har allerede reserveret een skriver (reservation)

Såfremt en reservation eller frigivelse ikke udføres som ønsket, fortsætter udskrivningnen blot på gældende udskrivningsmedium.

Samtidig med denne ændring er der indført en ny funktion, der suspenderer programudførelsen. Funktionen hedder DELAY og argumentet angiver antal sekunder programmet skal suspenderes. Ved kald af den interne venterutine benyttes argumentets værdi modulo 512, dvs at programmet højest kan suspenderes i 511 sekunder. Et negativt argument bevirker, at programmet suspenderes i 0 sekunder.

Bemærk at DELAY er en funktion og altså skal kaldes som fx I:=DELAY(10).

Anders Ansted
Dansk Data Elektronik A/S