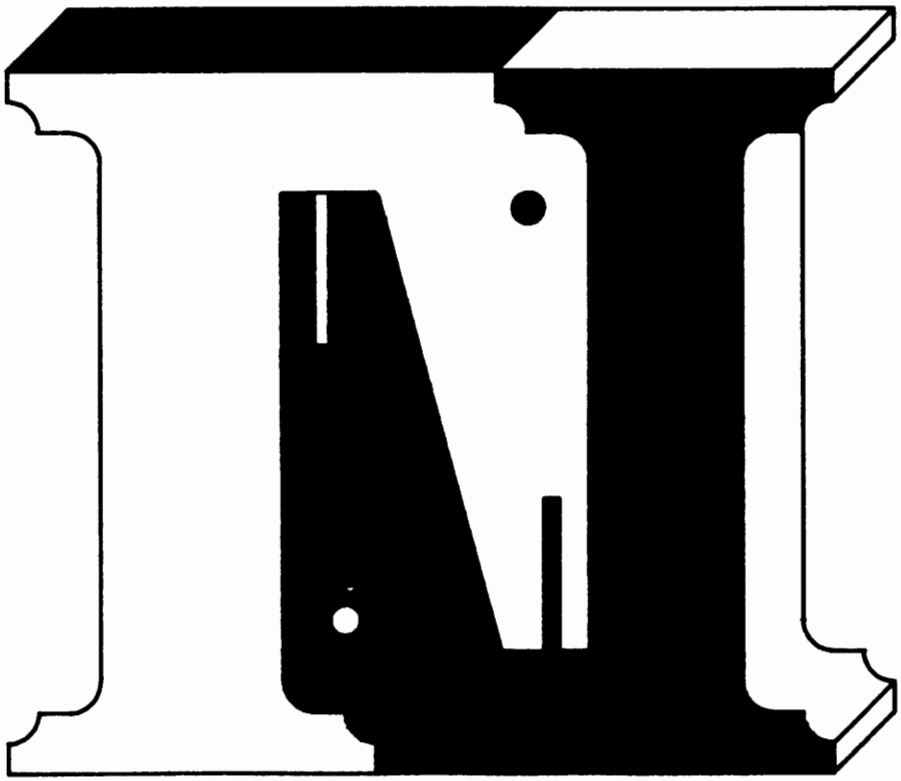
Ole Madsen

COMAL-80

Version 2 - BRUGERVEJLEDNING - SEP.84

JENSEN og JØRGENSEN og KNUDSEN og SØRENSEN Aps

FÅBORGVEJ 54 • 5762 VESTER SKERNINGE • TLF.: 09-24 37 24



ICL

1 Meddelelser

2 Indledning

INDHOLDSFORTEGNELSE KAPITEL 2 - INDLEDNING

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

2. INDLEDNING

2.1 GENEREL INDHOLDSFORTEGNELSE

5. COMAL-80 sætninger

2.2 SPROGET, COMAL-80

Grundlaget for sproget, COMAL, blev i 1974 defineret af Børge R. Christensen, Tønder Statsseminarium, og Benedict Løfstedt, Århus Universitet.

Idéen var at kombinere den nemme anvendelse af BASIC med de store muligheder for bl.a. struktureret programmering i PASCAL.

COMAL har vundet hurtig udbredelse i den danske undervisningssektor, og man har således haft mange erfaringer at bygge på.

COMAL-80 er den avancerede udgave af COMAL, som i 1980 blev lanceret på COMET'en.

2.3 GENERELT OM DENNE MANUAL

2.3.1 FORMÅL

Det er primært en referencemanual, d.v.s. et skrift, som bør give en korrekt, præcis og udtømmende beskrivelse af COMAL-80 med tilhørende kommandoer på COMET'en.

Det er kun sekundært en lærebog, idet det er meget vanskeligt at lave en stringent referencemanual, som samtidigt kan bruges som lærebog for skoleelever uden datalogisk baggrund.

Vi har dog søgt at give en særligt dybtgående forklaring på vanskelige områder såsom strengvariable, arrays, ind- og udlæsning samt TRAP-faciliteten.

Specialistområder såsom PEEK, POKE og VARPTR er imidlertid kun beskrevet til referenceformål.

Manualen kan bruges som støtte for undervisning eller i forbindelse med lærebøger.

Det må understreges, at andre lærebøger i COMAL ikke nødvendigvis er dækkende for COMAL-80: principielt er det COMAL-80 manualen, der skal give den endelige sandhed.

2.3.2 NUMMERERING

Manualen er opdelt i kapitler (eller afsnit) på normal vis. Hvert afsnit kan være underinddelt i afsnit.

Et afsnits niveau i hierakiet er angivet ved dets nummer. Således består f.eks. kapitel 5 af afsnittene:

På første niveau:

5.1, 5.2, 5.3 o.s.v.

På andet niveau:

5.1.1, 5.1.2, ..., ..., 5.2.1, 5.2.2, 5.2.3 o.s.v.

På tredie niveau:

5.1.1.1., 5.1.1.2, ..., 5.1.2.1, 5.1.2.2, ... o.s.v.

Siderne er fortløbende nummereret inden for hvert kapitel på formen: Kapitelnr..sidenr., f.eks. 5.17.

Opdatering vil foregå ved udskiftning eller fjernelse af sider og/eller tilføjelse af sider. Hvis der f.eks. skal tilføjes sider mellem side 5.18 og side 5.19, vil de få numrene 5.18.1, 5.18.2, 5.18.3 o.s.v.

2.3.3 BRUG AF MANUALEN

Kapitel 1 er beregnet til opdateringsmeddelelser o. lign. Det kan dog også anvendes til generelle oplysninger, f.eks. på en skole, hvor der er flere, der bruger maskinen.

I starten af hvert kapitel er anbragt en indholdsfortegnelse for kapitlet. I nærværende kapitel er der desuden en generel indholdsfortegnelse. Kapitel 10 er reserveret til brugerens egne notater samt til f.eks. ark i plasticlommer, der rager ud over A4-formatet.

2.3.4 SPROGBRUG

Der er så vidt muligt brugt danske betegnelser (det betyder termer). Undtagelser er f.eks. det engelsk prægede "fil" i stedet for "register", "array(s)" i stedet for sæt samt naturligvis alle nøgleordene, som i sig selv er engelske. Disse er i øvrigt stavet med store bogstaver i manualen (det behøver de ikke at være i programmer). Andre ord, der ønskes fremhævet, kan også være stavet med store bogstaver.

"Anførelsestegn" er andre steder i manualen stavet "anførselstegn", i overenstemmelse med almindelig praksis.

"Variabel" bruges bl.a. både som navneord og som tillægsord. De steder, hvor det udtrykkeligt skal opfattes som et navneord, har det formen: En variabel, variablen, flere variabler, alle variablerne. Normalt bruges det som tillægsord på formen: En variabel, den variable, flere variable, de variable.

Der er afvigelser fra den korrekte tegnsætning i de tilfælde, hvor denne kunne blive misfortolket. Afsluttende punktum er normalt udeladt i overskrifter.

2.3.5 KORREKTHED

ICL har bestræbt sig på at gøre denne manual så korrekt og fyldestgørende som muligt; ICL kan dog ikke påtage sig noget ansvar for konsekvenser af eventuelle fejl og mangler.

COMET'en og COMAL-80 er under stadig udvikling.

ICL forbeholder sig derfor retten til at foretage ændringer eller indføre faciliteter, der ikke er beskrevet heri, eller at beskrive faciliteter, der endnu ikke er implementeret, eller som kun kan anvendes af en del af brugerne.

Særlige omstændigheder kan gøre, at specifikationerne skal modificeres. Specifikationerne udgør ikke i sig selv nogen kontraktlig forpligtelse, men kan selvfølgelig eventuelt indgå i en kontrakt, hvis man bliver enige om noget sådant. I så fald skal det ske ved udtrykkelig henvisning til en bestemt udgave af manualen med angivelse af opdateringsniveau.

2.3.6 COPYRIGHT

2.3.7 UDGAVENR.

Denne manual erstatter COMAL-80 Brugervejledning, november 1980. Nærværende manual:

COMAL-80 Brugervejledning, vers. 2, april 1984

er bragt i overensstemmelse med COMAL-80, version 2.0.

I tilfælde af eventuelle fejl i den COMAL-80 version, der er implementeret, vil disse ved lejlighed blive omtalt i COMET Brugerorientering.

3 Indtastning

INDHOLDSFORTEGNELSE KAPITEL 3 - OPSTART

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

3. OPSTART AF COMAL - 80

Hvis man ønsker at starte den 7-cifrede version af COMAL-80, version 2.0, kan denne startes på to måder:

1. I lighed med tidligere versioner, skal man fra CP/M skrive: COMAL-80, hvorefter fortolkeren indlæses. På skærmen vil der foruden præsentationen være stillet et spørgsmål:

   Ønskes tekster ved fejlmeldinger (J/N) ?

   Ved at svare N(ej) vil man få noget mere lagerplads til rådighed. Størrelsen af den disponible lagerplads kan man få oplyst ved at skrive SIZE.

2. Under forudsætning af, at der på den diskette, hvorfra COMAL-80 indlæses, befinder sig en såkaldt initialiseringsfil, vil fortolkeren blive indlæst og udføre de instruktioner, der står i filen.

3.1 BESKRIVELSE OG ANVENDELSE AF INITIALISERINGSFILEN

Initialiseringsfilen for den 7-cifrede version **skal** have navnet **COMAL80I.NIT**, og for den 13-cifrede version **skal** navnet være **COMAL80DI.NIT**. Uanset navn kan filen genereres eller editeres som et COMAL-80 program bestående udelukkende af såkaldte **kommentarer**, fordi fortolkeren under indlæsning springer de første 7 tegn i hver linie over - altså netop 4 cifre (i linienummer) et blanktegn og (REM-) kommentarsumbolet //.

Kommandoerne AUTO (se side 8.2) og EDIT (se side 8.3) kan ikke benyttes i initialiseringsfilen.

I det følgende gives et eksempel på, hvorledes en initialiseringsfil kan se ud. En forklaring på de anvendte instruktioner kan man finde de respektive steder i denne brugervejledning.

Den første linie **skal** indeholde svaret på spørgsmålet, om der ønskes fejltekster. Mulighederne bliver J(a), N(ej) eller S(pørg). I samme (første linie) kan evt. angives det **decimale heltal**, der instruerer COMAL-80, om hvad den højeste adresse i CPU'en må være.

Eksempel: **Initialiseringsfilen indeholder kun kommandoer.**

```
0010 // S
0020 // ZONE:=8; PAGEWITH:=80; PAGELENGTH:=72
0030 // IDENTIFIERLOWER:=0; KEYWORDLOWER:=0
0040 // CLEAR
0050 // CURSOR 30,10
0060 // PRINT "Menu indlæses...";
0070 // LOAD MENU
0080 // RUN
```

Med ovenstående initialiseringsfil opnås følgende:

10 - Der spørges, om fejltekster ønskes.

20 - Skrivefeltets tabulatorværdi sættes til 8, og ved udskrift på printer fås linielængde 80 og 72 linier pr. side (A4-format).

30 - Identifikatorer (variabelnavne bestemt af bruger) udskrives med små bogstaver (0 er default-værdi for både IDENTIFIERLOWER, KEYWORDLOWER og ZONE) ved listning på skærm eller printer.

40-60 - Skærmen bliver (efter besvarelsen) blank, og det meddeles, at et "menuprogram" indlæses.

80 - Kommandoen RUN bevirker, at menuprogrammet eksekveres.

Initialiseringsfilen bør genereres med COMAL-80 og lagres på diskette ved kommendoen LIST COMAL80I.NIT (uden /C efter navn).

Dersom fortolkeren under indlæsning af initialiseringsfilen møder en fejl, meldes dette, men man kan ikke som ved almindelige programmer afbryde med ESC-tasten (se afsnit 3.4). Man må i stedet benytte sig af editeringsfunktionerne (se side 8.3) til at rette, evt. slette den/de forkerte linie(r). Man kan herefter indlæse filen med kommandoen ENTER (se 8.9) og foretage de fornødne rettelser.

Eksempel: **Initialiseringsfilen indeholder programsætninger.**

```
0010 // S
0020 // 0020 DIM A$ OF 1
0030 // 0030 ZONE:=8; PAGEWITH:=80; PAGELENGTH:=72
0040 // 0040 PROC JNINPUT(T$, REF S$) CLOSED
0050 // 0050   PRINT T$;
0060 // 0060   REPEAT
0070 // 0070     POKE 256, 255
0080 // 0080     REPEAT
0090 // 0090       A:=PEEK(256)
0100 // 0100     UNTIL A<255
0110 // 0110     IF A>96 THEN A:-32
0120 // 0120     S$=CHR$(A)
0130 // 0130   UNTIL S$ IN "JN"
0140 // 0140   PRINT S$
0150 // 0150 ENDPROC JNINPUT
0160 // 0160 EXEC JNINPUT("Variabelnavne med små bogstaver (J/N) ? ",S$)
0170 // 0170 IDENTIFIERLOWER:=(S$="J")
0180 // 0180 PRINT
0190 // 0190 EXEC JNINPUT("Nøgleord med små bogstaver      (J/N) ? ",S$)
0200 // 0200 KEYWORDLOWER:=(S$="J")
0210 // 0210 PRINT
0220 // 0220 END
0230 // RUN
0240 // NEW
```

Som det fremgår af ovenstående, kan man efter "REM"-symbolet tilføje et linienummer efterfulgt af en programsætning. I linie 150 står den kommando, som bevirker, at programmet afvikles inden kontrollen af COMAL-80 overlades til brugeren. Udelades linierne 150-160, vil alle programsætninger blot blive indlæst i hovedlageret som et program, der kan afvikles med kommandoen RUN. Kommandoen i linie 160 bevirker, at hovedlageret efter initialiseringsprogrammets afvikling ikke indeholder det i eksemplet indtastede program.

Da det er muligt at foretage ændringer i CP/M's loaderprogram, således at COMAL-80 indlæses automatisk, når COMET'en tændes, er det herved muligt at lade helt uøvede få adgang til en række færdige programmer.

3.2 PROGRAMSÆTNINGER

I COMAL-80 indledes en programsætning altid med et linienummer mellem 1 og 9999 efterfulgt af en eller flere instruktioner (som fortolkeren kan forstå) og afsluttes altid ved at trykke på RETURN-tasten.

Hvis sætningen er forkert konstrueret (syntax-fejl), gives en

fejlmelding, og den indtastede linie "leveres" tilbage, så den kan rettes. Dersom der er fejl i programstrukturen, stopper programmet og angiver en eller flere fejltyper samt nummer på de(n) linie(r), der kan være årsag til fejlen(e).

Dersom en programsætning tildeles et i forvejen eksisterende linienummer, vil den eksisterende programsætning blive erstattet af den sidst indtastede.

Programsætninger er beskrevet i kapitel 5.

3.3 KOMMANDOER

En kommando skal som en programsætning afsluttes ved at aktivere RETURN-tasten. Kommandoer adskiller sig fra de øvrige COMAL-80 instruktioner ved at være uden noget linienummer, og de bliver udført øjeblikkeligt. Nogle kommandoer kan slet ikke optræde som instruktion i en programsætning (f.eks. SIZE og AUTO).

Kommandoer er beskrevet i kapitel 8.

3.4 RETURN- OG ESC-TASTENS FUNKTION

Enhver kommando eller programlinie afsluttes ved at trykke på tastaturets RETURN-tast. Under indtastningen kan brugeren benytte de på side 8.3 beskrevne editeringstaster.

Under indtastning af en kommandolinie eller ved indtastning/-editering af en programlinie kan brugeren afbryde ved at trykke på ESC-tasten. Forudsat, at der i et program ikke er instruktioner, der forhindrer brug af ESC-tasten, vil aktivering af denne stoppe programafviklingen.

4 Sprogets grundelementer

INDHOLDSFORTEGNELSE KAPITEL 4 - SPROGETS GRUNDELEMENTER

International Computers Limited a/s

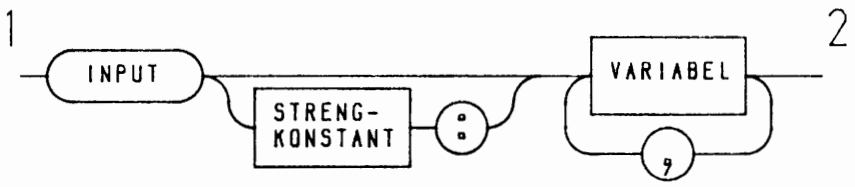
Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

4. SPROGETS GRUNDELEMENTER

4.1 INDLEDNING

COMAL-80 sprogets syntaks beskrives i denne vejledning ved hjælp af syntaksdiagrammer, som de kendes fra Wirth's PASCAL-rapport*. Et eksempel på tolkning af syntaksen, for et af COMAL-80's sprogelementer, gennemgås herunder.



Diagrammet angiver en lidt forenklet syntaks for en INPUT-sætning. En vej gennem diagrammet, startende ved pilen, som peger ind i INPUT-rammen (1), og sluttende ved pilen i diagrammets højre side (2), beskriver en syntaktisk korrekt INPUT-sætning. Terminal-symboler, d.v.s. de symboler, som direkte skrives i et COMAL-80 program, er omgivet af afrundede rammer; symbolerne omgivet af kantede rammer angiver navne på begreber, der er forklaret i andre syntaksdiagrammer eller beskrivelser.

Man kan således af diagrammet se, at en sådan INPUT-sætning skal begynde med nøgleordet 'INPUT'. Efter 'INPUT' kan evt. følge en strengkonstant afsluttet af et ':', efterfulgt af en række kommaadskilte variable. Eksempler på legale INPUT-sætninger er vist herunder:

```
INPUT ANTAL#, STKPRIS
INPUT "NAVN":NAVN$
```

ANTAL#, STKPRIS OG NAVN$ er variable og "NAVN" er en strengkonstant.

* Kathleen Jensen and Niklaus Wirth:
Pascal, User Manual and Report, Springer-Verlag, 1978.

I beskrivelsen af syntaksen for visse sammensatte sprogkonstruktioner er linieskift illustreret, ved at man har ladet diagrammet fortsætte på en ny linie. Se eksempelvis afsnit 5.9.3.3.

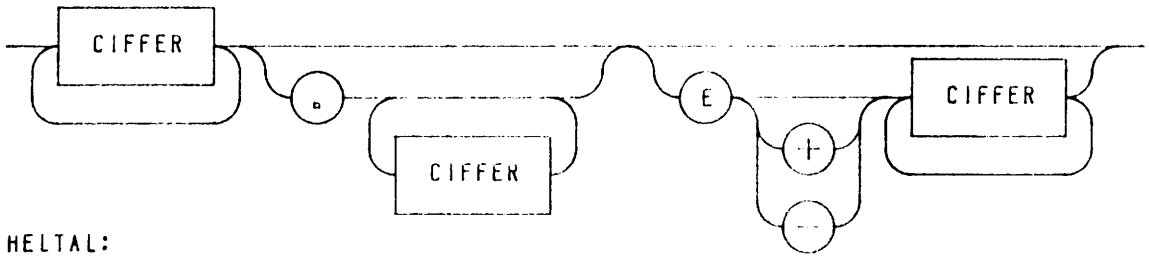
I det følgende kan blanktegn, med mindre andet udtrykkeligt er

anført, forekomme overalt i en sætning uden at ændre dennes betydning.

4.2 TALKONSTANTER

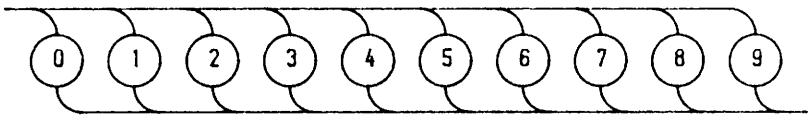
Tal benyttes som konstanter i aritmetiske udtryk, i inddata samt i CASE- og DATA-sætninger. Der findes reelle tal samt heltal, og deres opbygning er vist herunder.

REELT TAL:



HELTAL:

CIFFER:



Eksempler:

Heltal:	1	0	10	3002
Reelle tal:	348	2.	87.13E-14	0

Kommentarer:

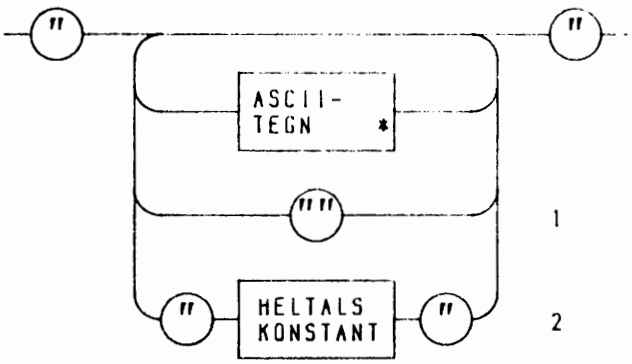
Blanktegn må ikke forekomme i talkonstanter.

Der bruges decimalpunktum i stedet for decimalkomma.
E'et er et stort E. I sætningen: "A:=-17" opfattes tallet, -17, ikke som en heltalskonstant. Det er at opfatte som et monadisk minus foran heltalskonstanten, 17.

4.3 STRENGKONSTANTER

Strenge benyttes som konstanter i strengudtryk, i inddata, i ledetekster ved brug af INPUT-sætningen samt i CASE- og DATA-sætninger. En strengkonstant har følgende opbygning:

STRENG-KONSTANT:



* Her må et vilkårligt, synligt ASCII-tegn (inklusive mellemrum), undtagen anførselstegn ("), forekomme.

Som det ses af diagrammet, startes og afsluttes enhver strengkonstant med et anførselstegn. Ønsker man, at en strengkonstant skal indeholde et anførselstegn, angives dette med 2 anførselstegn umiddelbart efter hinanden (1). Ønsker man, at en strengkonstant skal indeholde tegn, som ikke kan indtastes fra tastaturet, kan man angive tegnets decimale ASCII-kode omgivet af anførselstegn (2). Der må ikke forekomme blanktegn mellem tallet og anførselstegnene på venstre og højre side af det.

Eksempler:

```
"TASTES ""S"" STANDSER PROGRAMMET"
```

Ovenstående strengkonstant vil, hvis den eksempelvis udskrives på en dataskærm, se således ud:

```
TASTES "S" STANDSER PROGRAMMET
```

Strengkonstanten

```
""ABC"13""
```

indeholder de 3 tegn ABC samt tegnet med den decimale ASCII-kode 13 (CR).

Strengkonstanten

"SM4"

indeholder de tre karakterer: S, M og 4.

4.4 IDENTIFIKATORER

Identifikatorer benyttes til navngivning af forskellige størrelser i et COMAL-80 program. En identifikator må maksimalt indeholde 80 tegn, hvoraf det første skal være et bogstav. De 80 tilladte tegn i en identifikator er:

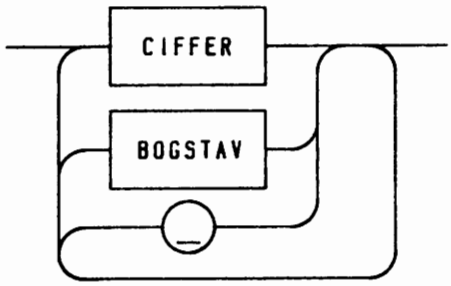
- små og store bogstaver
- cifre
- understregning

COMAL-80 skelner ikke mellem store og små bogstaver i variabelnavne, hvorfor

Hansen hhv.
HANSEN

af COMAL-80 vil opfattes som værende identiske.

IDENTIFIKATOR:



Eksempler på legale identifikationer:

IND-LØN
F16
Anne

Eksempler på illegale identifikationer:

1ABC	(Første tegn er ikke et bogstav).
IND-LØN	(Tegnet "-" må ikke forekomme i en identifikator).

4.5 VARIABLE

4.5.1 INDLEDNING

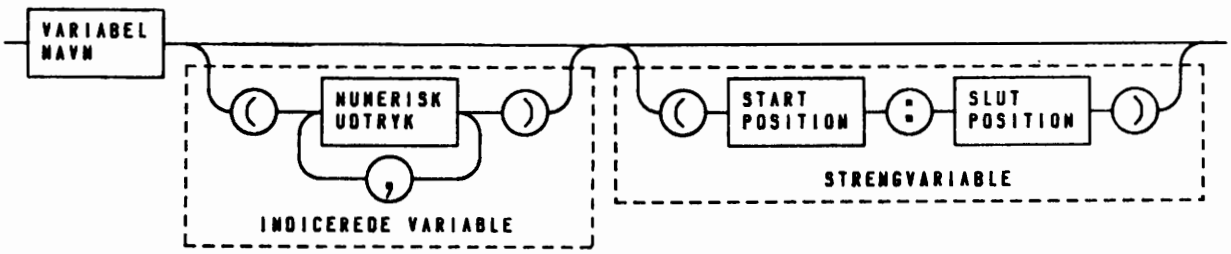
En variabel er et navngivet område i hovedlageret, hvis indhold man har mulighed for at ændre og bruge. Afhængigt af den variables art kan den tildeles et indhold bestående af:

Et tal
Et talsæt
En tegnfølge eller et sæt af tegnfølger

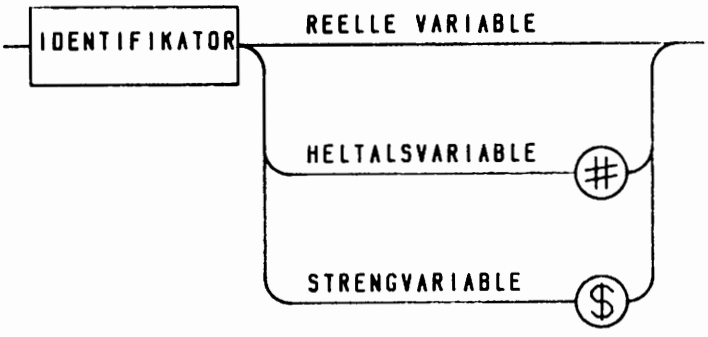
Udover de brugerdefinerede variable, som beskrives her, findes der systemvariable, som er beskrevet i afsnit 6.4.

Indledningsvis bruges her et diagram, der viser, hvorledes man kan referere til indholdet eller en del af indholdet af en variabel:

VARIABEL:



VARIABELNAVN:



Man ser, at variable kan deles op i 3 typer, nemlig reelle variable, heltalsvariable og strengvariable.

Reelle variable og heltalsvariable kaldes under et for numeriske variable. Betegnelserne refererer naturligvis til den type af værdier, som kan tildeles de variable.

Det absolutte værdiområde for et reelt tal, X, er:

Tallet 0 (nul) samt:

Med 7 cifres præcision:

$2.9387359E-39 \leq X \leq 1.7014117E38$

Med 13 cifres præcision:

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

Værdiområdet for et heltal, X , er ved begge grader af præcision:

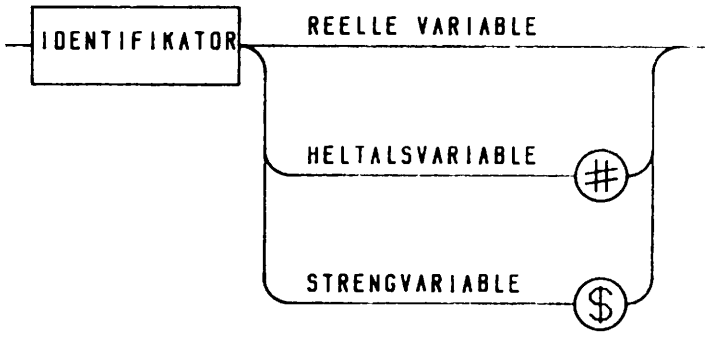
$-32767 \leq X \leq 32767$

Længden af en strengvariabel kan principielt variere mellem 0 (tom streng) og 65535. I praksis vil længden dog være begræset af størrelsen på det tilgængelige hovedlager for den maskine, man vælger at udføre programmet på, d.v.s. mange tusinde karakterer.

I det følgende vil indholdet af diagrammet blive gennemgået således, at man starter med det simpleste specialtilfælde, nemlig simple variable (eksklusive spørgsmålet om delstrenge).

Formatet i det simple tilfælde er:

SIMPEL VARIABEL:
(EKSKL. DELSTRENGE)



Spørgsmålet om delstrenge i forbindelse med simple eller indicerede strengvariable vil blive behandlet i et afsnit for sig.

4.5.2 SIMPLE VARIABLE (EKSKLUSIVE DELSTRENGE)

4.5.2.1 SIMPLE REELLE VARIABLE

En simpel reel variabel er en variabel, som kan tildeles en numerisk værdi (d.v.s. et reelt tal, som evt. kan være helt), og som beholder denne værdi, indtil den tildeles en ny værdi.

En simpel reel variabel kan bruges i et numerisk udtryk, det vil sige et udtryk, der angiver et tal.

F.eks. kan man bruge den simple reelle variabel, PRIS, i følgende lille program:

```
0010 PRIS:=13.5+6.5
0020 PRINT "Totalt momsbeløb = ";PRIS*0.22
0030 END
```

Der ved kørsel vil give udskriften:

```
TOTALT MOMSBELØB=4.4
```

4.5.2.2 SIMPLE HELTALSVARIABLE

En simpel heltalsvariabel er en variabel, som kan tildeles en heltallig værdi, og som bibeholder denne værdi, indtil den tildeles en ny heltallig værdi.
En simpel heltalsvariabel kan bl.a. bruges i et heltalligt udtryk, d.v.s. et udtryk, der angiver et heltal. Eksempelvis kan man bruge en simpel heltalsvariabel, som man f.eks. kalder ANTAL#, i følgende lille program:

```
0010 ANTAL#:=50*20
0020 PRINT ANTAL#+1:"nat"
0030 END
```

der ved kørsel vil give udskriften:

```
1001 NAT
```

Bemærk, at "#" efter identifikatoren, ANTAL, angiver, at det er en heltalsvariabel.

4.5.2.3 SIMPLE STRENGVARIABLE (EKSKL. DELSTRENGE)

En simpel strengvariabel er en variabel, som kan tildeles en strengværdi (en tegnfølge), og som bliver ved med at indeholde denne tegnfølge, indtil den tildeles en ny tegnfølge.

En simpel strengvariabel kan bl.a. bruges i et strengudtryk, d.v.s. et udtryk, der angiver en tegnfølge.

F.eks. kunne man vælge at kalde en simpel strengvariabel for NAVN$ og bruge den i følgende lille program:

```
0010 DIM NAVN$ OF 8
0020 NAVN$:="PETER"
0030 PRINT NAVN$+"SEN"
0040 END
```

der vil give udskriften:

PETERSEN

En simpel strengvariabel svarer altså til en simpel numerisk variabel; den indeholder blot en tegnfølge i stedet for en numerisk værdi.

4.5.2.4 ERKLÆRING AF SIMPLE VARIABLE

Som man ser, skal simple variable ikke erklæres før brugen, medens simple strengvariable skal erklæres først, i en DIM-sætning.

Dette hænger sammen med den forskel, der er i behov for lagerplads for de forskellige typer.

Lagerpladsen for en simpel numerisk variabel er entydigt bestemt ved dens type og den præcision, man kører med.

Man skal derfor ikke udtrykkeligt specificere den plads, der skal bruges til en simpel numerisk variabel.

Lagerpladsen for en simpel strengvariabel er bestemt ved længden (i antal tegn) af den længste tegnfølge, som den skal kunne indeholde.

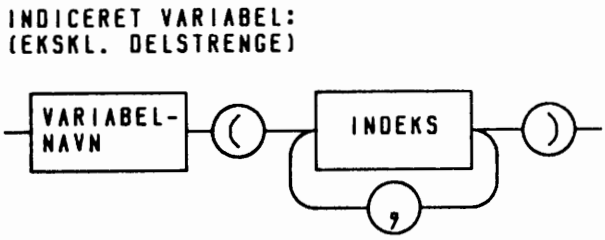
Bl.a. fordi det giver mulighed for hurtigere programeksekvering, har man valgt at kræve følgende:

Den største længde af det, som en simpel strengvariabel skal kunne indeholde, skal specificeres for denne variabel i en DIM-sætning (se afsnit 5.3).

4.5.3 INDICEREDE VARIABLE (EKSKL. SPØRGSMÅLET OM DELSTRENGE)

4.5.3.1 INDLEDNING

Reference til en enkelt indiceret variabel følger nedenstående format:



Indeks skal være et numerisk udtryk, der under udførelsen afrundes internt i maskinen i overensstemmelse med ROUND-funktionen.

Variabel-navn følger samme regler som for simple variable. Indicerede variable kan bortset fra i FOR ... NEXT løkker (beskrevet i 5.10.2) bruges ligesom simple variable; deres eget format er blot anderledes.

Med hensyn til sættet af en række sammenhørende indicerede variable henvises til afsnit 4.5.5, "Reference til arrays".

Begrebet, en indiceret strengvariabel, kan være vanskeligt at forstå. Nedenstående gennemgang vil derfor starte med en indgående beskrivelse af analoge begreber.

4.5.3.2 INDICEREDE REELLE VARIABLE

En indiceret reel variabel er et element i en tabel over reelle variable. Man refererer til den indicerede variable ved at angive tabellens navn efterfulgt af en parentes med det nødvendige antal indices. En sådan tabel kaldes en reel array-variabel eller blot: Et reelt array.

Da den plads, der skal bruges til en tabel, afhænger af, hvor mange elementer tabellen indeholder, skal tabeller specificeres i en separat DIM-sætning.

Sætningen:

```
DIM PR(5)
```

specificerer et endimensionalt reelt array (også kaldet en reel vektor), der består af 5 elementer, nemlig de indicerede variable:

```
PR(1) PR(2) PR(3) PR(4) PR(5)
```

Disse kan f.eks. tildeles tallene h.h.v.

```
7.5  8.22  -0.10  13.0  0.0
```

Som eksempel kan nævnes programmet:

```
0010 DIM TAL(6)
0020 TAL(1):=1
0030 FOR I:=2 TO 6 DO
0040   TAL(I):=2*TAL(I-1)
0050   PRINT TAL(I);
0060 NEXT I
0070 END
```

der vil give udskriften:

```
2   4   8   16   32
```

Sætningen:

```
DIM P(2,3)
```

specificerer et todimensionalt reelt array (også kaldet en reel matrix).

Denne består af 6 elementer, nemlig de indicerede reelle variable:

```
P(1,1)    P(1,2)    P(1,3)
P(2,1)    P(2,2)    P(2,3)
```

Disse kan f.eks. tildeles tallene h.h.v.:

```
2.1        2.1          5.701
10.0E20    22.01        13.0
```

Sætningen:

```
DIM R(2,3,2)
```

specificerer, d.v.s. sætter plads af til, et tredimensionalt reelt array.

Dette består af 2*3*2=12 elementer, nemlig de indicerede reelle variable:

```
R(1,1,1)        R(1,1,2)
R(1,2,1)        R(1,2,2)
R(1,3,1)        R(1,3,2)
R(2,1,1)        R(2,1,2)
R(2,2,1)        R(2,2,2)
R(2,3,1)        R(2,3,2)
```

Disse kan f.eks. tildeles tallene h.h.v.:

```
 7.2             3.1
 1.2             0.2
-1.3             5.8
 8.0             7.1
-9.44           -0.02
 1.3             5.4
```

4.5.3.3 INDICEREDE HELTALSVARIABLE

Disse optræder helt analogt med indicerede reelle variable.

Eksempelvis kan man med sætningen:

```
DIM PRIS (2,3,2)
```

specificere et tredimensionalt heltals-array med elementerne:

```
PRIS (1,1,1)   PRIS (1,1,2)
PRIS (1,2,1)   PRIS (1,2,2)
PRIS (1,3,1)   PRIS (1,3,2)

PRIS (2,1,1)   PRIS (2,1,2)
PRIS (2,2,1)   PRIS (2,2,2)
PRIS (2,3,1)   PRIS (2,3,2)
```

Disse kan f.eks. tildeles de heltallige værdier h.h.v.:

```
31              27
25              21
20              27

62              45
65              46
62              50
```

således som det gøres i program-eksemplet i nedenstående afsnit om indicerede strengvariable.

4.5.3.4 INDICEREDE STRENGVARIABLE (EKSKLUSIVE DELSTRENGE)

En indiceret strengvariabel er et element i et strengarray. Den svarer således nøje til en indiceret numerisk variabel.

Sætningen

```
DIM VARER$(5) OF 3
```

specificerer et endimensionalt strengarray (også kaldet en strengvektor), der består af 5 elementer, nemlig de indicerede strengvariable:

VARER$(1) VARER$(2) VARER$(3) VARER$(4) VARER$(5)

Disse kan hver især tildeles en tegnfølge på op til 3 tegn. F.eks. kan de tildeles h.h.v.:

ULD LAM SKY SNE KUL

VARER$(4) kan f.eks. tildeles karaktererne S, N og E ved sætningen:

```
VARER$(4):="SNE"
```

Sætningen:

```
DIM VAND$(2,3) OF 4
```

specificerer et variabelt todimensionalt strengarray (også kaldet en streng-matrix). Denne består af 6 elementer, nemlig de indicerede strengvariable:

```
VAND$(1,1)   VAND$(1,2)   VAND$(1,3)
VAND$(2,1)   VAND$(2,2)   VAND$(2,3)
```

Disse variable kan f.eks. tildeles strengfølgerne h.h.v.:

```
VAND         WODA         VANN
AGUA         AQUA         SKYL
```

Det kan diskuteres, om man i denne beskrivelse skulle angive indholdet på formatet for en strengkonstant, som det er defineret i afsnit 4.3:

```
"VAND"       "WODA"       "VANN"
"AGUA"       "AQUA"       "SKYL"
```

Hvis man ønsker at tildele VAND$(2,2) værdien "AQUA", kan det gøres med sætningen: VAND$(2,2):="AQUA"

Sætningen: MENU$(2,3,2) OF 14 specificerer et tredimensionalt strengarray.

Dette består af 12 elementer, nemlig de indicerede strengvariable:

MENU$(1,1,1)	MENU$(1,1,2)
MENU$(1,2,1)	MENU$(1,2,2)
MENU$(1,3,1)	MENU$(1,3,2)
MENU$(2,1,1)	MENU$(2,1,2)
MENU$(2,2,1)	MENU$(2,2,2)
MENU$(2,3,1)	MENU$(2,3,2)

Disse kan f.eks. tildeles tegnfølgerne h.h.v.:

HAKKEBØF	KØDBOLLER
GRILLKYLLING	HØNSESTUVNING
STEGTE SILD	KOGT RØDSPÆTTE
ENGELSK BØF	HAMBURGERRYG
AND A L'ORANGE	COQ AU VIN
STEGTE ÅL	KOGT PIGHVAR

således som det er gjort i nedenstående eksempel.

Eksempel:

```
0010 // Program til valg af middagsmad
0020 DIM MENU$(2,3,2) OF 14
0030 DIM PRIS(2,3,2)
0040 DATA "Hakkebøf", 31, "Kødboller", 27
0050 DATA "Grillkylling", 25, "Hønsestuvning", 2
0060 DATA "Stegte sild", 20, "Kogt hamburgerryg", 45
0070 DATA "And a l'orange", 65, "Coq au vin", 46
0080 DATA "Stegte ål", 62, "Kogt pighvar", 50
0090 //
0100 FOR I:=1 TO 2 DO
0110   FOR J:=1 TO 3 DO
0120     FOR K:=1 TO 2 DO
0130       READ MENU$(I,J,K)
0140       READ PRIS(I,J,K)
0150     NEXT K
0160   NEXT J
0170 NEXT I
0180 //
0190 // Nu er menuen indlæst i de indicerede variable
0200 //
```

```
0210 CLEAR
0220 PRINT "Restauranten er gået over til EDB."
0230 PRINT "De bedes derfor angive Deres ønske som følger:"
0240 PRINT "Skal det være aftensmad (1) eller souper (2) ?"
0250 INPUT "Svar med et tal fra 1 til 2 : ": I
0260 PRINT "Skal det være kød (1), fjerkræ (2) eller fisk (3) ?"
0270 INPUT "Svar med et tal fra 1 til 3 : ": J
0280 PRINT "Skal det være stegt (1) eller kogt (2) ?"
0290 INPUT "Svar med et tal fra 1 til 2 : ": K
0300 BELØB:=PRIS(I,J,K)
0310 PRINT "De kan få ";MENU$(I,J,K);
0320 PRINT " til en pris af ";BELØB;"kr."
0330 PRINT " - eller opvask af ";INT(47*BELØB);"tallerkener."
0340 END
```

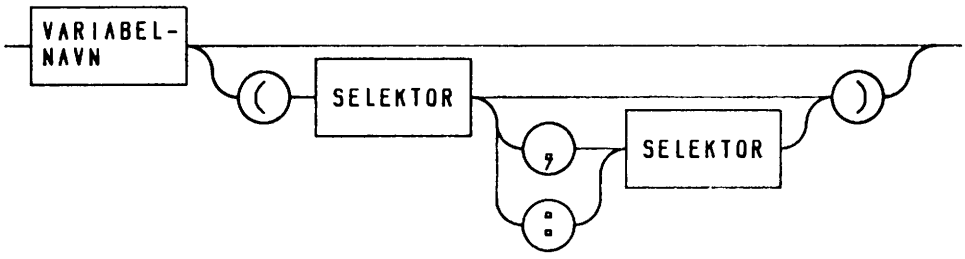
4.5.4 DELSTRENGE

Vi har set, hvorledes man kan referere til en simpel eller indiceret strengvariabel som en helhed.

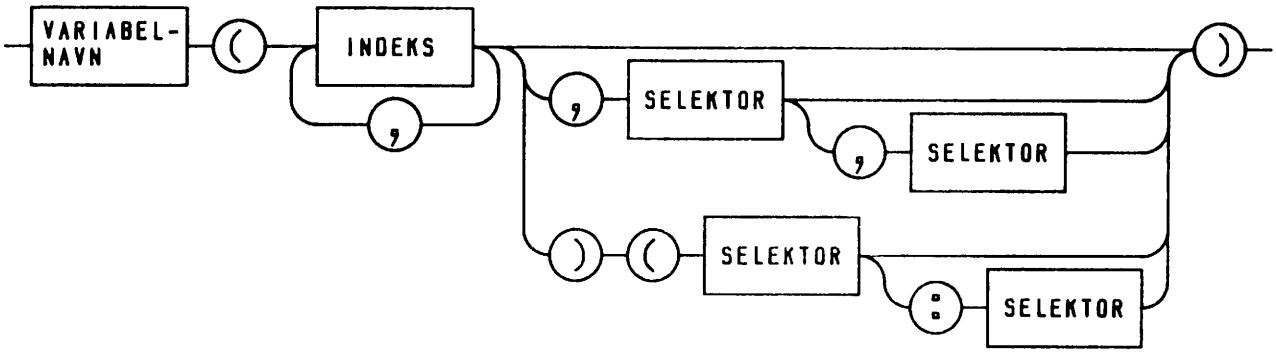
Man har yderligere mulighed for at referere til en del af en sådan variabel.

Når denne mulighed inkluderes, udføres referencen efter et af disse to formater:

SIMPEL STRENGVARIABEL:



INDICERET STRENGVARIABEL:



Variabelnavn er i dette tilfælde en identifikator, umiddelbart efterfulgt af et $ tegn. Ind. og Sel. står for h.h.v. Indeks og Selektor.

Indeks kommer af latin: Index, der betyder finger, dvs. noget der udpeger.

Selektor kommer af latin: Selecto, jeg udvælger.

Indices bruges som sædvanligt til at udpege det enkelte element inden for tabellen. Selektorer bruges til at udvælge den eller de ønskede positioner inden for elementet. Ind. og Sel. skal angives i form af et numerisk udtryk. Internt i maskinen bliver dette udtryk udregnet og afrundet i overensstemmelse med ROUND-funktionen.

FUNKTION

Hvis referencen i øvrigt er syntaktisk korrekt, starter maskinen med at betragte dimensionstallet, N, for strengvariablen, d.v.s. hvor mange indices, den er dimensioneret til.

For simple strengvariable kan man her opfatte N's værdi som nul. De N første komma-adskilte numeriske udtryk bliver derefter opfattet som indices. Det er et krav, at de alle er angivet i første parentes. De øvrige numeriske udtryk i skemaet bliver opfattet som selektorer. Selektorernes antal (ikke at forveksle med deres værdier) skal være enten 0, 1 eller 2.

Hvis der ikke er nogen selektorer, har vi det normale tilfælde, hvor der refereres til hele den simple eller indicerede strengvariable.

Hvis der er netop 1 selektor, betyder det, at man udvælger den dertil svarende position i elementet.

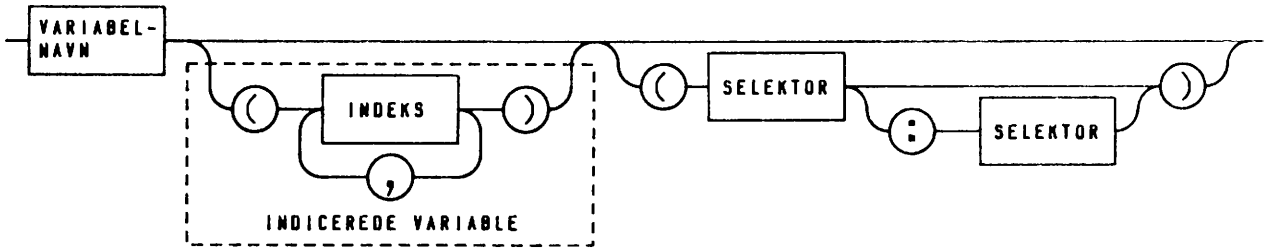
En selektorværdi på 12 vil f.eks. udvælge den tolvte position (den første karakter står i positionen helt til venstre).

Hvis der er 2 selektorer adskilt med kolon (:), betyder det, at man udvælger fra og med den først angivne til og med den sidst angivne position.
F.eks. vil A$(7:10) udvælge positionerne 7, 8 9 samt 10 i den variable A$

KOMMENTARER

Som man kan se, er der flere forskellige måder at angive den samme ting på. Det kan anbefales, at man bruger ét bestemt format, nemlig dette:

SIMPEL ELLER INDICERET STRENGVARIABEL:



Dertil kommer, at det er let at huske: Man bruger det normale format til at udpege og bruger den efterfølgende parentes til at udvælge. Yderligere er det godt til dokumentationsformål, fordi man undgår misforståelser m.h.t. antallet af indices.

Eksempler

```
0010 // Eksempel på brug af simple strengvariable
0020 DIM SIM$ OF 5
0030 SIM$:="ABCDE"
0040 PRINT SIM$;"=ABCDE"
0050 PRINT SIM$(1,5);"=ABCDE"
0060 PRINT SIM$(1:5);"=ABCDE"
0070 PRINT SIM$(2);"=B"
0080 PRINT SIM$(3);"=C"
0090 PRINT SIM$(3:3);"=C"
0100 // Følgende udtryk er ikke korrekte
0110 // SIM$(0)              SIM$(6)
0120 // SIM$(3,3,4)          SIM$(4,3)
0130 // SIM$(1,4)(2:3)       SIM$(4:1)
```

```
0140 //
0150 // Navnet SIM kan bruges i andre forbindelser:
0160 SIM:=13.7
0170 SIM#:=12
0180 PRINT "SIM$(1,2,3,4,5,6,7,8)"
0190 PRINT SIM;SIM#
0200 END


0010 // Eksempel på brug af indicerede strengvariable
0020 DIM IND$(3,2) OF 4
0030 IND$(3,1):="ABCD"
0040 PRINT IND$(3,1);"=ABCD"
0050 PRINT IND$(3,1,1,4);"=ABCD"
0060 PRINT IND$(3,1)(1:4);"=ABCD"
0070 PRINT IND$(3,1,2);"=B"
0080 PRINT IND$(3,1,2,2);"=B"
0090 PRINT IND$(3,1)(2:2);"=B"
0100 PRINT IND$(3,1,2,3);"=BC"
0110 PRINT IND$(3,1)(2:3);"=BC"
0120 // Følgende udtryk er ikke korrekte:
0130 // IND$(4,1)              IND$(3,1)
0140 // IND$(3,1,0,2)          IND$(3,1,3,6)
0150 // IND$(3,1)(2,3)         IND$(3,1,2)(3)
0160 // IND$(3,1)(2:3)         IND$(3,1,2,3,4)
0170 // IND$(3,1,3,1)          IND$(3)
0180 // IND$(3)(2:3)
0190 //
0200 // Navnet IND kan bruges i andre forbindelser:
0210 IND:="D" IN "D"
0220 INDDATA:=IND
0230 DIM IND#(7)
0240 MAT IND#:=TRUNC(INDDATA)
0250 PRINT "IND$(3,1,9,7,5,4)(3:5)"
0260 PRINT IND;IND#(5);INDDATA;"IND" IN "INDDATA"
0270 END
```

4.5.5 REFERENCE TIL ARRAYS

Når man taler om variable, så mener man almindeligvis: Simple eller indicerede variable. Imidlertid kan også et sæt (eller array) opfattes som en form for variabel.

En sådan variabel erklæres i en DIM-sætning (se afsnit 5.3).
Den kan tildeles værdier i en MAT-sætning.
Den kan bruges som parameter i en EXEC-sætning.
Den kan erklæres for IMPORT (GLOBAL) i en lukket programdel.

I forbindelse med MAT, EXEC og IMPORT refereres der blot til den variable ved dens navn. Med REF skal man referere til en array-variabel, der bruges som formel parameter i en proce-

dure-sætning (PROC), eller funktions-sætning (FUNC).

Ved denne reference angiver man navnet, efterfulgt af en parentes med en række kommaer i.

Antallet af kommaer er lig dimensionstallet minus 1. Hvis der hører 2 indices til en variabel, skal der altså 1 komma i parentesen.

Alle de nævnte typer af referencer til arrays optræder i nedenstående programeksempel, VOLAPYK. Dette program bruges til at kryptere tekst med, d.v.s. at ændre teksten fra klart sprog til en uforståelig tekststreng. Samme program bruges til at dechifrere teksten med igen.

Eksempel

```
0010 // VOLAPYK
0020 // Eksempel på reference til arrays.
0030 // Input til programmet er tekststrenge på 4 karakterer.
0040 // Programmet koder eller dekoder disse strenge.
0050 // Programmet stoppes ved tryk på ESC-tasten
0060 //
0070 DIM ALFA$(4,8) OF 1
0080 DIM B#(4)
0090 DIM TEKST$ OF 4
0100 DIM NYTEKST$ OF 4
0110 DATA 0, 4, 4, 0
0120 DATA "@", "A", "B", "C", "D", "E", "F", "G", "H", "I", "J"
0130 DATA "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U"
0140 DATA "V", "W", "X", "Y", "Z", "Æ", "Ø", "Å", "^", "_"
0150 MAT B#:=3
0160 FOR I:=1 TO 4 DO
0170   READ C#
0180   B#(I):+C#
0190 NEXT I
0200 //
0210 FOR I:=1 TO 4 DO
0220   FOR J:=1 TO 8 DO
0230     READ ALFA$(I,J)
0240   NEXT J
0250 NEXT I
0260 // Nu er værdierne læst ind i ALFA$ og B#
0270 LOOP
0280   INPUT TEKST$
0290   EXEC KODE(ALFA$,TEKST$)
0300   PRINT NYTEKST$
0310 ENDLOOP
```

```
0320 //
0330 PROC KODE(REF A$(,), C$) CLOSED
0340   IMPORT B#, NYTEKST$
0350   FOR E:=1 TO 4 DO
0360     D:=ORD(C$(E))-64
0370     I:=D DIV 8
0380     J:=D MOD 8
0390     I:=I*B#(E) MOD 4
0400     J:=J*B#(E) MOD 8
0410     NYTEKST$(E):=A$(I+1,J+1)
0420   NEXT E
0430 ENDPROC KODE
0440 END
```

4.6 UDTRYK

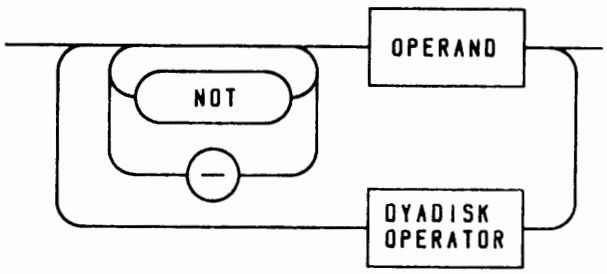
4.6.1 INDLEDNING

Udtryk kan indgå i en række forskellige sætninger. Resultatet af beregningen af et udtryk kan være:

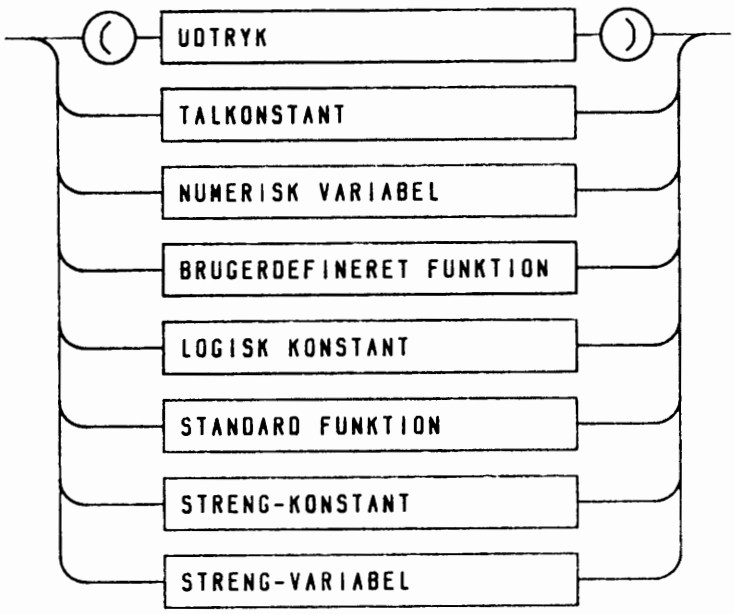
- numerisk
- logisk
  eller
- alfanumerisk

Et udtryk har følgende opbygning:

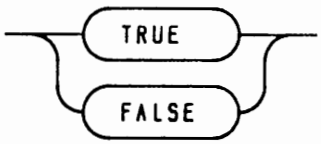
UDTRYK:



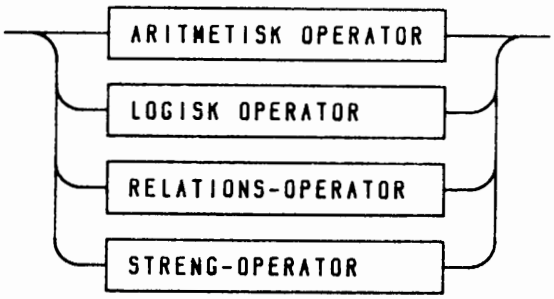
OPERAND:



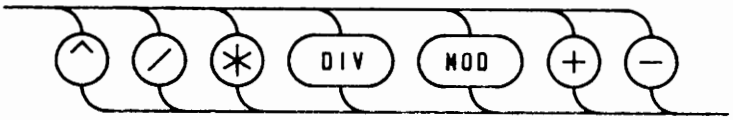
LOGISK KONSTANT:



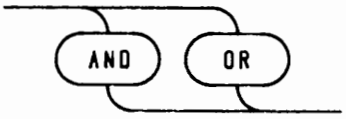
DYADISK OPERATOR:



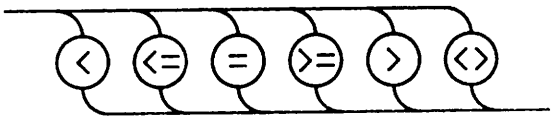
ARITMETISK OPERATOR:



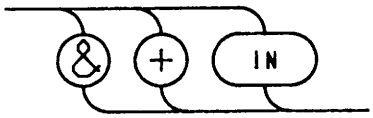
LOGISK OPERATOR:



RELATIONS OPERATOR:



STRENG OPERATOR:



4.6.2 OPERATORER

4.6.2.1 ARITMETISKE OPERATORER

De dyadiske aritmetiske operatorer er følgende:

```
^     : potensopløftning
/     : division
*     : multiplikation
DIV   : heltalsdivision
MOD   : modulus
+     : addition
-     : subtraktion
```

De dyadiske aritmetiske operatorer må kun optræde mellem udtryk, som antager aritmetiske værdier.

Udover de dyadiske operatorer findes den monadiske aritmetiske operator '-' (foranstillet minus), der må optræde foran et udtryk, som antager en aritmetisk værdi. Monadisk + kan ikke bruges.

Resultatet ved anvendelse af en aritmetisk operator er en aritmetisk værdi.

Den aritmetiske værdi af et sandt logisk udtryk er 1. Den aritmetiske værdi af et falsk logisk udtryk er 0.

Kommentarer

X DIV Y er defineret som X/Y nedrundet til nærmeste hele tal.

X MOD Y (modulus) er defineret som den rest, der bliver ved division af X med Y.

Eksempler:

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

4.6.2.2 LOGISKE OPERATORER

De dyadiske logiske operatorer er følgende:

AND: logisk 'og'
OR : logisk 'eller'

De dyadiske logiske operatorer må optræde mellem udtryk, som antager logiske værdier.

Udover de dyadiske operatorer findes den monadiske logiske operator "NOT" (negation), der må optræde foran et udtryk, som antager en logisk værdi.

Resultatet ved anvendelse af en logisk operator er en logisk værdi.

Den logiske værdi af et aritmetisk udtryk, hvis værdi er nul, er 'falsk'. Den logiske værdi af et aritmetisk udtryk, hvis værdi er forskellig fra nul, er 'sand'.

4.6.2.3 RELATIONSOPERATORER

Relationsoperatorerne er følgende:

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

Relationsoperatorerne er alle dyadiske og må optræde mellem to udtryk, som begge antager en aritmetisk værdi, eller to udtryk, der begge antager en alfanumerisk værdi (tegnstrenge).

Resultatet ved anvendelse af en relationsoperator er en logisk

værdi. Hvis relationen er opfyldt, er værdien 'sand', ellers 'falsk'.

Kommentarer

Sammenligningen af to tegnstrenge foretages tegn for tegn fra venstre mod højre. Sammenligningen sker på grundlag af tegnenes ASCII-værdier (se afsnit 9.4).

Hvis to strenge af forskellig længde sammenlignes, og de første tegn i den længste streng er identiske med tegnene i den korteste, da vil den korteste streng være mindst. Eksempelvis er følgende relation sand:

```
"OLE" < "OLESEN"
```

4.6.2.4 STRENGOPERATORER

Strengoperatorerne er følgende:

```
 & :  kontatenering (sammensætning)
 + :             do
IN :  forekomst af en streng i en anden
```

Strengoperatorerne er dyadiske og må optræde mellem udtryk, der antager alfanumeriske værdier.

Resultatet ved anvendelse af '+' eller '&' operatoren er en tegnstreng, nemlig strengen på venstre side af operatoren forlænget med strengen på højre side af operatoren.

Eksempel

```
"ABC" + "DE" = "ABCDE"
```

Resultatet ved anvendelse af 'IN'-operatoren er en logisk værdi. Hvis tegnstrengen på venstre side af operatoren forekommer i tegnstrengen på højre side af operatoren, er værdien 'sand', ellers 'falsk'.

Eksempel

```
"OMA" IN "AROMA" er 'sand'
"OMA" IN "MONA" er 'falsk'
```

4.6.3 BEREGNING AF UDTRYK

Beregning af udtryk sker efter følgende regler:

1. Fortolkningen af et udtryk, hvis interne værdi i maskinen er et tal, afhænger af sammenhængen. Ethvert sådant udtryk kan nemlig enten fortolkes som et logisk eller et aritmetisk udtryk.

   Eksempel

   Programmet:

```
0010 IF ("A"="B")=("B"="C") THEN A:=("A"="A")*17
0020 IF 13 THEN PRINT A
0030 END
```

   vil udskrive tallet, 17.

   Dette program sammenligner den falske værdi, "A"="B", med den falske værdi, "B"="C". Da de begge er falske, er det logiske udtryk sandt, hvorfor den efterfølgende (THEN-) sætning bliver udført. Denne sætning sætter A = et aritmetisk udtryk. Da "A"="A" er sandt, fortolkes det som heltalsværdien, 1. A bliver således = 1*17=17.

   I linie & optræder betingelsen, 13.
   Dette bliver opfattet som et logisk udtryk, der er sandt, fordi det er <> fra 0. Derfor bliver printsætningen udført, og denne vil udskrive tallet, 17.

2. Udtryk beregnes primært efter operatorernes prioritet og sekundært fra venstre mod højre. Operatorerne har følgende prioriteter:

Højest	^
	/ * DIV MOD
	+ - (dyadisk) - (monadisk)
	< <= > >= = <> IN
	NOT
	AND
Lavest	OR

3. Hvis en operand er et udtryk omgivet af parenteser, beregnes dette udtryk til bunds, før operatoren anvendes på operanden.

Eksempel:

```
1.  -3*2+3 ("AB" IN "ABC" AND 3)
2.  -3*2+3 ("AB" IN "ABC" AND 3)
3.  -3*2+3 (      1       AND 3)
4.  -3*2+3              1
5.  -3*2+      3
6.  -6  +      3
        -3
```

Understregningen viser, hvilken beregning der udføres i det aktuelle trin.

Resulterende type ved beregning af udtryk

I skemaet herunder gives en oversigt over, hvilke typer operanderne til de dyadiske operatorer må have samt, hvilken type resultatet får.

Operand type								
venstre	højre	^	/	*	DIV MOD	+ -	IN	AND OR
streng	streng	-	-	-	-	streng*	heltal	
heltal	heltal	reel	reel	heltal	heltal	heltal	-	heltal
heltal	reel	reel	reel	reel	reel	reel	-	heltal
reel	heltal	reel	reel	reel	reel	reel	-	heltal
reel	reel	reel	reel	reel	reel	reel	-	heltal

*) ikke '-' men gerne '&'

Relationsoperatorerne <, <=, >, >=, =, <> må benyttes på vilkårlige par af operander, som antager strengværdier, og på vilkårlige par af operander, som antager aritmetiske værdier. Resultatet bliver et heltal, 1 (sand), eller et heltal, 0 (falsk).

Når der i et resultatfelt i skemaet ovenfor er skrevet et '-', betyder det, at den tilsvarende operator ikke må benyttes med det aktuelle par af operander.

Overskridelse af værdiområdet

Hvis den absolutte værdi af et udtryk, hvis resulterende type er heltallig, beregnes til en værdi, som er større end 32767, forekommer et overløb. Tilsvarende forekommer et overløb, hvis den absolutte værdi af et udtryk, hvis resulterende type er reel, er større end eller lig med 1.7014117E38 (ved anvendelse af 7-cifret præcision).

I tilfælde af overløb, standses programudførelsen med en fejlmeddelelse, medmindre der er udført en TRAP ERR- Hvis den absolutte værdi af et udtryk, hvis resulterende type er reel, er mindre end 2.9387359E-39 uden at være 0, forekommer et underløb (ved 7-cifret præcision).

I tilfælde af underløb sættes værdien af udtrykket til 0, hvorefter programudførelsen fortsætter.

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

5 COMAL-80 sætninger

INDHOLDSFORTEGNELSE KAPITEL 5 - COMAL-80 SÆTNINGER

INDHOLDSFORTEGNELSE KAPITEL 5 - COMAL-80 SÆTNINGER

5. COMAL-80 SÆTNINGER

5.1 INDLEDNING

Et COMAL-80 program består af et antal programsætninger. Hver programsætning består af et linienummer og en COMAL-80 sætning. Linienummeret bestemmer programsætningens placering i programmet, og COMAL-80 sætningen indeholder en instruks til COMAL-80 systemet.

En COMAL-80 sætning indledes normalt med et nøgleord, som identificerer sætningstypen.

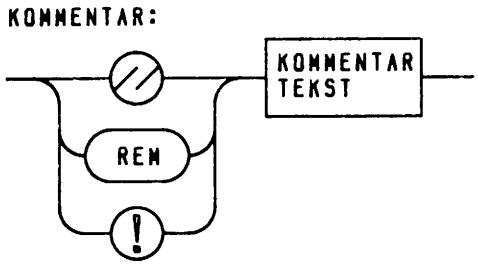
COMAL-80 sætningerne gennemgås i det følgende.

5.2 KOMMENTARER

Man har mulighed for at føje kommentarer ind i et COMAL-80 program. Kommentarer kan optræde som rene kommentarsætninger eller som kommentarer, der afslutter andre COMAL-80 sætninger.

Syntaks

En kommentar har følgende opbygning:



Udførelse

Kommentarer har ingen indflydelse på programudførelsen.

Kommentarer

I programudskrifter konverteres et indledende 'REM' eller '!' til '//'.

Eksempler:

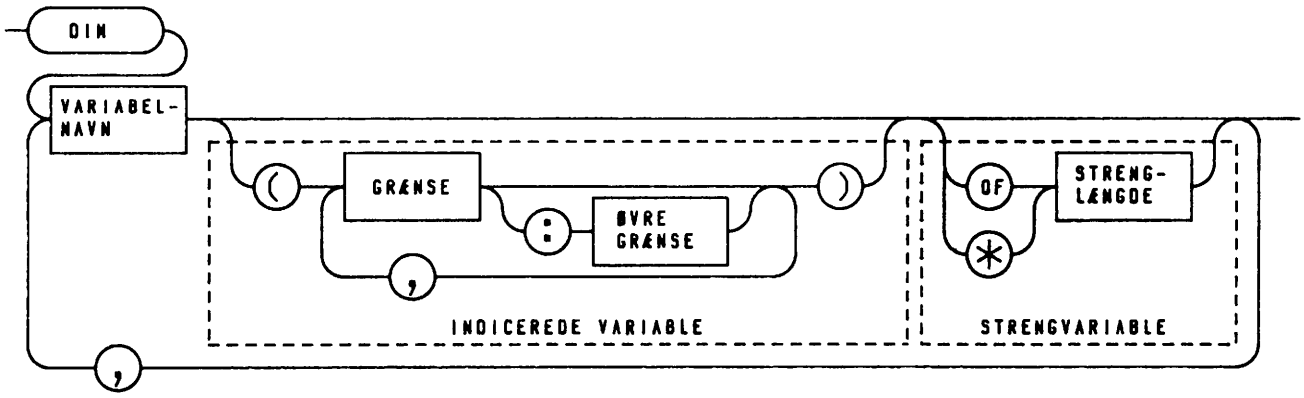
```
0010 REM Hovedprogram
0020 EXEC CPRNR // CPRnr indlæses og kontrolleres
0030 ! Udråbstegn kan også bruges
0040 ! men konverteres
0050 REM ligesom REM til //
0060 // i programudskrifter
0070 ////////////////////////
```

5.3 ERKLÆRINGER

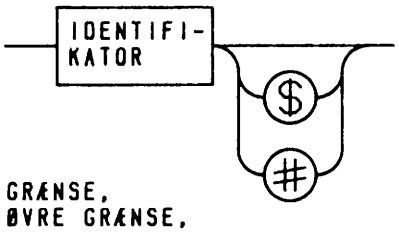
Erklæring af talsæt, tegnstrenge samt sæt af tegnstrenge foretages ved hjælp af DIM-sætningen.

Syntaks

DIM-sætningen har følgende opbygning:



VARIABEL-NAVN:



GRÆNSE,
ØVRE GRÆNSE,
STRENGLÆNGDE:

Udførelse

Ved udførelse af DIM-sætningen afsættes plads i dataområdet til de størrelser, der erklæres.

Kommentarer

1. Dimensionen af et variabel-sæt kan være vilkårligt stor og er kun begrænset af arbejdslagerets størrelse (ved udførelse).

2. Indeksgrænser angives på et af følgende to formater:

   1) 'nedre grænse' : 'øvre grænse'
   2) 'øvre grænse'

   Følgende ulighed skal være opfyldt for udtrykkene efter afrunding: 0≤ 'nedre grænse' ≤ 'øvre grænse'

   Hvis format nr. 2 anvendes, er en nedre grænse på 1 underforstået.

3. Hvis resultatet af beregningen af 'grænse', 'øvre grænse' eller 'strenglængde' ikke giver et helt tal, foretages en afrunding internt i maskinen i overensstemmelse med ROUND-funktionen.

4. Erklæringer skal udføres, inden de erklærede størrelser første gang benyttes i programmet.

5. Alle elementer i et erklæret talsæt gives værdien 0; værdien af en erklæret strengvariabel gives værdien "tom streng" (længde = 0).

6. Formelle parametre skal ikke dimensioneres i en procedure.

7. Groft sagt må en variabel kun erklæres én gang. En nøjere beskrivelse er lidt mere kompliceret:
   Samme variabel kan ikke dimensioneres mere end en gang.

   Med "dimensioneres" menes her, at den tilsvarende DIM-sætning faktisk bliver udført under programforløbet. En variabel kan altså i alle tilfælde optræde i flere DIM-sætninger, hvis kun en af dem bliver udført. Med "samme variabel" refereres der til et bestemt variabelnavn inden for samme lukkede programdel eller uden for de lukkede programdele. Hver gang en lukket programdel kaldes, kan hver enkelt af programdelens DIM-sætninger udføres en gang. Dette er endog tilladt og kan være påkrævet ved rekursive kald i en lukket programdel.

Eksempler:

```
DIM NAVN$ OF 20, HELTALSTABEL#(3,8)
DIM NAVNE$(2:4,10) OF 20
DIM REEL TABEL(10:20, 10:15)
```

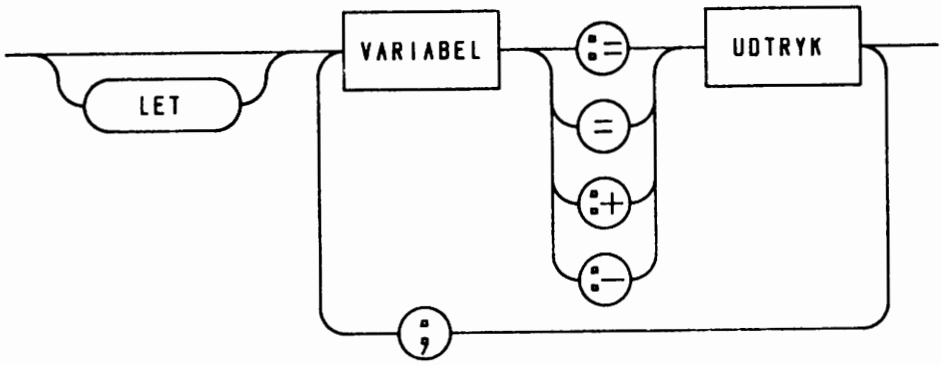
Se iøvrigt afsnit 4.5.3 om indicerede variable.

5.4 TILDELINGER

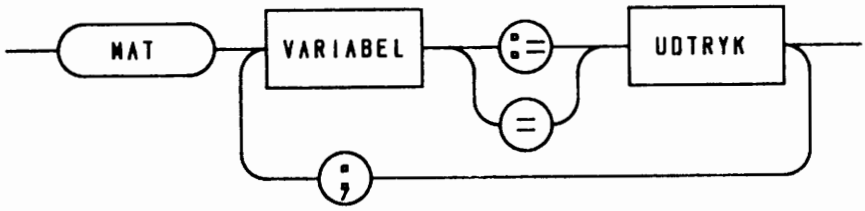
Tildeling af værdier til variable foretages i tildelingssætninger. Der findes 2 forskellige tildelingssætninger, nemlig LET- og MAT-sætningen.

Syntaks

LET-sætningen har følgende opbygning:



MAT-sætningen har følgende opbygning:



Udførelse

For hver tildeling i en LET-sætning udføres følgende:

a. Adressen på variablen på venstre side af ':=' beregnes.

b. Udtrykket beregnes.

c. Variablen på venstre side af ':=' tildeles resultatet af trin b.

'=' og ':=' er ækvivalente. I udskrifter konverteres sådanne lighedstegn til ':=', og LET udelades.

'variabel' :+ 'udtryk'

er ækvivalent med 'variabel' := 'variabel' + 'udtryk'

og 'variabel' :- 'udtryk' er ækvivalent med 'variabel' := 'variabel' - 'udtryk'

Denne form for minus kan ikke bruges for strengvariable.

Eksempel

Tildelingerne

```
TÆLLER#:= TÆLLER#+ 1
FORNAVN$ := FORNAVN$ + "-ERIK"
```

kan kort skrives

```
TÆLLER# + 1
FORNAVN$ :+ "-ERIK"
```

I MAT-sætningen tildeles værdien af 'udtryk' til alle elementer i 'variabel', som skal være et talsæt eller et sæt af tegnstrenge.

Også i MAT-sætningen er ':=' og '=' ækvivalente.

Kommentarer

1. Typen for 'variabel' og 'udtryk' skal stemme overens. Hermed menes, at kun kombinationer, som i tabellen herunder er markeret med et '+', er tilladte.

Type for 'variabel' \ Type for 'udtryk'	Heltal	Reel	Streng
Heltal	+	-	-
Reel	+	+	-
Streng	-	-	+

2. For tildelinger til strengvariable gælder følgende regler:

   a. Hvis 'udtryk' er længere end 'variabel', vil 'udtryk' blive afkortet, idet kun den første del bliver benyttet.

   b. Hvis 'udtryk' er kortere end 'variabel', vil 'udtryk' blive venstrestillet i 'variabel'.

3. For tildelinger til delstrenge gælder, at 'delstreng' og 'udtryk' skal have samme længde.

Eksempler:

```
LET TÆLLER#:+1; SUM=0
MAT TALTABEL#:=0; NAVNE$ := ""
```

5.5 LÆSNING OG SKRIVNING I COMAL-80

5.5.1 INDLEDNING

Læsning of skrivning i forbindelse med COMET'en kan foregå på en række forskellige måder, der hver især har sit fortrin.

Dels er der mange valgmuligheder, og dels kan visse nøgleord optræde i flere forskellige sammenhænge. Det er derfor nødvendigt med en samlet oversigt:

a. Mulighederne optræder typisk i par, således at der til en bestemt form for læsning svarer en bestemt form for skrivning.

b. Mulighederne kan yderligere opdeles i 4 grupper:

   1. "Læsning" og "skrivning" internt i maskinen.
   2. Til og fra en terminal (dataskærm eller evt. printer).
   3. Til og fra brugerfiler på baggrundslager.
   4. Læsning eller skrivning af hele programmer på baggrundslager.

   Bemærk, at de ting, der har med baggrundslagre at gøre, vil blive nøjere behandlet i afsnit 7, Filsystem, samt kapitel 8, Systemkommandoer. Maskinsprogsfunktioner vil blive behandlet i afsnit 6.5.

Tabel 5-1 - Læsning og skrivning i forbindelse med COMET'en

GRUPPE	NØGLEORD SKRIVNING	NØGLEORD LÆSNING	TYPE	KOMMENTARER	BEHANDLES I AFSNIT NR.
INTERNT	POKE	PEEK		MASKIN-SPROGS-FUNKTIONER	6.5
	DATA	READ		INTERNT I PROGRAM	5.5.2
VIA TERMI-NAL	PRINT USING	INPUT	ASCII	1) BRUG AF KOMMA OG SEMIKOLON ER ENS FOR PRINT OG INPUT	5.5.3 5.5.5
	PRINT				5.5.4
BRUGER FILER I PRO-GRAM-MER	PRINTFILE USING	INPUT FILE		2) FORMAT VED FIL-BRUG ER HELT ANALOGT MED FORMAT VED TERMINAL-BRUG	7.11 7.12
	PRINTFILE				7.11
	WRITE FILE	READ FILE	BINÆRT	PLADSBESPA-RENDE	7.13 7.14
FILER MED HELE PRO-GRAM-MER	LIST	ENTER	ASCII	LÆSNING OG SKRIVNING AF HELE PROGRAMMER PÅ BAG-GRUNDLAGER	8.3
	SAVE	LOAD	BINÆRT		

Bemærk, at ved åbning af en brugerfil skal man bruge en OPEN-sætning afsluttet af et af nøgleordene, READ, WRITE, APPEND eller RANDOM.

Disse nøgleord bruges også i helt andre forbindelser, nemlig h.h.v.:

READ-sætninger, READ-FILE-sætninger, WRITE FILE-sætninger samt RANDOM-sætninger eller RANDOM-kommandoer.

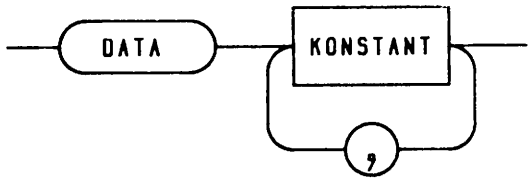
5.5.2 DATA/READ

5.5.2.1 DATA-SÆTNING

DATA-sætninger benyttes til at definere værdier i datalisten. Disse værdier kan indlæses ved hjælp af READ-sætninger (ikke READ FILE).

Syntaks

DATA-sætningen har følgende opbygning:



Udførelse

Ved starten af programmets udførelse, d.v.s. umiddelbart efter indtastning af RUN-kommandoen, gennemgås programmet i linienummerorden, og alle DATA-sætninger kædes sammen i en dataliste. Datalisten styres ved hjælp af en "pegepind", som altid udpeger den næste værdi, som skal tildeles. Denne "pegepind" bliver i starten sat til at udpege datalistens første værdi.

Under selve programudførelsen ignoreres DATA-sætningerne.

Kommentarer

1. Konstanter skal have den form, som er beskrevet i afsnit 4.2 og 4.3

2. Med RESTORE kan en bestemt DATA-sætning udpeges (se afsnit 5.6).

3. Systemvariablen EOD() er 'sand', når sidste DATA-værdi er læst (se afsnit 6.4.4).

Eksempel

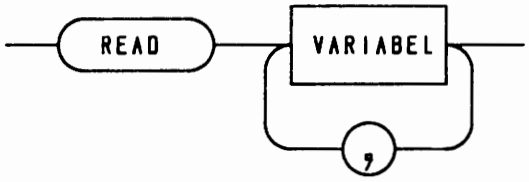
Se næste side.

5.5.2.2 READ-SÆTNING

READ-sætningen benyttes til at tildele startværdier til variable.

Syntaks

READ-sætningen har følgende opbygning:



Udførelse

Ved udførelsen af READ-sætningen vil variablerne i variabellisten få tildelt værdier i rækkefølge fra datalisten. Opbygningen af datalisten er beskrevet i forrige afsnit.

Kommentarer

1. Hvis typen af konstanten i datasætningen ikke stemmer overens med den angivne variabels type, eller hvis det sidste dataelement allerede er læst, stoppes programudføelsen med en fejludskrift.

2. Tildeling af værdier til strengvariable ved hjælp af READ-sætningen følger de samme regler, som er beskrevet i afsnit 5.4.

Eksempel

```
0010 DIM FORNAVN$ OF 10, EFTERNAVN$ OF 10
0020 DATA "HANS", "OLSEN", 10
0030 READ FORNAVN$, EFTERNAVN$
0040 PRINT FORNAVN$+" "+EFTERNAVN$
0050 READ ALDER
0060 PRINT ALDER:" ÅR"
0070 END
```

Køres ovenstående program, fås følgende resultat:

```
HANS OLSEN
10 ÅR
```

5.5.3 PRINT USING

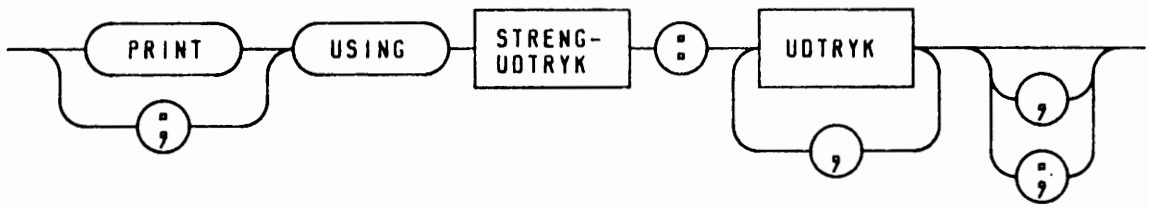
PRINT-sætninger bruges til at udskrive resultater i form af tal eller tegnstrenge.

PRINT-USING er behandlet særskilt, fordi dens formatteringsmåde er forskellig fra den, der gælder for egentlige PRINT-sætninger (og INPUT-sætninger).

PRINT-USING og PRINT kan bruges henholdsvis efter helt analoge regler i forbindelse med filer (se afsnit 7.11).

Syntaks

PRINT-USING-sætningen har følgende opbygning:



Udførelse

Med strengudtrykket angiver man det format, som man ønsker de efterfølgende udtryk udskrevet på. Tegnene i dette udtryk fortolkes på følgende måde:

'#' : cifferposition og fortegn
'.' : decimalpunkt (kun hvis omgivet af '#')
'+' : foranstillet plus (kun hvis '#' følger umiddelbart efter)
'-' : foranstillet minus (kun hvis '#' følger umiddelbart efter)

Alle andre tegn overføres direkte til udskriften. Hvis tallet er for stort til at være indenfor det opgivne format, udfyldes formatet med '*'-er. Hvis tallet har flere decimaler end angivet i udskrivningsformatet, foretages en afrunding inden udskrivningen.

Begynder et format med et '+', betyder det, at der afsættes en position til tallets fortegn, og at fortegn udskrives både for positive og negative tal.

Begynder et format med et '-', betyder det, at der afsættes en position til tallets fortegn, og at fortegn kun udskrives for

negative tal.

Et numerisk udskrivningsformat kan også benyttes til udskrivning af tegnstrenge. I dette tilfælde angiver hvert tegn i formatet en position i tegnstrengens udskrivningsformat. Ved udskrivning venstrestilles tegnstrengen i formatet. Er tegnstrengen for lang, afskæres et antal tegn i strengens højre ende. Er tegnstrengen for kort til at udfylde formatet, efterstilles et antal blanktegn.

PRINT USING-sætningen udføres ved, at formatstrengen gennemløbes tegn for tegn fra venstre mod højre. Alle tegn, som ikke indgår i et udskrivningsformat, udskrives direkte på skærm eller linieskriver. Hver gang et udskrivningsformat mødes, konverteres og udskrives det næste 'udtryk' i udtrykslisten i overensstemmelse med det opgivne format. Hvis der ikke er flere 'udtryk' i udtrykslisten, standses udførelsen af PRINT USING-sætningen, d.v.s. resten af formatstrengen ignoreres. Hvis der er flere 'udtryk' i udtrykslisten, end der er formater i formatstrengen, begyndes der forfra på formatstrengen.

Afsluttes PRINT USING-sætningen af et ',', vil næste udskrivning fortsætte i næste position (hvilket normalt er på den aktuelle linie). I modsat fald skiftes til ny linie inden næste udskrivning.

Med PRINT USING-sætningen bestemmer man altså også startpositionen for den efterfølgende PRINT- eller PRINT USING-sætning for samme terminal.

Ligeledes vil en PRINT-sætning bestemme startpositionen for den efterfølgende PRINT USING- eller PRINT-sætning for terminalen.

Dette vil da ske efter de regler, der gælder for tabuleringsangivelser i en egentlig PRINT-sætning (d.v.s. komma, semikolon og TAB(n)).

Eksempler

```
0010 // Eksempel på PRINT USING
0020 // * * *  sinustabel * * *
0030 PRINT USING "#inustabel": "Sådan er der så meget"
0040 FOR I:=0 TO 360 STEP 30 DO
0050   R:=I*3.141592/180
0060   PRINT USING "Sinus til ### gr. (#.### rad.) ": I,R;
0070   PRINT USING "er = -#.######": SIN(R)
0080 NEXT I
0090 END
```

```
0010 // Eksempel på PRINT USING
0020 SELECT OUTPUT "LP:"
0030 PRINT USING "-##.#": -3,44.47,1000,"ABC";
0040 PRINT " Slut!"
0050 END
```

```
-3.0 44.5*****ABC  Slut!
```

```
0010 // Dette program viser hvorledes man (ligesom i FORTRAN)
0020 // kan vælge udskrift-format ved reference til et tal.
0030 //
0040 DIM FORM$(3) OF 40
0050 DIM A(2)
0060 // Strengvektoren FORM$ tildeles formaterne
0070 FORM$(1):="##.### "
0080 FORM$(2):="#.### "
0090 FORM$(3):="#.###### - sådan cirka "
0100 //
0110 INPUT "Skriv et tal mellem 0 og 20 : ": A(1)
0120 INPUT "- og et til                 : ": A(2)
0130 INPUT "Format (1, 2 eller 3)       : ": I
0140 // værdien af A(1) og A(2) skrives ud på
0150 // et af de tre formater.
0160 PRINT USING FORM$(I): A(1),A(2)
0170 END
```

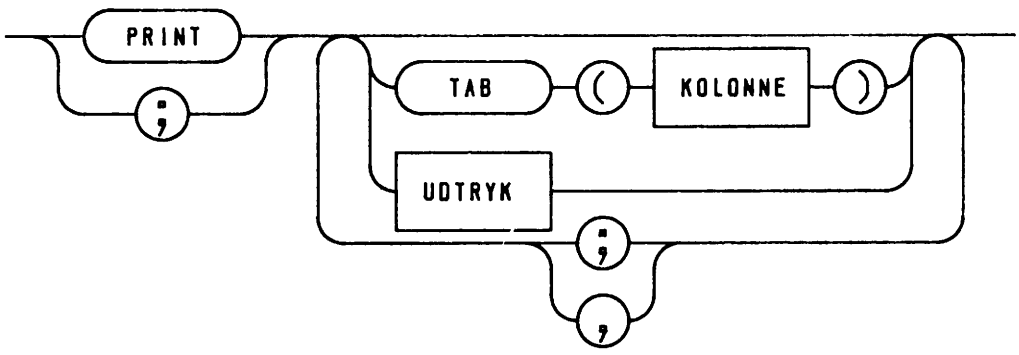
5.5.4 PRINT-SÆTNING

Med PRINT USING angiver man udskrivningsformatet på en meget direkte måde.

I egentlige PRINT-sætninger er formatet dels fastlagt ved standardregler og dels ved brugerens egne tabuleringsangivelser.

Syntaks

En PRINT-sætning har formen:



Som man ser, optræder der et semikolon 2 steder i diagrammet, nemlig et indledende, (1), og et fortsættende, (2). Det indledende semikolon kan i enhver henseende bruges i stedet for PRINT brugt som nøgleord. Medmindre andet er nævnt, vil den efterfølgende omtale af brugen af semikolon kun omhandle den anden type.

Kolonne, der angiver den kolonne, man tabulerer frem til, er et aritmetisk udtryk.

Som hovedregel vil en PRINT-sætning have følgende virkning:

1. Værdierne af 'udtryk' udskrives i rækkefølge.
2. TAB(n) tabulerer frem til kolonne n.
3. Komma tabulerer frem til start af næste zone.
4. Semikolon fastholder næste printposition, dog med mellemrum efter tal.

Disse 4 punkter kræver en nærmere omtale:

1. Udskrivning af 'udtryk'

NPP (Næste Print-Position

Systemet holder hele tiden styr på, hvad næste printposition er for hver terminal, der udskrives på (f.eks. dataskærm eller skriver). Lad os kalde en sådan printposition for NPP.

Printpositionerne er nummereret fra 1 og opefter (f.eks. til og med 80). Ved start af programmet er NPP = Første kolonne på næste linie. Hver gang en karakter er skrevet, rykker NPP automatisk 1 tak frem, hvilket eventuelt kan være til 1. kolonne på den følgende linie. Hvis det printelement, der er skrevet, er et tal, rykkes der yderligere en tak frem, medmindre andet er angivet med et efterfølgende komma, mere end 1 semikolon eller en TAB( )-angivelse.

Printelementer

Karakterværdierne af 'udtryk' betegnes: Printelementer. Printelementerne er altså de ASCII-tegnstrenge, som rent faktisk skal printes ud (dette gælder også for talværdier).

Hvis 'udtryk' er et strengudtryk, er printelementet = værdien af dette udtryk. Hvis 'udtryk' er et aritmetisk udtryk, så vil printelementet være de karakterer (inkl. cifre), der indgår i tallet efter følgende regler:

- Negative tal udskrives med foranstillet minus.
- Efterstillede nuller i decimaldel undertrykkes.
- Eventuelt 0 foran decimalpunkt udskrives.

For 7 (eller 13) cifres præcision gælder iøvrigt:

- Tal i området 1≤ X ≤9999999(999999) udskrives på decimalform uden eksponent og med maksimalt 7 (13) cifre. Hvis et sådant tal kun har nuller efter decimalpunktet, udskrives det som heltal uden decimalpunkt.

- Tal i området 0< X <1E-07 (1E-013) eller X >=1E+07 (1E+013) udskrives på eksponentform med 7 (13) cifre i mantissen, heraf 1 ciffer foran decimalpunkt, samt 4 (5) karakterer til eksponentangivelse.

Hvis 'udtryk' kun kan fortolkes som et logisk udtryk, vil printelementet være enten "0" eller "1".

Udskrivning af printelementer

Hvis der er plads til printelementet på resten af linien, så udskrives det startende i NPP.

I modsat fald starter udskrivningen i første kolonne på første frie linie (hvilket eventuelt kan være den aktuelle linie). Hvis printelementet yderligere er længere end en hel linie, vil det blive udskrevet på et antal hele linier, eventuelt efterfulgt af en delvist fyldt linie.

2. TAB(n)

TAB(n) tabulerer frem til den N'te kolonne på samme linie. Det er et krav, at NPP ≤ n ≤ linielængde. Den nye værdi af NPP vil da blive sat = n.

3. Effekten af kommaer

Hver gang, der optræder et komme, vil systemet undersøge, hvilken printzone NPP ligger i. Derefter vil det rykke NPP frem til starten af næste zone. Zonerne er bestemt ved zonebredden og linielængden. Hvis f.eks. zonebredden er 30, og linielængden er 80, vil zonerne i rækkefølge være:

kol. 1-30 inkl. i linie 1
kol. 31-60 inkl. i linie 1
kol. 61-80 inkl. i linie 1
kol. 1-30 inkl. i linie 2
kol. 31-60 inkl. i linie 2
o.s.v.

Zonebredden fastlægges ved den afrundede værdi af systemvariablen, ZONE, der ikke har noget at gøre med tabuleringsfunktionen TAB( ).

Ved indlæsning af COMAL-80 starter ZONE med værdien, nul. ZONE kan tildeles et aritmetisk udtryk, hvis afrundede værdi er ≥ 0.

TAB beholder sin værdi. indtil den tildeles en ny værdi. Tildelingen kan ikke ske midt i en PRINT-sætning, men må foretages i en tidligere tildelings-sætning eller -kommando.

Et komma med ZONE = 0 bevirker, at NPP fastholdes.

4. Brugen af semikolon

Semikolon fastholder generelt NPP, således at to på hinanden følgende printelementer kan skrives i forlængelse af hinanden, hvis det kan gøres inden for linielængden. Dette gælder også, hvis en printsætning afsluttes med semikolon, og der på et senere tidspunkt bliver printet pøny. Den eneste undtagelse er et taludtryk, der er direkte efterfulgt af semikolon. I dette tilfælde vil der blive sat et mellemrum, før en eventuel sammenhægtning med et efterfølgende printelement foretages.

Eksempler:

```
0010 // PRINT-eksempel 1
0020 SELECT OUTPUT "LP:"
0030 DIM A$ OF 3
0040 A$:="ABC"
0050 B:=1.234567E+15
0060 PRINT A$
0070 PRINT B
0080 PRINT "A"
0090 PRINT 67
0100 PRINT "A"="A"
```

Programmet giver følgende udskrift:

```
ABC
1.234567E+15
A
67
1
```

```
0010 // PRINT eksempel 2
0020 SELECT OUTPUT "LP:"
0030 A:=3.5
0040 PRINT "12345678901234567890123456789012345678901234567890"
0050 PRINT TAB(2);A;TAB(8);56;TAB(11);"A"
0060 END
```

Programmet giver følgende udskrift:

```
12345678901234567890123456789012345678901234567890
 3.5   56 A
```

```
0010 // PRINT eksempel 3
0020 // brugen af komma.
0030 SELECT OUTPUT "LP:"
0040 PRINT "12345678901234567890123456789012345678901234567890"
0050 FOR I:=0 TO 4 DO
0060   ZONE:=I // Zone-bredden defineres
0070   FOR J:=1 TO 5 DO
0080     PRINT "AAA", // Læg mærke til kommaets virkning
0090   NEXT J
0100   PRINT TAB(32);I
0110 NEXT I
0120 END
```

Programmet giver følgende udskrift:

```
12345678901234567890123456789012345678901234567890
AAAAAAAAAAAAAAA                 0
AAA AAA AAA AAA AAA             1
AAA AAA AAA AAA AAA             2
AAA   AAA   AAA   AAA   AAA     3
AAA AAA AAA AAA AAA             4
```

```
0010 // PRINT eksempel 4
0020 // Brug af semikolon.
0030 SELECT OUTPUT "LP:"
0040 PRINT "A";"B";"C";"D";
0050 PRINT 1;2;3;4;5;6
0060 PRINT "A";1;"2";3;"4";5;;;;"6"
0070 END
```

Programmet giver følgende udskrift:

```
ABCD1 2 3 4 5 6
A1 23 45 6
```

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

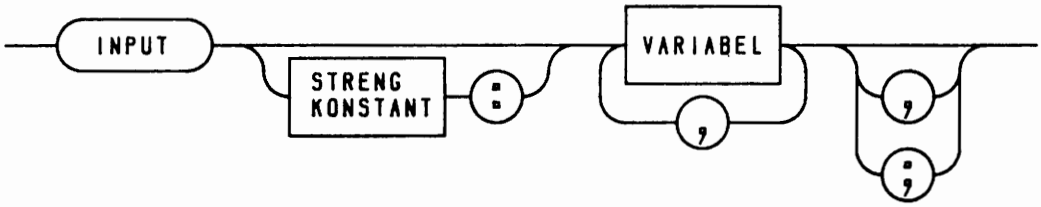
Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

5.5.5 INPUT-SÆTNING

INPUT-sætningen benyttes til at indlæse værdier fra terminalen og tildele disse til variable.

Syntaks

INPUT-sætningen har følgende opbygning:



Udførelse

Hvis INPUT-sætningen indeholder en 'strengkonstant', udskrives denne på dataskærmen; ellers udskrives tegnene '? ' (spørgsmålstegn mellemrum) for at indikere, at systemet forventer inddata.

Brugeren indtaster herefter det antal værdier, som skal tildeles variablerne i INPUT-sætningen. Indtastningen afsluttes ved, at man trykker på 'RETURN'-tasten.

Listen af variable i INPUT-sætningen gennemløbes fra venstre mod højre, og de tilsvarende indtastede værdier tildeles.

Hvis variabellisten afsluttes med et ',' eller et ';' gælder tilsvarende regler som for PRINT-sætninger. INPUT- og PRINT-(USING)-sætninger kan bruges i flæng efter de tabuleringsregler, som gælder for sætningerne hver især.

Kommentarer

1. Talkonstanter indtastes på den form, som er beskrevet i afsnit 4.2. Flere talkonstanter adskilles med en karakter, som ikke kan indgå i et tal, f.eks. et blanktegn.

2. En strengkonstant indtastes blot som en række af ASCII-tegn. Det er ikke muligt at indtaste værdier på inddatalinien efter en strengkonstant, idet resten af linien vil blive opfattet som værende en del af strengen. En strengkonstant, som indtastes efter en talkonstant, starter efter det første tegn, som ikke kan indgå i tallet.

Eksempel

```
0010 INPUT "Styktal ": STYKTAL#
0020 INPUT "Stykpris ": STYKPRIS
0030 PRINT "Totalpris:";STYKTAL#*STYKPRIS
0040 END
```

Køres ovenstående program, fås følgende skærmdialog, hvor brugersvar er understreget.

```
STYKTAL 10
STYKPRIS 33.5
TOTALPRIS:335
```

Programmet:

```
0010 ZONE:=10
0020 PRINT "12345678901234567890123456789012345678901234567890"
0030 INPUT A;
0040 PRINT A;
0050 INPUT B, C,
0060 ZONE:=0
0070 INPUT D,
0080 INPUT E
0090 END
```

giver ved kørsel følgende skærmdialog, hvor brugersvar er understreget:

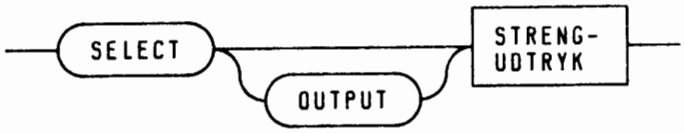
```
12345678901234567890123456789012345678901234567890
? 12.812.8 ? -5 ? 456          ? 7? 23
```

5.5.6 SELECT (OUTPUT)

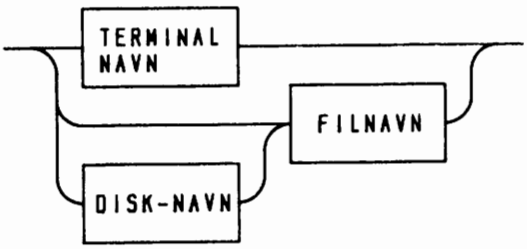
Som navnet antyder, bruges denne sætning til at ændre udskrivningsmedium for PRINT (USING) sætninger.

Syntaks

SELECT-sætningen har følgende format:



STRENGUDTRYK:



Fra starten af kørslen (RUN) er det dataskærmen, der er standard-udskrivningsmediet. Det SELECT'ede medium bruges, indtil et nyt medium er udvalgt. Hvis man har angivet et filnavn uden at angive enheden, vil systemet oprette filen på standard-baggrundsenheden. Dette er enten den med lavest nummer eller den, der sidst er angivet med UNIT.

Eksempler

```
SELECT OUTPUT "LP:" //              Printer
SELECT OUTPUT "DK0:MINFIL" //      Fil
SELECT OUTPUT "DINFIL" //          Fil
SELECT OUTPUT "DS:" //             Dataskærm
```

5.5.7 SKÆRMKONTROL (CURSOR OG CLEAR)

5.5.7.1 INDLEDNING

I dette afsnit beskrives 2 sætninger, som kan benyttes til henholdsvis at kontrollere markørens placering på skærmen og til at slette dataskærmens indhold.

5.5.7.2 CURSOR-SÆTNING

CURSOR-sætningen benyttes til at styre markørens placering på dataskærmen.

Syntaks

CURSOR-sætningen har følgende opbygning:

X-KOORDINAT,
Y-KOORDINAT:

Udførelse

X-koordinat og Y-koordinat angiver koordinaterne for en position på dataskærmen. Ved udførelse af CURSOR-sætningen flyttes markøren til den angivne position.

X-aksen regnes positiv fra venstre mod højre, og dens koordinater må ligge i intervallet: 1 <= X <= 80

Y-aksen regnes positiv fra oven og nedefter, og dens koordinater må ligge i intervallet: 1 <= Y <= 24

Skærmens øverste venstre hjørne har således koordinaterne (1,1), og nederste højre hjørne har koordinaterne (80.24)

Kommentarer

1. Hvis beregningen af X- eller Y-koordinaten ikke giver et helt tal, foretages en afrunding internt i maskinen ifølge ROUND-funktionen.

Eksempel Se under CLEAR (næste afsnit).

5.5.7.3 CLEAR-SÆTNING

CLEAR-sætningen benyttes til at slette dataskærmens aktuelle indhold

Syntaks

CLEAR-sætningen har følgende opbygning:

Udførelse

Udførelse af CLEAR-sætningen bevirker, at dataskærmens indhold slettes, hvorefter markøren flyttes til øverste venstre hjørne af dataskærmen.

Eksempel

```
0010 // Eksempel på brug af CURSOR og CLEAR.
0020 RANDOM
0030 FOR I:=1 TO 50 DO
0040   CLEAR
0050   CURSOR RND(1,60), RND(1,24)
0060   PRINT "Kilroy was here";
0070   FOR J:=1 TO 200 DO // Vent
0080   NEXT J
0090 NEXT I
0100 END
```

5.5.8 BRUG AF PRINTER

Normalt vil en printer ved opstart skrive med 10 tegn pr. tomme (10 CPI) og 6 linier pr. tomme. De fleste (især større) printere kan dog tillige skrive med en lang række skrifttyper som f.eks.:

Tegntæthed	Tegntæthed angives i tegn pr. tomme (Characters Per Inch): 5 CPI 6 CPI 8.5 CPI 10 CPI 12 CPI 17 CPI Proportional-skrift: Hvert tegn fylder tegnets reelle bredde ('i' fylder f.eks. mindre end 'm')
Linietæthed	6 LPI 8 LPI
Skrifttype	Almindelig skrift Fremhævet skrift Korrespondance kvalitet (tegnet sammensættes af flere prikker (dots)) Understreget skrift Subscript/superscript som f.eks. i x^2, H_2O Grafik

Da forskellige lande har afvigelser i tegnsættet i forhold til ASCII-tegnsættet (American Standard Code for Information Interchange), har de fleste printere mulighed for at vælge nationale tegnsæt (i Danmark drejer det sig f.eks. om: æ, ø, å, Æ, Ø, Å.

Da udskriftsmulighederne varierer fra printerfabrikat til printerfabrikat (og fra model til model), vil det her være uoverkommeligt at give en udtømmende beskrivelse af de printere, der i dag anvendes sammen med COMET'en. Vi vil derfor kun her beskæftige os med principperne bag printerstyring ved hjælp af kontrolkoder og i øvrigt henvise til den manual, der følger med ved købet af en printer. De eksempler, der vil blive anvendt i dette afsnit, vil være tilpasset OKI Microline 93 printeren.

For at få printeren til at vælge en bestemt skrifttype, skal der sendes en eller flere kontrolkoder (ASCII-værdi 0-31) for hver funktion, der vælges. Disse koder angives i forbindelse med PRINT-sætninger. Hvis man ønsker at sende ASCII-kode 31 til printeren, (hvilket får OKI 93'eren til at skrive med brede bogstaver) kan dette i COMAL 80 gøres på følgende 2 måder:

```
PRINT CHR$(31);      eller

PRINT ""31"";
```

Bemærk semikolon efter print-ordren, hvilket medfører, at der ikke sker linieskift.

EKSEMPEL

```
0010 DIM CPI16$ OF 1, CPI10$ OF 1, BRED$ OF 1
0020 CPI16$:=""29""; CPI10$:=""30""; BRED$:=""31""
0030 SELECT OUTPUT "LP:"
0040 PRINT CPI10$;"Printerstyring med kontrol-koder:"
0050 PRINT CPI16$;"16.5 tegn pr. tomme.";BRED$;" Dobbelt bredde."
0060 PRINT CPI10$;"10 tegn pr. tomme.";BRED$;" Dobbelt bredde."
0070 END
```

Programmet giver følgende udskrift:

```
Printerstyring med kontrol-koder:
16.5 tegn pr. tomme. Dobbelt bredde.
10 tegn pr. tomme. Dobbelt bredde.
```

5.5.9 PAGE

Hvis en printer er tilsluttet, vil udførelse af PAGE-sætningen medføre sideskift på printeren. Længden (antallet af linier) på papiret defineres ved hjælp af PAGELENGTH variablen. Ved opstart er denne sat til 72 (standard A4 'stående' format).

Syntaks

Eksempel

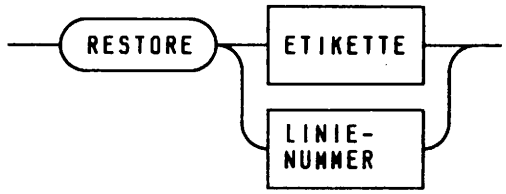
```
PAGE
```

5.6 RESTORE-SÆTNING

RESTORE-sætningen benyttes til at flytte "pegepinden" i datalisten, som er omtalt i afsnit 5.2.

Syntaks

RESTORE-sætningen har følgende opbygning:



Udførelse

Udførelse af RESTORE-sætningen uden etikette eller linienummer-angivelse bevirker, at "pegepinden" flyttes til datalistens første værdi.

Efterfølges RESTORE af et linienummer, skal den refererede linie indeholde en DATA-sætning. Udførelse af en sådan RESTORE-sætning bevirker, at "pegepinden" flyttes til den første værdi i den angivne DATA-sætning.

En etikette refererer til en linie, som er blevet navngivet i en LABEL-sætning (se afsnit 5.8). Efterfølges RESTORE af en etikette, skal linien, som følger umiddelbart efter den pågældende LABEL-sætning, være en DATA-sætning. Udførelse af en sådan RESTORE-sætning bevirker, at "pegepinden" flyttes til den første værdi i denne DATA-sætning.

Eksempel

```
0010 DIM LINIE$ OF 20
0020 DATA "linie 20"
0030 LABEL LINIE_30
0040 DATA "linie 40"
0050 READ LINIE$
0060 PRINT LINIE$
0070 READ LINIE$
0080 PRINT LINIE$
0090 RESTORE LINIE_30
0100 READ LINIE$
0110 PRINT LINIE$
0120 RESTORE 0020
0130 READ LINIE$
0140 PRINT LINIE$
0150 RESTORE
0160 READ LINIE$
0170 PRINT LINIE$
0180 END
```

Programmet giver følgende udskrift:

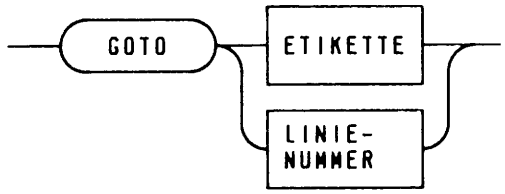
```
linie 20
linie 40
linie 40
linie 20
linie 20
```

5.7 GOTO-SÆTNING

GOTO-sætning benyttes til at bryde den normale sekventielle programudførelse for at fortsætte udførelsen i en angivet sætning.

Syntaks

GOTO-sætningen har følgende opbygningen:



Udførelse

Udførelse af en GOTO-sætning medfører, at programudførelsen fortsættes i den angivne programlinie. 'Etikette' er navnet, som en linie er blevet tildelt i en LABEL-sætning (se afsnit 5.8).

Eksempel

```
0010 DIM BELØB(3)
0020 I:=1
0030 LABEL LÆS_BELØB
0040 INPUT "Indtast beløb ": BELØB(I)
0050 I:+1
0060 IF I<=3 THEN GOTO LÆS_BELØB
0070 PRINT "3 beløb er nu indtastet"
0080 END
```

Udføres ovenstående program, fås følgende skærmdialog, hvor brugersvar er understreget:

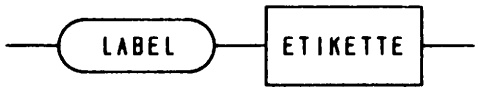
INDTAST BELØB 10.25
INDTAST BELØB 30
INDTAST BELØB 5.5
3 BELØB ER NU INDTASTET

5.8 LABEL-SÆTNING

LABEL-sætningen benyttes til at navngive en linie. Linien kan således refereres til ved sit navn i forbindelse med RESTORE- og GOTO-sætninger.

Syntaks

LABEL-sætningen har følgende opbygning:



ETIKETTE:

Udførelse

LABEL-sætningen ignoreres under selve programudførelsen.

Eksempel

```
0010 LABEL FORFRA
       .
       .
       .
0180 GOTO FORFRA
```

Se også eksemplet i afsnit 5.6.

5.9 BETINGEDE SÆTNINGER

5.9.1 INDLEDNING

COMAL-80 rummer muligheder for at gøre udførelsen af en række sætninger betinger af værdien af et logisk udtryk.

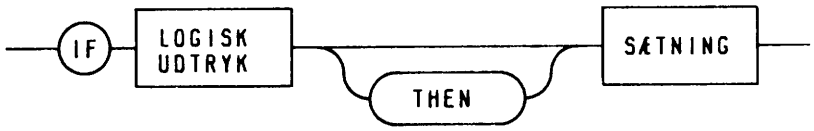
Der findes i COMAL-80 tre principielt forskellige konstruktioner for betingede sætninger:

1. Simple IF-sætninger
2. Sammensatte IF-sætninger
3. CASE-sætninger

Pkt. 2, Sammensatte IF-sætninger, kunne godt beskrives samlet, men er af pædagogiske grunde opdelt i hovedgrupper.

5.9.2 SIMPLE IF-SÆTNINGER

Syntaks



Udførelse

Er 'Logisk udtryk' sand, udføres 'sætning', ellers ignoreres 'sætning'.

Kommentarer

1. 'Sætning' må ikke indeholde:

   - en erklæring (DIM, PROC/ENDPROC, DEF/ENDDEF, LABEL)
   - en DATA-sætning
   - en del af en sammensat betinget sætning (ELIF, ELSE, ENDIF)
   - en del af en sammensat sætningskonstruktion (REPEAT/UNTIL, WHILE/ENDWHILE, CASE/WHEN/OTHERWISE/ENDCASE, FOR/NEXT, LOOP/ENDLOOP)

EXIT, der kan opfattes som en del af LOOP/ENDLOOP, må dog gerne bruges, og vil ofte blive brugt, i en simpel IF-sætning.

'Sætning' må gerne indeholde en ny Simpel IF-sætning.

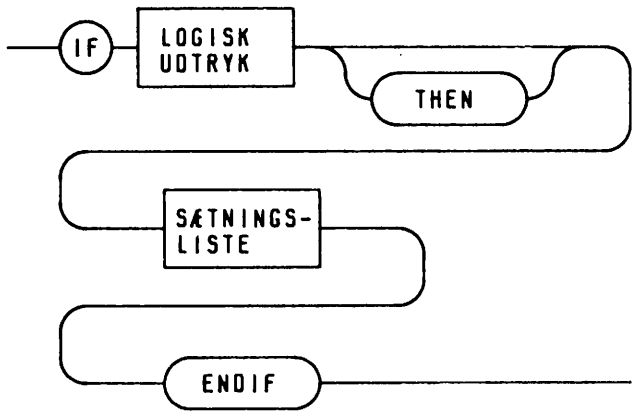
Eksempler

Følgende IF-THEN-sætninger er korrekte:

```
IF A <=100 THEN IF A >=90 PRINT "OK"
eller den ækvivalente
IF A <=100 AND A>=90 PRINT "OK"
IF TAL THEN PRINT "TAL <> 0="
```

5.9.3 SAMMENSATTE IF-SÆTNINGER

Syntaks



'Sætningsliste' betegner et vilkårligt antal COMAL-80 sætninger.

Udførelse

Er 'Logisk udtryk' sand, udføres sætningerne i 'sætningsliste'. Er 'Logisk udtryk' falsk, fortsætter programudførelsen i sætningen umiddelbart efter ENDIF-sætningen.

Kommentarer

1. Hvis en sætningsliste indeholder en eller flere sammensatte sætninger, d.v.s. sætninger, som strækker sig over flere linier, skal disse sætninger være helt indeholdt i sætningslisten.

Eksempler

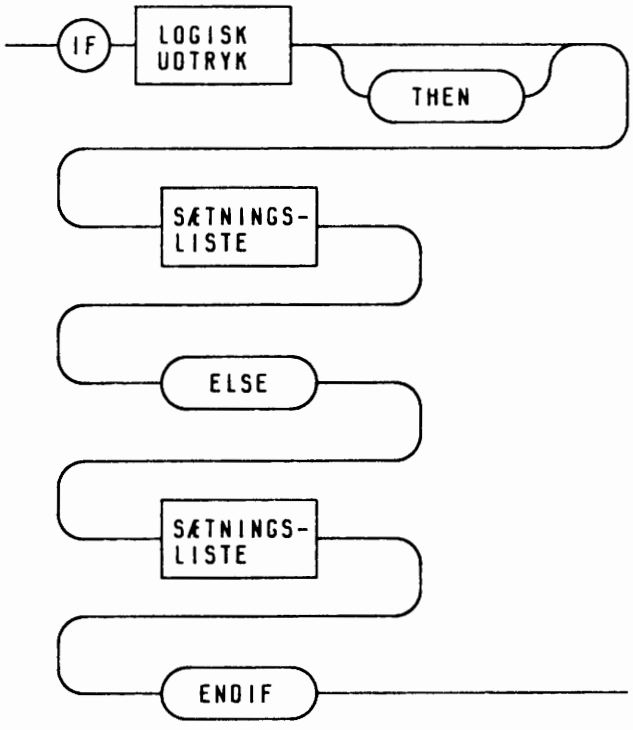
```
0010 TAL1:=3; TAL2:=4
0020 IF TAL1<>TAL2 THEN
0030   PRINT "TAL1 <> TAL2"
0040   PRINT "Summen er lig med ";TAL1+TAL2
0050 ENDIF // Undersøgelse slut
0060 END
```

Programmet giver følgende udskrift:

```
TAL1 <> TAL2
Summen er lig med 7
```

5.9.3.2 IF-ELSE-ENDIF

Syntaks



Udførelse

Er 'Logisk udtryk' sand, udføres sætningerne i sætningslisten efter THEN, hvorefter programudførelsen fortsætter med den første sætning efter den tilhørende ENDIF-sætning.

Er 'Logisk udtryk' falsk, udføres sætningerne i sætningslisten efter ELSE.

Kommentarer

1. Hvis en sætningsliste indeholder en eller flere sammensatte sætninger, d.v.s. sætninger som strækker sig over flere linier, skal disse sætninger være helt indeholdt i sætningslisten.

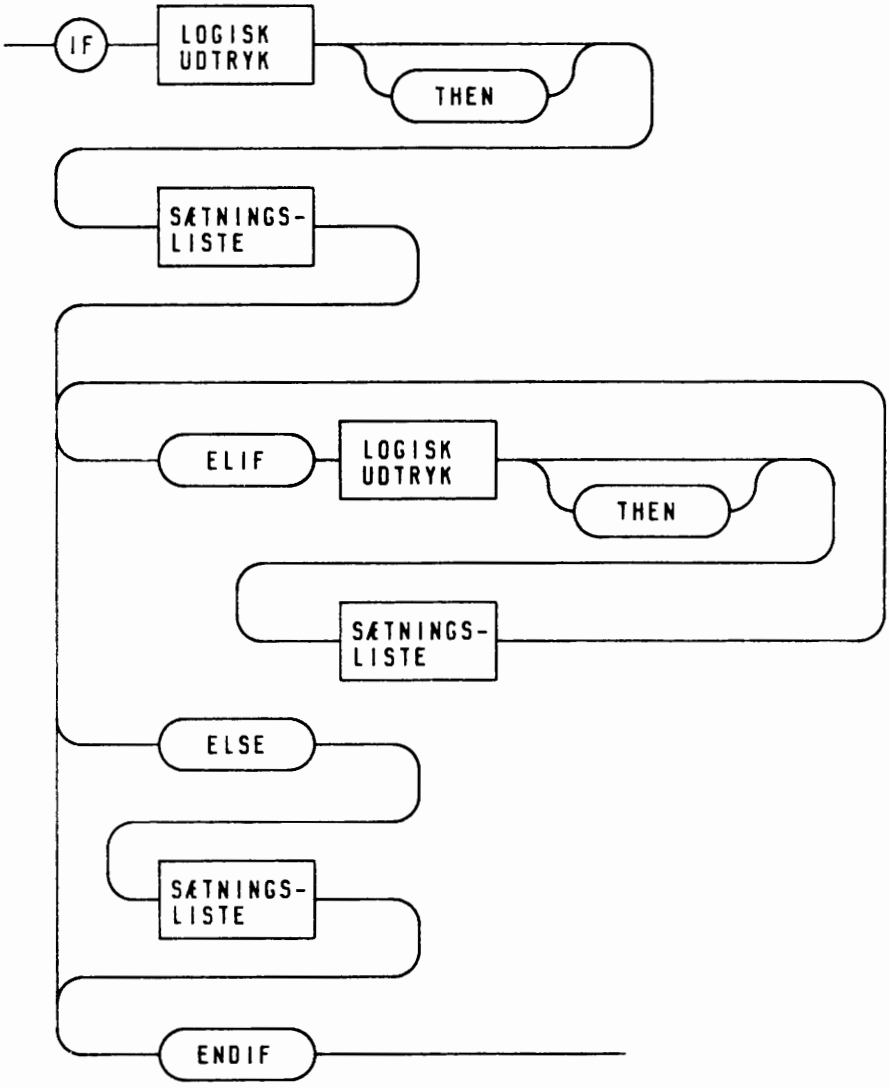
Eksempler

```
0010 INPUT "3+5 = ": SUM
0020 IF SUM=8 THEN
0030   PRINT "Rigtigt!"
0040 ELSE
0050   PRINT "Forkert!"
0060   PRINT "Det rigtige svar er 8."
0070 ENDIF
0080 END
```

5.9.3.3 IF-ELIF-(ELSE)-ENDIF

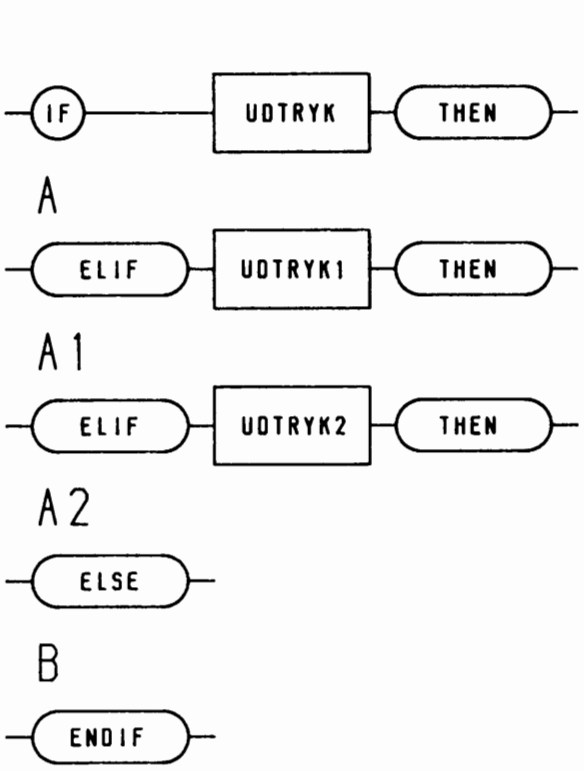
Nøgleordet ELIF er en forkortelse for ELSE IF.

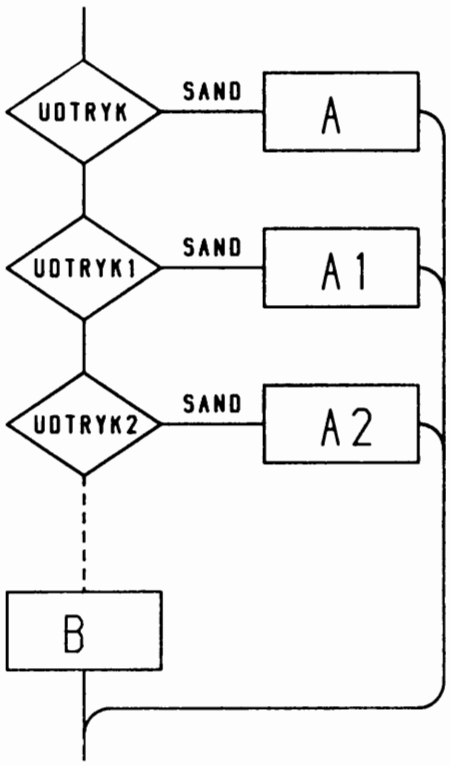
Syntaks

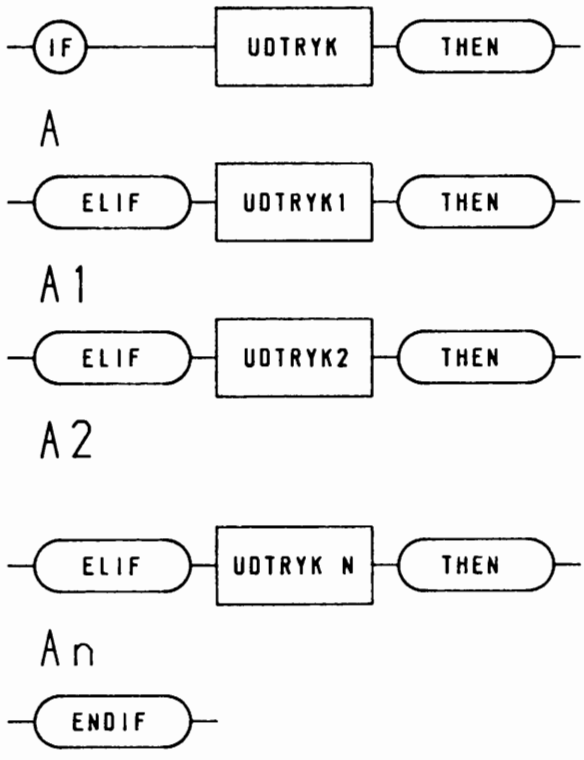


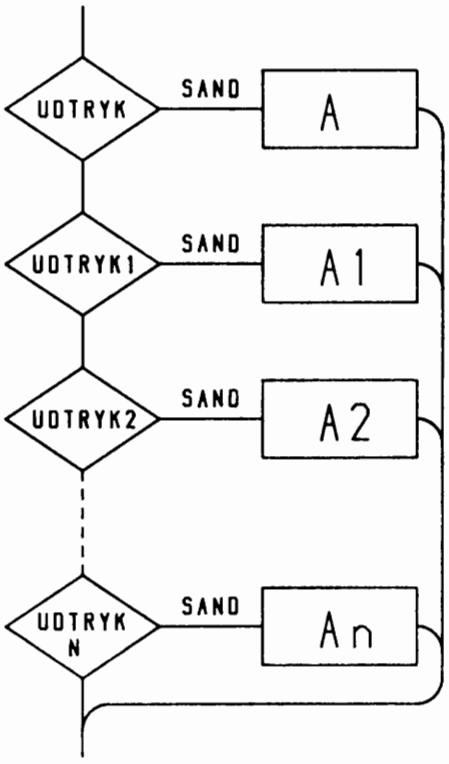
Udførelse

Udførelsen skitseres ved hjælp af følgende 2 rutediagrammer:
(se næste side)









International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

A, A1, A2, An og B betegner sætningslister. Læs specielt mærke til, at selv om flere af udtrykkene udregnes til værdien sand, vil kun sætningslisten efter det første af disse udtryk blive udført.

Kommentarer

1. Hvis en sætningsliste indeholder en eller flere sammensatte sætninger, d.v.s. sætninger, som strækker sig over flere linier, skal disse sætninger være helt indeholdt i sætningslisten.

Eksempler

```
0010 INPUT "Indtast et tal ": TAL
0020 IF TAL>1000 THEN
0030   PRINT "TAL > 1000"
0040 ELIF TAL<0 THEN
0050   PRINT "TAL < 0"
0060 ELIF TAL=0 THEN
0070   PRINT "TAL = 0"
0080 ELSE
0090   PRINT "0 < TAL <= 1000"
0100 ENDIF
0110 END
```

Køres ovenstående program, fås følgende skærmdialog, hvor brugersvar er understreget:

```
INDTAST ET TAL 10
0 < TAL <= 1000
```

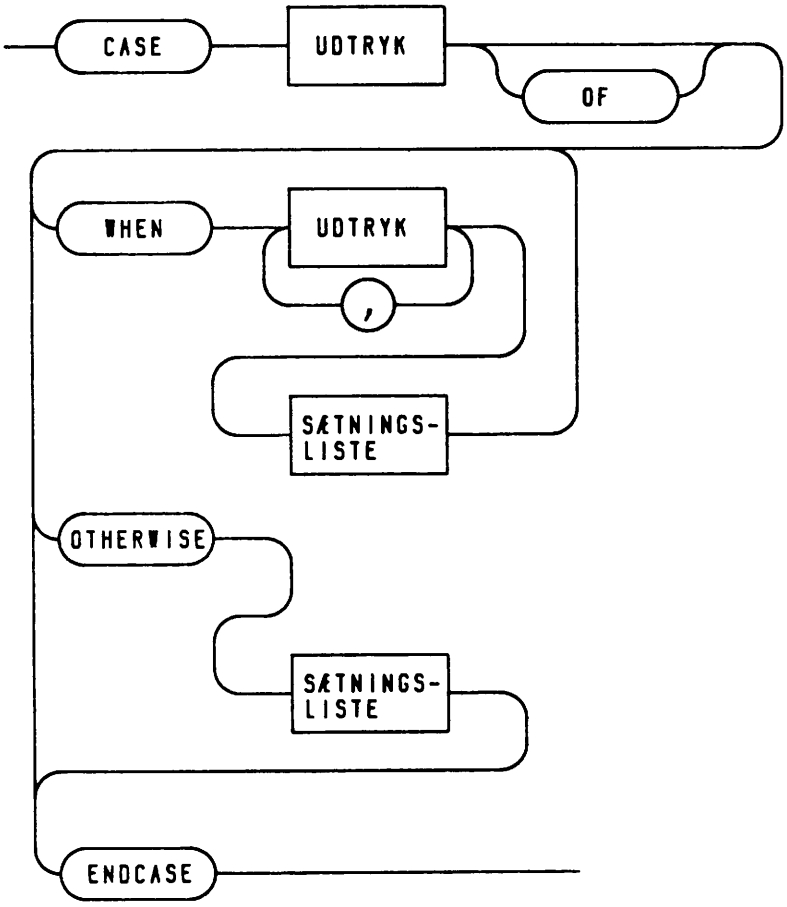
5.9.4 CASE-SÆTNING

CASE-sætningen benyttes, når een af flere sætningslister skal udføres afhængigt af værdien af et udtryk.

Syntaks

CASE-sætningen har følgende opbygning:

(se næste side)



'Sætningsliste' betegner et vilkårligt antal COMAL-80 sætninger.

Udførelse

'Udtryk' beregnes. Herefter gennemløbes udtrykkene i WHEN-sætningerne, indtil der findes en værdi, som er identisk med værdien af det beregnede udtryk. Hvis en sådan identisk værdi er fundet, udføres sætningslisten svarende til den pågældende WHEN-sætning, hvorefter udførelsen fortsætter efter ENDCASE.

Hvis der ikke blandt udtrykkene i WHEN-sætningerne findes en værdi, som er identisk med det beregnede udtryks værdi, udføres den alternative sætningsliste efter OTHERWISE-sætningen. Indeholder CASE-sætningen ingen alternativ sætningsliste, vil programudførelsen i ovennævnte tilfælde standse med en fejlmeddelelse.

Kommentarer

1. Udtrykkene i WHEN-sætningerne skal være af samme type som det beregnede udtryk.

2. Hvis en sætningsliste indeholder en eller flere sammensatte sætninger, d.v.s. sætninger som strækker sig over flere linier, skal disse sætninger være helt indeholdt i sætningslisten.

Eksempel

```
0010 DIM TAL$ OF 1
0020 INPUT "Indtast et ciffer ": TAL$
0030 //
0040 CASE TAL$ OF
0050 WHEN "0"
0060   PRINT "Tallet er et nul"
0070 WHEN "1", "2", "3", "5", "7"
0080   PRINT "Tallet er et primtal"
0090 WHEN "4", "8"
0100   PRINT "Tallet er en potens af 2"
0110 WHEN "6"
0120   PRINT "Tallet er et sekstal"
0130 WHEN CHR$(57)
0140   PRINT "Tallet er et nital"
0150 OTHERWISE
0160   PRINT "Jeg bad dig indtaste et CIFFER!"
0170 ENDCASE
0180 END
```

Herunder vises skærmdialogen ved 2 forskellige kørsler af ovenstående program. Brugersvar er understregede.

1. Indtast et ciffer A
   Jeg bad dig om at indtaste et ciffer!

2. Indtast et ciffer 4
   Tallet er en potens af 2

5.10 REPETERENDE SÆTNINGER

5.10.1 INDLEDNING

I COMAL-80 har man mulighed for at udføre en række sætninger et antal gange afhængigt af en stop-betingelse. Der findes 4 forskellige typer af repeterende sætninger:

FOR-NEXT

Hvor man angiver antallet af gange, en løkke skal gennemløbes.

WHILE-ENDWHILE

Hvor man i starten af en løkke angiver, om den skal gennemløbes.

REPEAT-UNTIL

Hvor man i slutningen af en løkke angiver, om man skal undlade atter at gennemløbe den.

LOOP-ENDLOOP

Hvor man inde i en stadigt gentaget løkke angiver, om man skal hoppe ud af den.

Man kan godt hoppe ud af en løkke med en GOTO-sætning. Under visse omstændigheder kan man med GOTO hoppe ind i det indre af en løkke, men dette må stærkt frarådes.

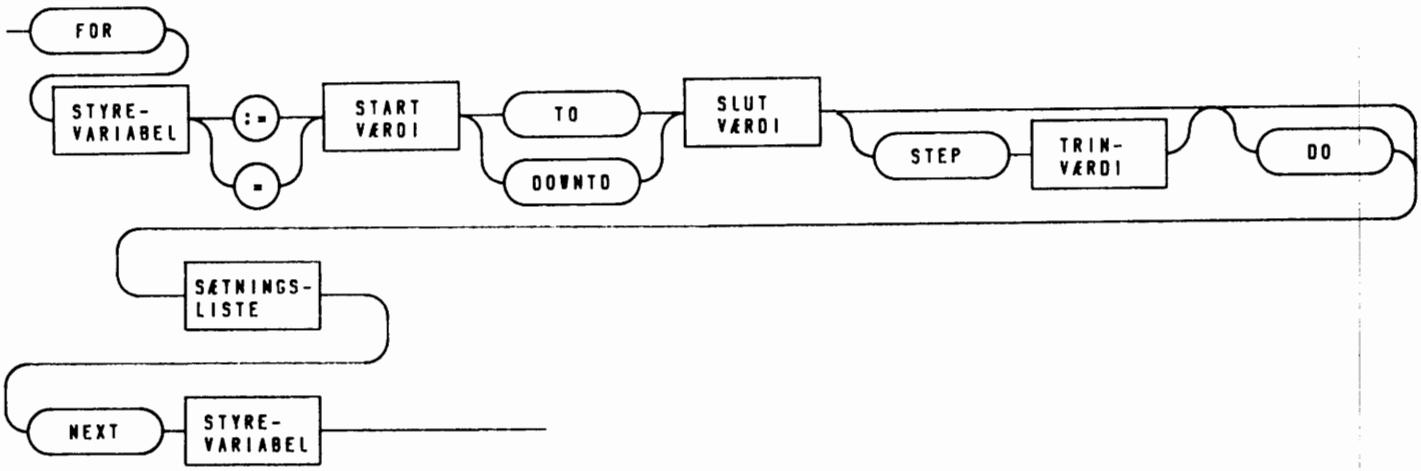
De 4 konstruktioner vil blive gennemgået i det følgende.

5.10.2 FOR-NEXT-SÆTNING

Syntaks

FOR-NEXT-sætningen har følgende opbygning:

(se næste side)



STYRE-VARIABEL:

START-VÆRDI, SLUT-VÆRDI, TRIN-VÆRDI:

'Sætningsliste' betegner et vilkårligt antal COMAL-80 sætninger.

Udførelse

a. Først beskrives udførelsen af en FOR-NEXT-sætning, hvor nøgleordet 'TO' er benyttet.

Er trinværdien ikke angivet, benyttes værdien 1 som trinværdi.

Ved udførelsen af en FOR-sætning vil styrevariablen få tildelt startværdien. Herefter undersøges, om de opgivne start-, slut- og trinværdier har nogen mening, d.v.s. om

$$(\text{slutværdi}-\text{startværdi})*\text{SGN}(\text{trinværdi})\geq 0$$

Er dette ikke tilfældet, overspringes FOR-NEXT-sætningen.

Ved udførelsen af den tilsvarende NEXT-sætning bliver trinværdien adderet til værdien af styrevariablen. Såfremt styrevariablens nye værdi ligger i området mellem start- og slutværdien inkl., fortsættes programudførelsen i den første sætning efter FOR-sætningen. I modsat fald fortsættes med sætningen efter NEXT-sætningen.

b. Er nøgleordet 'DOWNTO' benyttet i FOR-sætningen, udføres FOR-NEXT-sætningen fuldkommen som beskrevet i punkt a., men med en negeret trinværdi. D.v.s.

benyttet trinværdi = -trinværdi

Kommentarer

1. Der skal være overensstemmelse mellem styrevariablens type og typen af start-, slut- og trinværdi på følgende måde:

Styrevariablens type	Tilladt type for start-, slut- og trinværdi
heltal	heltal
reel	reel/heltal

2. Hvis sætningslisten indeholder en eller flere sammensatte sætninger, d.v.s. sætninger, som strækker sig over flere linier, skal disse sætninger afsluttes inden for sætningslisten.

Eksempler

```
0010 SUM:=0
0020 FOR TAL#:=1 TO 500 DO
0030   SUM:+TAL#
0040 NEXT TAL#
0050 PRINT "Sum (1..500) : ";SUM
0060 END
```

```
0010 SUM:=0
0020 FOR TAL#:=500 DOWNTO 1 DO
0030   SUM:+TAL#
0040 NEXT TAL#
0050 PRINT "Sum (500..1) : ";SUM
0060 END
```

Begge ovenstående programmer beregner og udskriver summen af de hele tal fra 1 til 500, begge inkl.

```
0010 Y:=3
0020 FOR X:=1.2 TO 4.7 STEP Y/2 DO
0030   PRINT X;
0040 NEXT X
0050 END
```

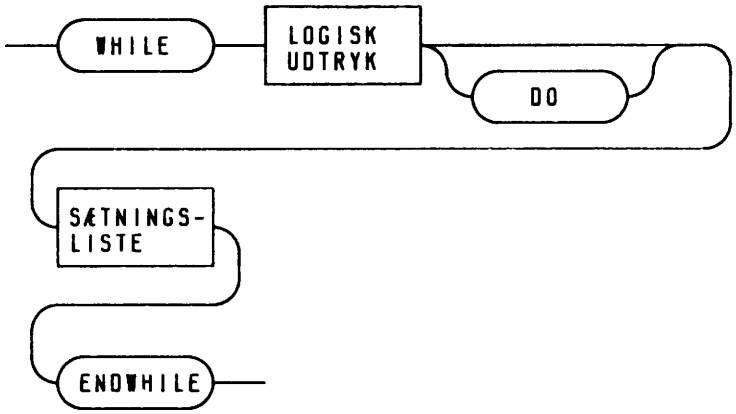
Ovenstående program giver følgende udskrift:

```
1.2 2.7 4.2
```

5.10.3 WHILE-ENDWHILE-SÆTNING

Syntaks

WHILE-ENDWHILE-sætningen har følgende opbygning:



'Sætningsliste' betegner et vilkårligt antal COMAL-80 sætninger.

Udførelse

Når en WHILE-sætning udføres, udregnes værdien af det logiske udtryk efter WHILE. Er udtrykkets værdi sand, fortsættes programudførelsen i den efterfølgende sætning. Er udtrykkets værdi falsk, fortsættes programudførelsen i sætningen umiddelbart efter den tilhørende ENDWHILE-sætning.

Udførelsen af en ENDWHILE-sætning bevirker blot, at programudførelsen fortsættes i den tilhørende WHILE-sætning.

Kommentarer

1. Hvis sætningslisten indeholder en eller flere sammensatte sætninger, d.v.s. sætninger, som strækker sig over flere linier, skal disse sætninger være helt indeholdt i sætningslisten.

Eksempel

```
0010 // Beregning af kvadratrødder
0020 X:=10
0030 DELTA:=X
0040 SQRT:=X/2
0050 // Iteration
0060 WHILE ABS(DELTA)>0.001 DO
0070   DELTA:=(X/SQRT-SQRT)/2
0080   SQRT:+DELTA
0090 ENDWHILE
0100 PRINT SQRT
0110 END
```

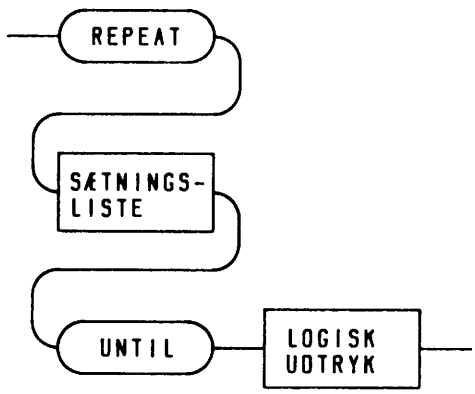
Programmet giver følgende udskrift:

```
3.162278
```

5.10.4 REPEAT-UNTIL-SÆTNING

Syntaks

REPEAT-UNTIL-sætningen har følgende opbygning:



'Sætningsliste betegner et vilkårligt antal COMAL-80 sætninger.

Udførelse

REPEAT-sætningen markerer blot starten på en REPEAT-UNTIL-sætning, og udførelsen af den har ingen effekt.

Når en UNTIL-sætning udføres, udregnes det logiske udtryk efter UNTIL. Er udtrykkets værdi sand, fortsætter programudførelsen i sætningen umiddelbart efter UNTIL-sætningen. Er udtrykkets værdi falsk, fortsætter programudførelsen i sætningen umiddelbart efter den tilhørende REPEAT-sætning.

Kommentarer

1. Hvis sætningslisten indeholder en eller flere sammensatte sætninger, d.v.s. sætninger, som strækker sig over flere linier, skal disse sætninger afsluttes indenfor sætningslisten.

Eksempel

```
0010 // Beregning af kvadratrødder
0020 X:=10
0030 DELTA:=X
0040 SQRT:=X/2
0050 // Iteration
0060 REPEAT
0070   DELTA:=(X/SQRT-SQRT)/2
0080   SQRT:+DELTA
0090 UNTIL ABS(DELTA)<0.001
0100 PRINT SQRT
0110 END
```

Programmet giver følgende udskrift:

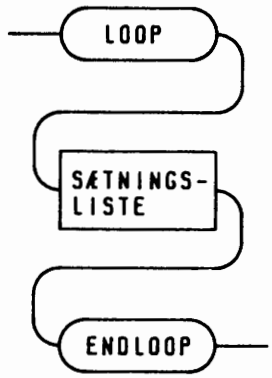
```
3.162278
```

Sammenlign dette eksempel med eksemplet i afsnit 5.10.3.

5.10.5 LOOP/ENDLOOP

LOOP-sætningen bruges til at skabe en løkke, der stadigt gentages, indtil man hopper ud af den.

Syntaks



Udførelse

'Sætningsliste' gentages, indtil man med EXIT eller en GOTO-sætning hopper ud af løkken. EXIT bevirker, at programudførelsen fortsætter i sætningen umiddelbart efter den tilhørende ENDLOOP-sætning.

Kommentarer

Hvis sætningslisten indeholder en eller flere sammensatte sætninger, skal disse sætninger afsluttes inden for sætningslisten.

Eksempel

```
0010 A:=0
0020 LOOP
0030   A:+1
0040   IF A=10 THEN EXIT
0050 ENDLOOP
0060 PRINT A
0070 END
```

Dette program vil udskrive værdien, 10.

5.11 PROCEDURE- OG FUNKTIONSERKLÆRINGER

Procedurer og funktioner kan opfattes som underprogrammer, der kan "kaldes" vilkårligt mange gange fra hovedprogram og som hovedregel også fra andre underprogrammer eller i underprogrammet selv.

Procedurer indledes med nøgleordet PROC <procedurenavn> eventuelt efterfulgt af en parantes, som indeholder et antal parametre.

(En parameter er enten en konstant eller en identifikator, ved hvilken værdien overføres til en procedure eller en funktion. I underprogrammet optræder parameteren som en almindelig variabel).

Procedurer **skal** afsluttes med nøgleordet ENDPROC <procedurenavn>. Mellem PROC og ENDPROC befinder sig procedurens "krop", som består af en eller flere sætninger. Fra hovedprogrammet eller fra andre underprogrammer kaldes en procedure ved anvendelse af nøgleordet EXEC <procedurenavn>.

Funktioner indledes med nøgleordet FUNC <funktionsnavn> og kan ligesom procedurer i erklæringsdelen have en parameterdel. Også parameterløse funktioner afsluttes med en parantes. En funktion **skal** afsluttes med ENDFUNC <funktionsnavn>. I funktionsblokken skal forekomme mindst én sætning med nøgleordet RETURN. Er funktionen konstrueret til at RETURNere en streng (eller et tegn), **skal** <funktionsnavn> afsluttes med "$". Skal funktionen RETURNere et heltal, afsluttes <funktionsnavn> med "#". Hvis hverken tegnet "#" eller "$" benyttes, er den returnerede datatype et reelt tal.

Fra hovedprogrammet eller fra andre underprogrammer kan en funktion kaldes, idet man behandler den som en almindelig variabel. Dersom man "inden i" en procedure/funktion kalder den samme procedure/funktion (underprogrammet kalder sig selv), siges proceduren/funktionen at være **rekursiv.**

```
0010 // Cursorstyring under EPROM F3/F4/F6
0020 FUNC SLUKCURSOR$
0030   RETURN ""27""20""
0040 ENDFUNC SLUKCURSOR$
0050 //
0060 PROC TÆNDCURSOR
0070   PRINT ""27""19""
0080 ENDPROC TÆNDCURSOR
```

Ovenstående funktion og procedure har ingen parametre. Bemærk,

at procedurer og funktioner med fordel kan tildeles meningsfyldte navne. Fra hovedprogrammet kan udføres følgende kald:

```
0010 DIM CR$ OF 1
0020 PRINT SLUKCURSOR$()+"Markøren vises ikke"
0030 INPUT "Tryk <RETURN> ": CR$
0040 EXEC TÆNDCURSOR
0050 END
```

Se i øvrigt brugervejledningen for COMET'en side 4.63, hvor andre kontroltegn vedrørende cursor (markør) omtales.

En parameterløs funktion **skal** kaldes ved funktionsnavnet efterfulgt af en "tom" parantes:

```
0010 X:=FUNKTION1() // X tildeles værdien af FUNKTION1
0020 PRINT FUNKTION2()+X // Summen af funktionsværdierne udskrives
0030 END
```

Bemærk i nedenstående programeksempel anvendelsen af den indbyggede standardfunktion i linie 140. Undlader man at skrive den tomme parantes her, tilføjer fortolkeren selv parantesen.

```
0010 // Eksempel på anvendelse af funktioner
0020 //
0030 FUNC MAX(A, B)
0040   IF A>B THEN
0050     RETURN A
0060   ELSE
0070     RETURN B
0080   ENDIF
0090 ENDFUNC MAX
0100 //
0110 CLEAR
0120 INPUT "Indtast to tal : ": P, Q
0130 PRINT "Største tal er ";MAX(P,Q)
0140 PRINT "Ledig lagerplads: ";FREESTORE();"bytes."
0150 END
```

5.12 EFFEKTEN AF NØGLEORDET REF

Når der i en procedure/funktionserklæring angives en parameterliste, omtales hver af parametrene som værende **formelle parametre.** I en programsætning, hvor en procedure eller en funktion kaldes med en parameterliste, omtales parametrene som **aktuelle parametre.** Dersom formelle og aktuelle parametre ikke stemmer overens med hensyn til antal, rækkefølge og type, afbrydes programmet med en fejlmelding.

Dersom en parameter i en parameterliste er specificeret med nøgleordet REF i erklæringslinien, oprettes der ved procedure/-funktionskaldet en variabel (den formelle parameter), hvis startværdi tildeles den aktuelle parameter.

Indekserede variable skal i erklæringsdelen **altid** være REF-specificerede. I øvrigt gælder det alment, at en aktuel parameter **skal** være en variabel identifikator (ikke en konstant), dersom den tilhørende formelle paramter er REF-specificeret.

```
1000 PROC INDTAST(REF MODTAG$, GODKENDTE$)
1010   REPEAT
1020     POKE 256, 255
1030     WHILE PEEK(256)=255 DO
1040     ENDWHILE
1050     MODTAG$:=CHR$(PEEK(256))
1060   UNTIL MODTAG$ IN GODKENDTE$
1070   PRINT MODTAG$
1080 ENDPROC INDTAST
```

Med ovenstående procedure kan man kontrollere programafviklingen, således at denne først fortsætter, når det indkomne tegn (modtag$) konstateres i strengen, der er repræsenteret i parameteren godkendte$. Hvert "godkendt" indtastet tegn vil blive vist på skærmen (linie 9070) uden forudgående aktivering af VOGN-RETUR tasten. Eksempelvis kunne en programlinie, som kalder proceduren, få følgende form:

```
0200  EXEC indtast(tegn$,"0123456789")
```

Dersom underprogrammet konstrueres som en funktion, får man:

```
1000 FUNC INDTAST$(GODKENDTE$) CLOSED
1010   DIM MODTAG$ OF 1
1020   REPEAT
1030     POKE 256, 255
1040     WHILE PEEK(256)=255 DO
1050     ENDWHILE
```

```
1060     MODTAG$:=CHR$(PEEK(256))
1070   UNTIL MODTAG$ IN GODKENDTE$
1080   RETURN MODTAG$
1090   // Denne linie bliver aldrig gennemløbet
1100 ENDFUNC INDTAST$
```

Funktionen RETURNerer modtag$, og i hovedprogrammet kan den returnerede værdi enten tilskrives en variabel eller udskrives direkte ved følgende funktionskald:

```
0200  tast$:=indtast$("0123456789")
eller
0200  PRINT indtast$("0123456789")
```

I begge tilfælde **skal** variablen tast$ være **defineret:**

```
0010  DIM tast$ OF 1
```

En funktion kan returnere en "delstreng":

```
0010 DIM DAGE#(12)
0020 DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
0030 //
0040 FOR I:=1 TO 12 DO
0050   READ DAGE#(I)
0060   PRINT MANED$(I)+" har ";DAGE#(I);"dage."
0070 NEXT I
0080 //
0090 FUNC MANED$(I) CLOSED
0100   DIM AR$ OF 36
0110   AR$:="JanFebMarAprMajJunJulAugSepOktNovDec"
0120   RETURN AR$(3*I-2:3*I)
0130 ENDFUNC MANED$
```

I det følgende eksempel kan det anbefales interesserede at læse tidsskriftet Datalære, November 1980 for at opnå en dybere forståelse af den i programeksemplet anvendte algoritme.

```
0010 // Eksempel på sortering v.h.a. QUICKSORT algoritme
0020 //
0030 MAX:=100 // Antal elementer i tabel
0040 DIM LISTE(MAX)
0050 //
0060 PRINT "Nu dannes der ";MAX;"tilfældige tal..."
0070 RANDOM
```

```
0080 FOR I:=1 TO MAX DO
0090   LISTE(I):=RND(0,9999) // Værdier mellem 0 og 9999
0100 NEXT I
0110 //
0120 PRINT "Ikke sorteret:"
0130 EXEC SKRIVLISTE
0140 //
0150 PRINT "Nu sorteres tallene..."
0160 EXEC QUICKSORT(LISTE,1,MAX)
0170 //
0180 PRINT "Sorteret:"
0190 EXEC SKRIVLISTE
0200 //
0210 END
0220 //
0230 PROC SKRIVLISTE
0240   FOR I:=1 TO MAX DO
0250     PRINT USING "#### ": LISTE(I);
0260     IF I MOD 15=0 THEN PRINT // 15 tal pr. linie
0270   NEXT I
0280   PRINT
0290 ENDPROC SKRIVLISTE
0300 //
0310 PROC QUICKSORT(REF A(), V, H) CLOSED
0320   I:=V; J:=H; X:=A((V+H) DIV 2)
0330   REPEAT
0340     WHILE A(I)<X DO
0350       I:+1 // Venstre del
0360     ENDWHILE
0370     WHILE X<A(J) DO
0380       J:-1 // Højre del
0390     ENDWHILE
0400     IF I<=J THEN
0410       Y:=A(I); A(I):=A(J); A(J):=Y // Ombyt A(I) og A(J)
0420       I:+1; J:-1
0430     ENDIF
0440   UNTIL I>J // Opdeling afsluttet
0450   IF V<J THEN EXEC QUICKSORT(A,V,J) // Sorter venstre del rekursivt
0460   IF I<H THEN EXEC QUICKSORT(A,I,H) // Sorter højre del rekursivt
0470 ENDPROC QUICKSORT
```

5.13 LUKKEDE PROCEDURER OG FUNKTIONER

Dersom procedure/funktionserklæringen afsluttes med nøgleordet CLOSED, siges proceduren/funktionen at være "lukket".

- Alle variable i det lukkede underprogram er **lokale.** Det betyder, at man såvel i hoved- som underprogram kan benytte samme identifikatorer uden, at disse kommer i konflikt med hinanden.

- Man kan **ikke** fra en **lukket** procedure/funktion **kalde** en **åben** procedure/funktion.

- Variable, som skal gøres tilgængelige for hovedprogrammet, skal erklæres "globale" ved foran en variabelliste at skrive nøgleordet IMPORT. Af hensyn til kompatibilitet med tidligere versioner af COMAL-80 vil fortolkeren, når den møder ordet GLOBAL, oversætte det til IMPORT. Nøgleordet kan i øvrigt **kun** benyttes i en lukket procedure eller funktion.

- Generelt for funktioner: PRINT USING og INPUT kan **ikke** benyttes.

Af nedenstående eksempel fremgår en særlig egenskab ved lukkede underprogrammer:

```
0010 // Program til kontrol af CPR-numre
0020 DIM CPR$ OF 10
0030 //
0040 LOOP
0050   REPEAT
0060     REPEAT
0070       INPUT "Indtast CPR-nummer uden bindestreg: ": CPR$:
0080       IF CPR$="" THEN EXIT
0090     UNTIL LEN(CPR$)=10
0100     EXEC CPRNUMMER(CPR$)
0110   UNTIL NOT ERR()
0120 ENDLOOP
0130 //
0140 PROC CPRNUMMER(REF NUMMER$) CLOSED
0150   DIM VÆGT$ OF 10
0160   VÆGT$:="4327654321"
0170   KONTROL:=0
0180   TRAP ERR-
0190   FOR I:=1 TO 10 DO
0200     KONTROL:+VAL(NUMMER$(I))*VAL(VÆGT$(I))
0210   NEXT I
0220   TRAP ERR+
0230   OKFLAG:=(NOT ERR()) AND (KONTROL MOD 11=0)
```

```
0240   IF NOT OKFLAG THEN
0250     PRINT " - CPR-nummer illegalt!"
0260   ELSE
0270     PRINT " - CPR-nummer OK. ";
0280     IF VAL(NUMMER$(10)) MOD 2=0 THEN
0290       PRINT "(kvinde)"
0300     ELSE
0310       PRINT "(mand)"
0320     ENDIF
0330   ENDIF
0340 ENDPROC CPRNUMMER
0350 END
```

Bemærk i linie 150, at variablen "vægt$" DIMensioneres hver gang, proceduren "cprnummer" kaldes. Dersom dimensionering af samme variabel forekommer mere end en gang i hovedprogram eller i en åben procedure/funktion, vil fortolkeren afbryde med en fejlmelding.

I linie 200 benyttes systemfunktionen VAL, som konverterer en streng bestående af cifre til et reelt tal. Man kunne her lige så vel have benyttet systemfunktionen IVAL, som konverterer en streng til et heltal. I begge tilfælde er det nødvendigt at gardere sig mod eventuel fejlindtastning - og dermed en programafbrydelse - ved i linie 180 at slå fortolkerens "fejlflag" fra med instruktionen TRAP ERR-.

I det følgende eksempel er variablen "produkt" benyttet i både hoved- og underprogram:

```
0010 PRODUKT:=0; TAL1:=6; TAL2:=7
0020 PRINT "Funktionen GANGE returnerer værdien 42"
0030 PRINT TAL1;"* ";TAL2;"= ";GANGE(TAL1,TAL2)
0040 PRINT "Værdien af PRODUKT i hovedprogrammet er stadig 0: ";PRODUKT
0050 //
0060 FUNC GANGE(A, B) CLOSED
0070   PRODUKT:=A*B
0080   RETURN PRODUKT
0090 ENDFUNC GANGE
0100 END
```

5.14 ANVENDELSEN AF NØGLEORDET IMPORT

Nøgleordet IMPORT kan **kun** benyttes i **lukkede procedurer/funktioner** og foranstilles en enkelt variabel eller en variabelliste i underprogrammet. Rækkefølgen er uden betydning. Alle variable i variabellisten skal være definerede i hovedprogrammet og bliver IMPORTeret herfra af underprogrammet.

Følgende programeksempel bør sammenlignes med ovenstående:

```
0010 PRODUKT:=0; TAL1:=6; TAL2:=7
0020 PRINT "Værdien af PRODUKT er 0: ";PRODUKT
0030 PRINT "Proceduren GANGE returnerer PRODUKT =42"
0040 PRINT TAL1;"og ";TAL2;"= "; // linien fortsættes i proceduren GANGE
0050 EXEC GANGE(TAL1,TAL2)
0060 PRINT PRODUKT // PRODUKT har nu fået tildelt værdien 42
0070 PRINT "Hovedprogrammets TAL1 og TAL2 er stadig ";
0080 PRINT TAL1:"og ";TAL2
0090 //
0100 PROC GANGE(A, B) CLOSED
0110   IMPORT PRODUKT // Global variabel
0120   PRINT A;"og ";B
0130   PRODUKT:=A*B
0140   TAL1:=123; TAL2:=456; A:=0; B:=0
0150   PRINT "I underprogrammet har TAL1 og TAL2 værdierne ";
0160   PRINT TAL1;"og ";TAL2
0170   PRINT "produktet er "; // linien fortsættes i hovedprogrammet
0180 ENDPROC GANGE
0190 END
```

```
Værdien af PRODUKT er 0: 0
Proceduren GANGE returnerer PRODUKT =42
6 og 7 = 6 og 7
I underprogrammet har TAL1 og TAL2 værdierne 123 og 456
produktet er 42
Hovedprogrammets TAL1 og TAL2 er stadig 6 og 7
```

Følgende program, der udfører en sortering på strenge, kan sammenlignes med sorteringsprogrammet side 5.48, selvom der ikke er benyttet samme algoritme.

```
0010 INPUT "Hvor mange navne ønskes soreret? ": ANTAL
0020 EXEC ALFASORT(ANTAL)
0060 //
```

```
0070 PROC ALFASORT(ANTAL) CLOSED
0080   DIM NAVN$(ANTAL) OF 25, NR(ANTAL), SORT(ANTAL)
0090   FOR I:=1 TO ANTAL DO
0100     PRINT "Nr. ";I;
0110     INPUT ": ": NAVN$(I)
0120     NR(I):=I
0130   NEXT I
0140   PRINT "Der sorteres...";
0150   FOR N:=1 TO ANTAL DO
0160     L:=0; H:=N
0170     WHILE (H-L)>1 DO
0180       G:=(H-L) DIV 2+L
0190       IF NAVN$(N)>NAVN$(NR(G)) THEN
0200         L:=G
0210       ELSE
0220         H:=G
0230       ENDIF
0240     ENDWHILE
0250     FOR I:=N-1 DOWNTO H DO
0260       NR(I+1):=NR(I)
0270     NEXT I
0280     NR(H):=N
0290   NEXT N
0291   PRINT
0292   FOR I:=1 TO ANTAL DO
0293     PRINT I;" : ";NAVN$(NR(I))
0294   NEXT I
0300 ENDPROC ALFASORT
0310 END
```

```
Hvor mange ønskes sorteret  5
Nr. 1 Wagner
Nr. 2 Beethoven
Nr. 3 Liszt
Nr. 4 Tchaikowskij
Nr. 5 Bach

Der sorteres.

Bach
Beethoven
Liszt
Tchaikowskij
Wagner
```

5.15 FUNKTIONER OG PROCEDURER. OVERSIGT, STRUKTUR, SYNTAX

5.15.1 SYMBOLER

<FNAVN>	:	Funktionens navn
<pnavn>	:	Procedurens navn
<sætningsblok>	:	En eller flere sætninger
<typ>	:	Variabeltype
udeladt	-	Numerisk
$	-	Streng
#	-	Heltal
<param>	:	REF Parameter
udeladt	-	() efter <fnavn>
streng/numerisk	-	valgfrit
indekseret	-	nødvendig
<ident>	:	Identifikator for variabel
<var typ>	:	Variabel af samme type som <typ>. Kan dog også være en konstant.

5.15.2 FUNKTION: FUNC - ENDFUNC

```
FUNC <fnavn> <typ> (<param>,... ) CLOSED
   IMPORT  <ident>,...
   <sætningsblok>
   RETURN <var typ>
ENDFUNC <fnavn> <typ>
```

Bemærkning: Sætninger med INPUT eller PRINT USING må ikke forekomme.

5.15.3 PROCEDURE: PROC - ENDPROC

```
PROC <pnavn> (<param>,... ) CLOSED
   IMPORT  <ident>,...
   <sætningsblok>
ENDPROC <pnavn>
```

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

5.15.4 GENERELLE BEMÆRKNINGER

- Mindst én RETURN-sætning skal forekomme i en funktions sætningsblok.

- Man kan RETURNere fra en procedure med instruktionen RETURN. For anvendelse af andre RETURN-sætninger henvises til afsnit 5.21.2 og 5.21.3 (GOSUB - RETURN).

- Formelle parametre skal være REF-specificerede, når de tilhørende aktuelle parametre er indekserede. (Se afsnit 5.12).

- LABELs (etiketter afsnit 5.8 i hovedprogrammet er lokale for hovedprogrammet. På samme måde er LABELs lokale i den funktion/procedure, hvor de er definerede. Dette bevirker, at hopinstruktioner til/fra funktioner/procedurer ikke kan udføres ved anvendelse af sætningen GOTO <label>. Selvom instruktionen GOTO <linienummer> kan udføres i ovennævnte sammenhæng, anses denne fremgangsmåde som dårlig programmeringspraksis og strider i øvrigt mod idéen i et strukturorienteret programmeringssprog.

- I CLOSED (lukkede) funktioner/procedurer er de anvendte variable lokale, dersom de ikke gøres globale ved en IMPORT-sætning. Det samme gælder de formelle parametre, dersom disse ikke er REF-specificerede.

- En IMPORT-sætning må kun forekomme i CLOSED funktioner/-procedurer.

- Kald af åbne funktioner/procedurer fra lukkede funktioner/procedurer kan ikke udføres.

- Brugerdefinerede funktioner/procedurer kan ikke udføres som kommando.

- Funktions-, label-, procedure- og variabelnavne kan være ens uden konflikt, idet sammenhængen under anvendelsen afgør betydningen af navnet.

5.16 SAMMENKÆDNING AF PROGRAMMER VED EN CHAIN-SÆTNING

Der kan være flere grunde til at sammenkæde programmer:

- Fra et "menu-program" ønskes at vælge forskellige, indbyrdes uafhængige programmer, der efter eksekvering returnerer til "menu-programmet".
- Det kan ske, at et program bliver så stort, at det ikke kan være i hovedlageret på en gang, hvorfor programmet må deles op i to eller flere programdele.

Ved hjælp af **instruktionen**

CHAIN <PROGRAMNAVN>,

kan man fra en programdel kalde og eksekvere en anden programdel (eller selvstændigt program) under forudsætning af, at <programnavn> er gemt på binær form (programmet er lagret på disketten ved ordren **SAVE**) med typebetegnelsen CSB efter <programnavn>.

5.16.1 ANVENDELSE AF NØGLEORDET RECEIVE

Det CHAINede program vil kunne modtage variable af enhver type. Først et simpelt eksempel:

```
0010 DIM TXT$ OF 50
0020 TXT$:="Denne tekst er overført fra programmet NR1"
0030 PRINT "Nu CHAINes til programmet NR2"
0040 X:=4
0050 CHAIN "NR2", TXT$, X
0060 END
```

Programmet giver følgende udskrift:

```
Nu CHAINes til program NR2:
Denne tekst er overført fra program NR1.
Fra program NR1 er modtaget:
x = 4
```

* LIST

```
0010 RECEIVE TXT$, X
0020 PRINT TXT$
0030 PRINT "Fra programmet NR1 er modtaget x = ";X
0040 END
```

RECEIVE skal indeholdes i første programsætning. Parameterlisten behøver ikke at have de samme identifikatorer, men typen og rækkefølgen **skal** stemme overens i såvel det "sendende" som det "modtagende" program.

Strengvariable og indekserede variable fra parameterlisten skal i det CHAINede program ikke defineres i en DIM-instruktion.

Det er naturligvis en forudsætning, at det program, der refereres til ved en CHAIN-instruktion, eksisterer (Diskette kan være udskiftet). Det kan derfor være nødvendigt med en test for at sikre sig, at programmet eksisterer:

```
0010 PROC CHAINFIL(ENHED$, FILNAVN$)
0020   TRAP ERR-
0030   CHAIN ENHED$+FILNAVN$
0040   TRAP ERR+
0050 ENDPROC CHAINFIL
```

Ovenstående procedure kan eksempelvis benyttes i følgende programsekvens:

```
0010 DIM CR$ OF 1
0020 LOOP
0030   EXEC CHAINFIL("DK1:","MENU")
0040   PRINT "Monter disketten med MENU-programmet i drev B:"
0050   INPUT "Tryk <RETURN> når du er klar ": CR$
0060 ENDLOOP
0070 END
```

5.17 RANDOM-SÆTNING (RANDOMIZE)

RANDOM-sætningen benyttes, når det ønskes, at de tilfældige tal, genereret af RND-funktionen (se afsnit 6.2), skal starte et tilfældigt sted i sekvensen af (pseudo-) tilfældige tal.

Syntaks

RANDOM-sætningen har følgende opbygning:

Udførelse

Som omtalt ovenfor.

Kommentarer

1. Normalt vil RND-funktionen generere samme sekvens af tilfældige tal for hver gang, en RUN-kommando udføres. Dette kan være nyttigt under programafprøvning.

Eksempel

```
0010 FOR I:=1 TO 10 DO
0020   PRINT RND(1,20);
0030 NEXT I
0040 END
```

Udføres ovenstående program et antal gange, vil det hver gang producere den samme sekvens af pseudo-tilfældige tal. Indsættes linien

```
0005 RANDOM
```

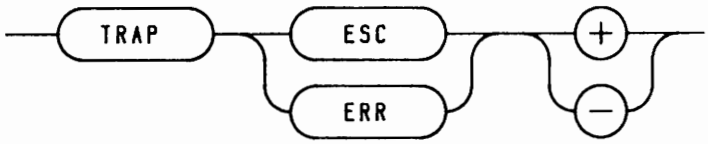
i programmet, vil gentagne kørsler ikke producere den samme sekvens af pseudo-tilfældige tal.

5.18 TRAP-SÆTNING

TRAP-sætningen benyttes til at skifte mellem programstyret reaktion og standardreaktion på aktivering af ESC-tast samt på visse fejlsituationer.

Syntaks

TRAP-sætningen har følgende opbygning:



Udførelse

TRAP-sætningens funktion kan forklares på følgende måde:

ESC og ERR er navnene på to logiske systemvariable. Hver gang en programudførelse starter (RUN), tildeles ESC og ERR automatisk værdien sand. Har ESC værdien sand, betyder det, at et tryk på tastaturets ESC-tast medfører en øjeblikkelig afbrydelse af programudførelsen. Har ERR værdien sand, betyder det, at fejl i programudførelsen vil medføre, at programmet standser med en fejlmeddelelse.

Har ESC værdien falsk, vil aktivering af tastaturets ESC-tast ikke medføre en programafbrydelse, men vil blot medføre, at den logiske systemvariabel ESC() bliver tildelt værdien sand. Har ERR værdien falsk, vil visse fejlsituationer ikke resultere i programafbrydelse, men blot medføre, at den logiske systemvariabel ERR() bliver tildelt værdien sand (i form af en værdi <> 0, der angiver fejlens nummer, jfr. afsnit 9.5).

Når ESC og ERR starter med værdien falsk, er det således muligt fra brugerprogrammet at konstatere, om ESC-tasten har været aktiveret, og om fejlsituationer er forekommet, ved at man aflæser værdien af ESC() og ERR().

F.eks. ved forsøg på åbning af filer kan programmet sættes til at reagere som ønsket på evt. fejl.

ESC og ERR omtales nærmere i afsnit 6.4.

Efterfølges 'ESC' eller 'ERR' af et '+' i TRAP-sætningen, tildeles den tilsvarende af systemvariablene ESC og ERR værdien 'sand'. Efterfølges ESC eller ERR af et '-' i TRAP-sætningen,

tildeles den tilsvarende af systemvariablene ESC og ERR værdien 'falsk'.

Kommentarer

1. De fejlsituationer, som ikke vil medføre programafbrydelse, når ERR er falsk, er situationer, som COMAL-80 fortolkeren kan omgå på en "naturlig" og veldefineret måde. Sådanne fejl kaldes i det følgende: Ikke-fatale fejl.

   Division med 0 er et eksempel på en ikke-fatal fejl. Resultatet af en sådan division er defineret ved

   X/0 = SGN(X)*MAX_REEL

   hvor MAX_REEL er den største værdi, som et reelt udtryk kan antage.

   En indeksfejl er et eksempel på en fatal fejl, idet den ikke kan omgås på en "naturlig" og veldefineret måde.

Eksempel

```
0010 // Eksempel på brug af TRAP ESC
0020 //
0030 // Programmet afsluttes ved tryk på ESC tasten
0040 //
0050 TRAP ESC-
0060 REPEAT
0070   PRINT "Om hundrede år er alting glemt"
0080   PRINT "men et vil blive husket:"
0090   FOR I:=1 TO 400 DO // Vent
0100   NEXT I
0110 UNTIL ESC()
0120 //
0130 PRINT "Hov..."
0140 PRINT "Århundreder går jo meget nemt"
0150 PRINT "på denne vis, men gudskelov"
0160 PRINT "så stopper jeg dette dumme skæmt"
0170 PRINT "og håber det bli'r husket."
0180 END
```

Ovenstående program udskriver en række linier. Programmet kører i en løkke og kan kun standses ved tryk på ESC-tasten. På grund af TRAP-sætningen i linie 50 er det imidlertid umuligt at standse programmet midt i udskrivningen af en linie inde i løkken. Programudførelsen kan kun bringes til afslutning, hvis man træder ud af løkken i linie 110.

5.19 STOP-SÆTNING

STOP-sætningen anvendes til at stoppe udførelsen af et program.

Syntaks

STOP-sætningen har følgende opbygning:

Udførelse

Udførelse af en STOP-sætning medfører, at programudførelsen standser med følgende udskrift på skærmen:

```
STOP i linie XXXX
```

hvor XXXX erstattes med STOP-sætningens linienummer.

Eksempel

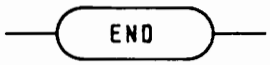
Se næste afsnit.

5.20 END-SÆTNING

END-sætningen anvendes til at stoppe udførelsen af et program.

Syntaks

END-sætningen har følgende opbygning:



Udførelse

Udførelse af en END-sætning medfører, at programudførelsen standser.

Kommentarer

1. END-sætningen benyttes oftest til at angive den fysiske afslutning på et program, hvor den altså placeres i programmets sidste linie. Det er imidlertid ikke nødvendigt at angive programmets fysiske afslutning på denne måde, idet COMAL-80 fortolkeren selv placerer et for brugeren usynligt END efter den sidste linie i et program. Når dette END mødes under programudførelsen, udskrives meddelelsen:

   Programudførelse afsluttet

Eksempel

```
0010 FOR I:=1 TO 2 DO
0020   INPUT "Skriv et tal ": TAL
0030   IF TAL=0 THEN STOP
0040 NEXT I
0050 END
```

Herunder vises skærmdialogen for to forskellige udførelser af ovenstående program. Brugersvar er understreget.

1. Udførelse:

   SKRIV ET TAL 0

   STOP I LINIE 0030

2. Udførelse:

```
SKRIV ET TAL 1
SKRIV ET TAL 2
```

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

5.21 SPECIELLE SÆTNINGER

5.21.1 INDLEDNING

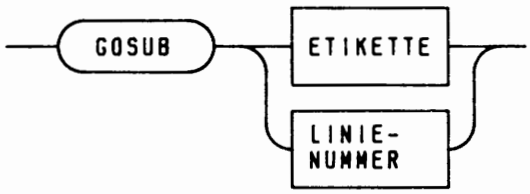
I dette afsnit beskrives nogle sætninger, som ikke falder naturligt ind i sprogstrukturen i COMAL-80. Sætningerne er "BASIC-levn", som kun er medtaget i COMAL-80 for ikke at umuliggøre kørsel med BASIC-programmer.

5.21.2 GOSUB-SÆTNING

GOSUB-sætningen er et subrutinekald.

Syntaks

GOSUB-sætningen har følgende opbygning:



Udførelse

Udførelse af GOSUB-sætningen bevirker, at subrutinen, som starter i linie 'linienummer', kaldes.

Programudførelsen starter i sætningen med det angivne linienummer og fortsætter herfra, indtil en RETURN-sætning mødes. Herefter fortsætter programudførelsen i sætningen umiddelbart efter GOSUB-sætningen.

Det er tilladt subrutiner at kalde (GOSUB) andre subrutiner, og det er også tilladt subrutiner at kalde sig selv, hvorved en slags rekursivitet opnås.

5.21.3 RETURN-SÆTNING

RETURN-sætningen benyttes til at angive, at returhop fra en subrutine skal foretages.

Syntaks

RETURN-sætningen har følgende opbygning:

Udførelse

Når en RETURN-sætning mødes, vil programmet fortsætte programudførelsen i sætningen umiddelbart efter den sidst eksekverede GOSUB-sætning.

Mødes en RETURN-sætning, når ingen subrutine er kaldt, standser programudførelsen med en fejlmeddelelse.

Kommentar

Linienummeret, som angives i GOSUB-sætningen samt den tilhørende RETURN-sætning, skal findes i hovedprogrammet (ikke i en funktion eller procedure), med mindre både den kaldende sætning samt hele subrutinen indgår i samme sætningsliste.

Eksempel

```
0010 X:=3; Y:=5
0020 GOSUB 0060
0030 X:=8; Y:=11
0040 GOSUB 0060
0050 STOP
0060 // Kvadratsum
0070 PRINT "X^2 + Y^2 = ";X^2+Y^2
0080 RETURN
```

Udføres ovenstående program fås følgende udskrift:

```
X^2 + Y^2 =  34
X^2 + Y^2 = 185

STOP i linie 0050
```

Bemærk, at udeladelse af STOP-sætningen i linie 50 i ovenstå-

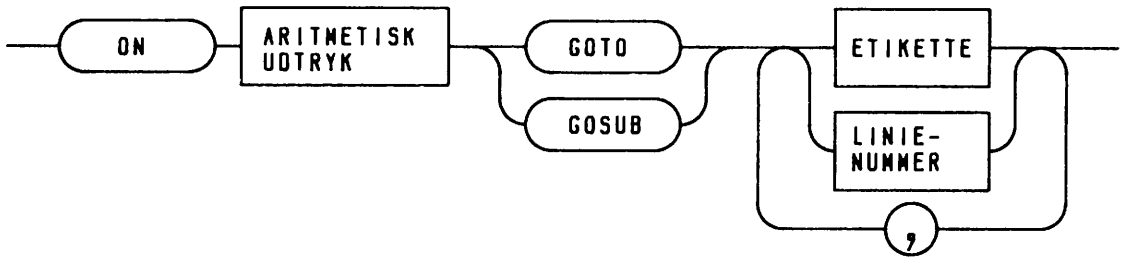
de eksempel ville resultere i en fejlmeddelelse i linie 80.

5.21.4 ON-GOTO/ON-GOSUB-SÆTNINGER

ON-GOTO- og ON-GOSUB-sætningerne benyttes til at vælge et af en række linienumre, hvori programudførelsen ønskes fortsat. Valget af linienummer foretages ud fra værdien af et aritmetisk udtryk.

Syntaks

Sætningerne har følgende opbygning:



Udførelse

Det aritmetiske udtryk beregnes. Lad os kalde værdien af udtrykket for n. Programudførelsen fortsætter herefter i linien, hvis linienummer forekommer som det N'te i listen af linienumre.

Ved udførelsen af en ON-GOSUB-sætning foretages returhop til sætningen umiddelbart efter denne, når et RETURN mødes.

Kommentarer

1. Hvis beregningen af det aritmetiske udtryk ikke giver et helt tal, foretages en afrunding.

2. Hvis den afrundede værdi af n ikke opfylder kravet

   1 <= n <= antal linienumre i listen,

   udføres sætningen umiddelbart efter ON-GOTO/ON-GOSUB sætningen.

Eksempel

```
0010 TAL:=2
0020 ON TAL GOTO 0050, 0070, 0090
0030 PRINT "TAL<1 eller TAL>3"
0040 GOTO 0100
0050 PRINT "TAL=1"
0060 GOTO 0100
```

```
0070 PRINT "TAL=2"
0080 GOTO 0100
0090 PRINT "TAL=3"
0100 PRINT "Undersøgelse slut"
0110 END
```

Programmet giver følgende udskrift:

```
TAL=2
Undersøgelse slut
```

Ændres linie 10 til

```
0010 TAL := -3
```

vil udførelsen af programmet give følgende udskrift:

```
TAL<1 eller TAL>3
Undersøgelse slut
```

6 Standardfunktioner og systemvariabler

INDHOLDSFORTEGNELSE KAPITEL 6 - STANDARDFUNKTIONER OG SYSTEMVARIABLE

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

6. STANDARDFUNKTIONER OG SYSTEMVARIABLE

6.1 INDLEDNING

I dette afsnit beskrives standardfunktioner, systemvariable og maskinsprogsfunktioner. Det er valgt at dele beskrivelsen op i følgende 4 dele:

1. Aritmetiske standardfunktioner:

SIN, COS, TAN, ATN, LOG, EXP, SQR, INT, FRAC, TRUNC, ROUND, RND, POS, LEN, ORD, IVAL, VAL og BVAL.

2. Tegnorienterede standardfunktioner:

CHR$, SPC$, STR$, BSTR$ og ERRTEXT$.

3. Systemvariable:

ZONE; EDS og ERR; EOD og EOF.

4. Maskinsprogsfunktioner:

INP, OUT, PEEK, POKE, CALL og VARPTR samt den særlige QUIT-sætning.

6.2 ARITMETISKE STANDARDFUNKTIONER

Aritmetiske standardfunktioner kan indgå i aritmetiske udtryk.

Parametrene til de aritmetiske standardfunktioner må, hvis intet andet udtrykkeligt er nævnt, være udtryk af den angivne type. Til angivelse af parametrenes type er følgende navneregler anvendt:

r : reel
h : heltallig
x : reel eller heltallig
s$: tegnstreng

Typesøjlen angiver resultatets type, og her er følgende forkortelser anvendt:

H : heltallig
R : reel

De aritmetiske funktioner er følgende:

Navn	Type	Resultat
SIN(x)	R	Sinus til x. x i radianer.
COS(x)	R	Cosinus til x. x i radianer.
TAN(x)	R	Tangens til x. x i radianer.
ATN(x)	R	Arcus tangens til x, udtrykt i radianer.
LOG(x)	R	Den naturlige logaritme af x.
EXP(x)	R	Eksponentialfunktionen taget af x.
SQR(x)	R	Kvadratroden af x. x >= 0
ABS(x)	som x	Den absolutte værdi af x.
SGN(x)	H	0 hvis x = 0; 1 hvis x > 0; -1 hvis x < 0
INT(r)	R	Heltalsdelen af r, d.v.s. tallet som fremkommer ved bortkastning af decimalerne. Bemærk, at INT giver et heltalligt resultat af reel type. Dette gør, at man kan operere med heltallige værdier, der er numerisk langt større end den grænse på ca. 32.000, der er for værdier af heltalstype.
FRAC(r)	R	ABS(r - INT(r)); decimaldelen
TRUNC(r)	H	Det hele tal, der fås ved bortkastning af decimalerne.
ROUND(r)	H	Det hele tal, der fås ved at runde af (op eller ned). Hvis decimaldelen af x er ½, rundes der op i retning af den positive talakse.
RND()	R	Et (pseudo-) tilfældigt tal i intervallet 0,1
RND(x,y)	H	Et (pseudo-) tilfældigt tal i intervallet x,y . Er x og y ikke heltallige, foretages en afrunding.
POS(s1$,s2$)	H	Lad os kalde resultatet n. Hvis s1$ forekommer i s2$, er n lig med startpositionen for den første forekomst (regnet fra venstre mod højre) af s1$ i s2$. Hvis s1$ er lig med den tomme streng, er n=1. Hvis s1$ ikke forekommer i s2$, er n=0. Eks. POS ("BC","ABCD") = 2 POS ("" ,"BBB") = 1 POS ("BC","BBB") = 0
LEN(s$)	H	Aktuel længde af s$. s$ skal være en strengvariabel uden selektor-angivelse.

ORD(s$)	H	Den decimale ASCII-kode for første tegn i s$.
IVAL(s$)	H	Det hele tal, der forefindes som en tegnstreng i s$. Eks. IVAL ("357") = 357
VAL(s$)	R	Det reelle tal, der forefindes som en tegnstreng i s$. Eks. VAL("35.7") = 35.7
BVAL(s$)	H	Det hele tal, der forefindes som en binær tegnstreng på nøjagtigt 8 pladser i s$. Eks. BVAL("00110101") = 53

6.3 TEGNORIENTEREDE STANDARDFUNKTIONER

Tegnorienterede standardfunktioner kan indgå i strengudtryk. Værdien af en sådan standardfunktion er en tegnstreng.

De tegnorienterede standardfunktioner er følgende:

Navn	Resultat
CHR$(x)	Tegnet, som har den decimale ASCII-kode x. Er x ikke heltallig, foretages afrunding. x skal (efter en evt. afrunding) ligge i intervallet 0,255
BSTR$(x)	Tallet,x, udtrykt som en binær tegnstreng på nøjagtigt 8 pladser. Er x ikke heltallig, foretages en afrunding. x skal (efter en evt. afrunding) ligge i intervallet 0,255
ERRTEXT$(x)	En tegnstreng, som indeholder den af systemets fejlmeldinger, hvis nummer er angives med x. Denne funktion virker kun, hvis systemets fejlmeldinger ikke er frakoblet.
SPC$(x)	En tegnstreng bestående af x mellemrum (blanktegn). Er x ikke heltallig, foretages afrunding.
STR$(x)	Tallet x, som en tegnstreng. Eks. A$:= STR$(12.2*3)

Efter ovenstående tildeling indeholder A$ tegnstrengen "36.6".

Parameteren til de ovenfor beskrevne tegnorienterede standardfunktioner er et aritmetisk udtryk.

6.4 SYSTEMFUNKTIONER OG SYSTEMVARIABLE

COMAL-80 har nogle forud definerede variable, som i nogle tilfælde kan tildeles en værdi af brugeren. I andre tilfælde returnerer fortolkeren en variabel med en værdi, brugeren kan benytte til at kontrollere en bestemt tilstand under et programs afvikling.

6.4.1 SYSTEMFUNKTIONERNE EOD() OG EOF()

EOD er et acronym for End Of Data, og EOF er et acronym for End Of File. Begge funktioner returnerer værdien 0 (falsk), så længe der kan indlæses data fra **datalinier** (EOD) eller fra **datafiler** (EOF). Efter afsluttet indlæsning returneres værdien 1 (sand).

Følgende to eksempler viser anvendelsen af funktionerne:

```
0010 DATA 2, 6, 7, 5, 4, 6, 2, 5, 7
0020 //
0030 WHILE NOT EOD() DO
0040   READ TAL
0050   PRINT TAL
0060 ENDWHILE
0070 END
```

```
0010 OPEN FILE 1, "TAL", READ
0020 //
0030 WHILE NOT EOF(1) DO
0040   READ FILE 1: TAL
0050   PRINT TAL
0060 ENDWHILE
0070 CLOSE FILE 1
0080 END
```

6.4.2 SYSTEMFUNKTIONEN ERR()

Under programafviklingen skelner fortolkeren mellem **fatale fejl** og **ikke-fatale fejl**. I begge tilfælde vil programmet stoppe, og fortolkeren giver en fejlmelding. Når der er tale om ikke-fatale fejl, kan brugeren opbygge sit program således, at det ikke bliver afbrudt. Se eksempler under afsnit 5.13 og 5.16.1.

6.4.3 SYSTEMFUNKTIONEN ESC()

Under indlæsning af en initialiseringsfil kan man ikke stoppe

programmet ved at trykke på ESC-tasten. En tilsvarende tilstand vil kunne opnås under hele eller dele af programafviklinger ved henholdsvis at slå ESC-'flaget' til eller fra. Dette sker med instruktionerne TRAP ESC- og TRAPESC+.

6.4.4 VÆRDITILDELING AF SYSTEMVARIABLE

INDENTION

Ved opstart af COMAL-80 er værdien af denne systemvariabel lig med 2, hvilket medfører, at alle strukturerede programlinier indrykkes 2 pladser i forhold til den linie, der indeholder nøgleordet til indledning af strukturen. Dersom INDENTION tildeles en af værdierne 0..10, vil indrykningen udføres med den nye værdi.

IDENTIFIERLOWER

Har ved opstart værdien 0, som bevirker, at brugerdefinerede variable, procedure- og funktionsnavne, ved listning på skærm eller printer, udskrives med store bogstaver. Tildeles variablen værdien 1, udskrives disse variabelnavne i stedet med små bogstaver.

KEYWORDLOWER

De samme forhold som ved IDENTIFIERLOWER - blot gældende nøgleordene i COMAL-80.

PAGELENGTH

Er en intern tællervariabel, som ved opstart har værdien 72. Ved udskrift via printer vil instruktionen PAGE føre en hel (A4) side frem. Dersom der forud for instruktionen har været sendt f.eks. 20 linier frem til printeren, vil instruktionen bevirke, at papiret føres 52 linier frem.

PAGELENGTH kan tildeles værdierne 0..254, og dersom man benytter værdien 0, vil printeren ved udførslen af PAGE-instruktionen modtage en Form Feed kode (ASCII værdi 12).

PAGEWIDTH

Er ligeledes en intern tællervariabel, der styrer antal tegn pr. linie. Ved opstart er værdien 80, men kan tildeles enhver af værdierne 0..254, idet værdien 0 i denne sammenhæng har betydningen 'uendelig'.

Ved tegning af grafik, som skal sendes til en printer, bør både PAGELENGTH og PAGEWIDTH tildeles værdien 0.

ZONE

Er den variabel, hvis værdi bestemmer zonebredden ved PRINT- og INPUT-sætninger. Ved opstart er værdien 0.

Ovennævnte systemvariable bevarer den tildelte værdi efter kommandoen NEW.

6.5 MASKINSPROGSFUNKTIONER

Disse funktioner tillader at knytte maskinsprogsprogrammer for Z-80 mikroprocessoren til et COMAL-80 program. Denne facilitetet skal anvendes med stor forsigtighed, da alle normale sikkerhedscheck er frakoblet.

INP(x)	Læser systemport nummer x og omsætter resultatet til decimaltal.
OUT x,y	Skriver y til port x
PEEK(x)	Læser hukommelsesplads x og omsætter værdien til decimaltal.
POKE x,y	Skriver y i hukommelsesplads x
CALL x	Starter afviklingen af maskinsprogsprogrammel liggende i hukommelsesplads x og fremefter. Man kommer tilbage til COMAL-80 ved at afslutte maskinsprogsprogrammet med værdien for RETURN (201).
VARPTR(I)	Finder den absolutte adresse i decimaltal i hukommelsen, hvor 1. byte af variablen "I" gemmes.

x og y er her numeriske udtryk, der opfattes som decimaltal.

QUIT

Det er naturligt på dette sted at nævne QUIT-sætningen, der gør, at man træder ud af det normale COMAL-80 regi og kommer tilbage til operativsystemet CP/M.

Syntaks

7 Filsystem

INDHOLDSFORTEGNELSE KAPITEL 7 - FILSYSTEM

7. FILSYSTEM

7.1 INDLEDNING

COMAL-80 giver mulighed for at overføre data til og fra sekundære lagre som kasettebånd eller disketter.

En sammenhørende mængde af data, som er lagret på et sekundært lager, vil i det følgende blive kaldt en fil.

I dette kapitel beskrives sætninger af følgende art:

a. Det, der har at gøre med håndtering af enheder og filer:

   CAT, UNIT, GETUNIT, RENAME, DELETE, INIT, RELEASE.

b. Det, der har at gøre med skrivning og læsning af data under programudførelse:

   OPEN, CLOSE, PRINT FILE (USING), INPUT FILE, WRITE FILE og READ FILE.

I forbindelse med brug af filer kan det i starten være svært at se, hvad der skal gøres, og hvornår det skal gøres. Nedenfor er derfor beskrevet, hvorledes en typisk brug af maskinen kan forme sig.

Eksemplet refererer til brug af disketteversionen.

Det, som brugeren indtaster, står i klart sprog. (RETURN er dog underforstået). Maskinens svar eller handling er her indikeret ved understregning eller sidestreger og er ikke altid skrevet fuldt ud i dette eksempel:

```
* NEW
* AUTO
0010    OPEN # 8, "RÅDATA", WRITE
0020    DIM A$ OF 3
0030    FOR I=1 TO 5
0040    INPUT "DYR? ": A$
0050    PRINT FILE 8: A$
0060    NEXT I
0070    CLOSE # 8
0080    END
0090    (tryk på ESC)
* EDIT 80
0080 END // NU ER DET FÆRDIGT
* LIST
```

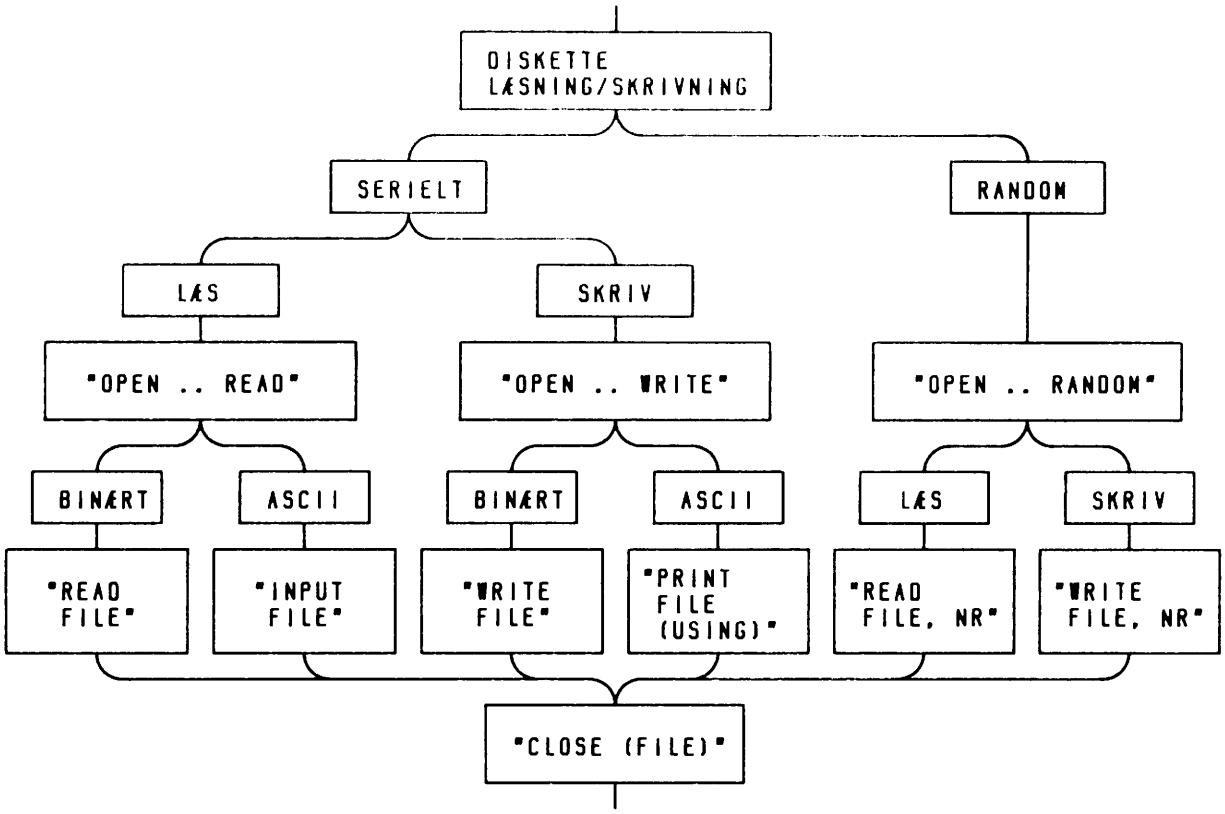
```
0010    OPEN FILE 8, "RÅDATA", WRITE
0020    DIM A$ OF 3
0030    FOR I:=1 TO 5 DO
0040    INPUT "DYR? ": A$
.
.
.
.
0080    END // NU ER DET FÆRDIGT

* INIT (efter indsættelse af diskette)
* RUN
DYR? FÅR
DYR? LAM
DYR? SÆL
DYR? GED
DYR? LOS
* LIST LP: (udskrift af program på printer)
* LIST RÅDATPR (listning af diskette på program)
* NEW
* AUTO
0010    OPEN # 2, "RÅDATA.DAT", READ
0020    DIM A$ OF 3
0030    FOR I=1 TO 4
0040    INPUT FILE 2: A$
0050    PRINT A$;" ";
0060    NEXT I
0070    CLOSE
0080    END
0090    (tryk på ESC)
* RUN

FÅR LAM SÆL GED
* RELEASE (for en sikkerheds skyld)
* PRINT "SLUT PÅ EKSEMPEL"
SLUT PÅ EKSEMPEL
```

Det anbefales yderligere at studere nedenstående diagram over, hvorledes læsning og skrivning i filer kan udføres.

DIAGRAM OVER LÆSNING OG SKRIVNING I COMAL-80:



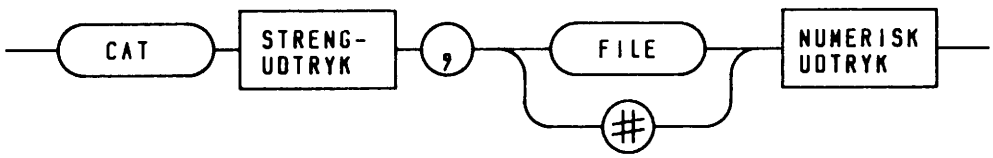
Da det ikke fremgår af diagrammet, må det bemærkes, at der kan være flere forskellige filer åbne ad gangen, og der kan læses/-skrives til dem i flæng.

7.2 CAT

CAT, brugt som sætning, bruges til at udskrive oplysninger fra kataloget for en enhed til en specificeret fil.

Syntaks

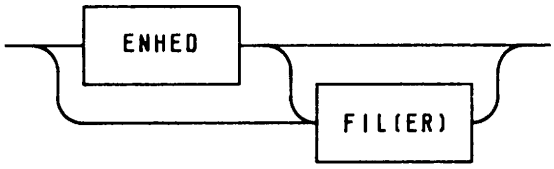
CAT-sætningen har følgende form:



Det, der står efter kommaet, specificerer den serielle fil, hvori man ønsker oplysningerne nedskrevet.

Den skal i forvejen i programmet være åbnet for skrivning (WRITE) med en OPEN...WRITE-sætning og med et filnummer = den afrundede værdi af "Numerisk udtryk".

Indholdet af 'strengudtryk' angiver enheden/filerne på følgende format:



Man kan desuden gøre brug af "Fuzzy characters", nemlig "*" og "?".

"*" refererer til en gruppe af karakterer.

"?" refererer til 1 karakter.

Hvis man f.eks. vil referere til alle filer, der har B og C som 2. og 3. karakter i navnet, og som er af vilkårlig type, og som ligger på enheden DK1:, kan det gøres ved sætningen:

```
CAT "DK1:?BC?????*",#6
```

medens reference til alle COMAL programmer på standardenheden kan ske med sætningen:

```
CAT "*.CSB",#2
```

Udførelse:

Kataloget over alle filer på enheden, der passer til beskrivelsen, udskrives linievis på filen.

Eksempler

```
CAT "DK1":, FILE 1
CAT "DK0:*.C??", FILE 0
CAT "*.CSB", # 0
```

Anvendelse

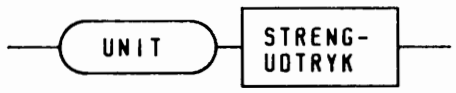
Efter lukning og efterfølgende åbning af filen for læsning kan man med INPUT FILE... indlæse oplysningerne i en strengvariabel. Herved kan man altså under programkontrol undersøge, hvilke filer af en bestemt slags, der er til stede.

7.3 UNIT

UNIT fastlægger den magnetiske baggrundsenhed, som herefter skal være standardenheden. Fra starten af, er det enheden med det laveste nummer (DK0).

Syntaks

UNIT-sætningen har følgende udseende:



Strengudtryk skal indeholde navnet på den ønskede standard-enhed.

Udførelse

Den udpegede enhed er standard, indtil en ny enhed udpeges ved hjælp af en UNIT-sætning eller -kommando.

Eksempler

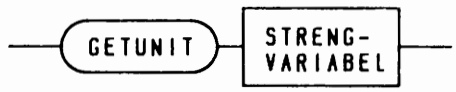
```
UNIT "DK1:"
UNIT "DK0:"
```

7.4 GETUNIT

GETUNIT bruges til at undersøge, hvad den aktuelle standarden-hed er.

Syntaks

GETUNIT-sætningen har følgende udseende:



Udførelse

Navnet på den aktuelle standardenhed indlæses i 'Strengvaria-bel' efter de samme regler som ved tildeling til en simpel

eller indiceret strengvariabel eller en del heraf.

Eksempler

```
GETUNIT A$
GETUNIT ENHED$
```

7.5 RENAME

RENAME bruges til at give en fil til et nyt navn.

Syntaks

RENAME-sætningen har følgende form:

De to strengudtryk betegner henholdsvis den gamle og den nye fil. Hvis "enhed" ikke angives, bruges standardenheden. Hvis mindst en enhed er angivet, skal de to enheder, der bruges, stemme overens. Hvis filen har en type, skal den angives.

Det anbefales, at man både for den gamle og den nye filbetegnelse angiver type.

Udførelse

Den nye filbetegnelse erstatter den gamle, således at den gamle betegnelse altså ikke længere optræder på enheden.

Eksempler

```
RENAME "DK1:NAVN.1.CSB","DK1:A.CSB"
RENAME "GLFIL.DAT","NYFIL.DAT"
```

7.6 DELETE

DELETE bruges til at slette filer (programmer og data) på en diskette.

Syntaks

Strengudtryk er en beskrivelse af den eller de filer, der skal slettes.

Udførelse

Der kan slettes flere filer ad gangen ved hjælp af et udtryk, der følger de samme regler for filangivelser som ved CAT-sætningen.

Eksempler

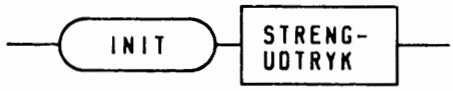
```
DELETE "DK1:PROGRAM.CSB"
DELETE "*.CSB"
DELETE "C???????.CSB"
DELETE "DK0:PROG1.*"
```

7.7 INIT

INIT bruges til at gøre katalogblokken for en diskette kendt for systemet.

Syntaks

INIT-sætningen har følgende form:



Strengudtryk er en enhedsbetegnelse.

Kommentarer

Det er tilrådeligt at brug INIT hver gang, man monterer en ny diskette i en af diskettestationerne, og nødvendigt inden man skriver på en diskette, der ikke har været INIT'ieret.

Eksempler

```
INIT "DK0:"
INIT DK0        (kan kun benyttes som kommando)
```

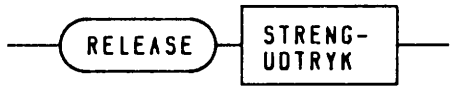
7.8 RELEASE

7.8.1 RELEASE VED DISKETTESYSTEMER

RELEASE bruges til at undersøge, om alle filer på en diskette er lukket.

Syntaks

RELEASE-sætningen har følgende form:



Værdien af 'Strengudtryk' skal være en enhedsbetegnelse.

Udførelse

Hvis der stadig er åbne filer på disketten, vil RELEASE-sætningen resultere i en fejl (som man evt. derefter kan tage sig af).

Eksempler

```
RELEASE "DK0:"
RELEASE "DK1:"
RELEASE "DK"+A$+":"
```

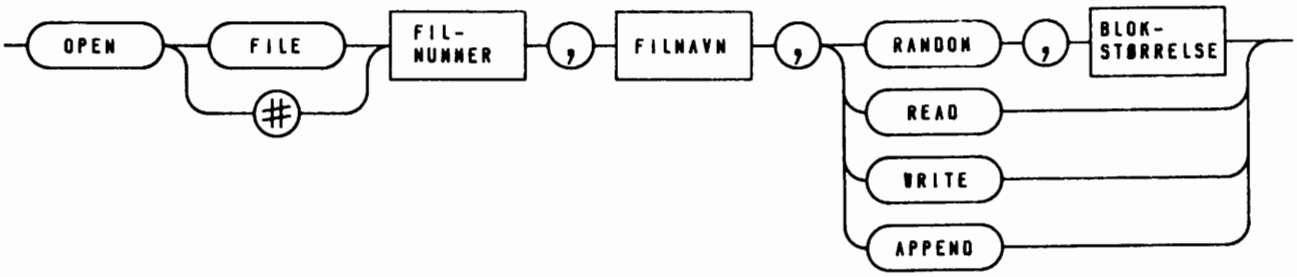
hvor A$ f.eks. kan have værdien "0" eller "1".

7.9 OPEN

Før en fil kan benyttes af et program, skal den åbnes. Ved åbningen informeres filsystemet om, at filen ønskes benyttet til at læse data fra og/eller skrive data på. I afsnit 7.1 er givet en oversigt over brugen af OPEN-sætninger.

Syntaks

OPEN-sætningen har følgende opbygning:



BLOKSTØRRELSE, FILNUMMER:

FILNAVN:

File og '#' kan bruges i flæng. Ved programudskrifter konverteres '#' til FILE.

Udførelse

Ved udførelse af OPEN-sætningen åbnes filen 'filnavn' til læsning (READ) eller skrivning (WRITE), eller læsning/skrivning (RANDOM). Filen får tildelt nummeret 'filnummer', som benyttes til at identificere filen i lukke-, læse- og skrive-sætninger. Er 'filnummer' ikke heltallig, foretages en afrunding.

Følgende regler er gældende i forbindelse med OPEN-sætningen:

1. De tilladte filnumre er 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9.

2. Der kan være op til 8 filer åbne ad gangen. Et filnavn, der er åbnet for READ, kan dog ikke samtidigt være åbnet for WRITE. En særlig fil kan være åbnet ved hjælp af SELECT OUTPUT-sætningen. Hvis dette er tilfældet, vil alle PRINT (USING) uden FILE-angivelse gå til denne fil. En RANDOM-fil er samtidigt åben for både læsning og skrivning.

3. Ved OPEN...READ og OPEN...WRITE vil læsning/skrivning starte forfra i filen.

4. Ved OPEN-WRITE vil den pågældende fil blive dannet. Hvis filnavnet dog findes i forvejen, vil det resultere i en fejl. Sådanne filer kan der altså ikke tilføjes i.

5. Ved OPEN...APPEND åbner en allerede eksisterende fil til skrivning, startende efter det sidst skrevne. Det kan således lade sig gøre at tilføje til en allerede eksisterende fil. Efter filen er åbnet, kan der skrives til den efter de samme regler som ved OPEN...WRITE.

6. RANDOM-filer kan der godt tilføjes i, hvis der ellers er plads på disketten. Første gang den åbnes, bliver den dannet automatisk. Siden hen kan den meget vel være fysisk placeret på vidt forskellige områder på disketten. Der henvises i øvrigt til det særlige afsnit om RANDOM-filer.

Kommentarer

1. Filnavne, som de optræder i et katalog, har normalt en "udvidelse", der betegner filens type. Navnet:

   PROG1.CSB

   betegner f.eks. filen PROG1, der er et COMAL-program (CSB). Hvis filnavnet ikke indeholder nogen "udvidelse", vil DAT (for data) være underforstået.

2. Et af formålene med en OPEN-sætning er at etablere en midlertidig sammenhæng mellem et navn og et nummer. Når dette er sket, kan man nøjes med en simpel reference til et nummer i stedet for en omstændelig reference til et navn hver gang, man skal skrive/læse i filen.

   Hvis man en anden gang skal have adgang til filen, kan den sagtens åbnes med et helt andet nummer, som derefter vil være gældende, indtil filen lukkes igen.

3. Bemærk ved studiet af oversigten i afsnit 7.1, at nøgleordene READ, WRITE og RANDOM bruges i en række forskellige sammenhænge.

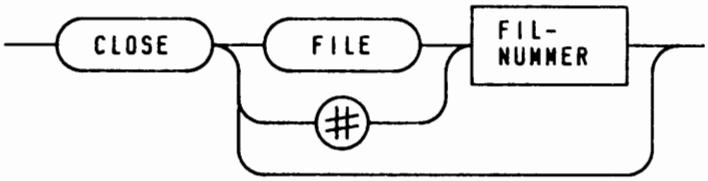
   Specielt skal man huske, at både OPEN...WRITE, OPEN...APPEND og OPEN...RANDOM kan bruges til både PRINT FILE (USING)... og WRITE FIL..., og at både OPEN...READ og OPEN...RANDOM kan bruges til både INPUT FILE... og READ FILE...

7.10 CLOSE

Når et program er færdig med at benytte en fil, skal den lukkes. Først når filen er lukket, kan en anden fil åbnes med samme nummer.

Syntaks

CLOSE-sætningen har følgende opbygning:



FILNUMMER:

Udførelse

Består CLOSE-sætningen kun af nøgleordet 'CLOSE', bevirker udførelsen, at alle åbne filer lukkes. Ellers medfører udførelsen af en CLOSE-sætning, at filen, som ved åbningen er blevet tildelt nummeret 'filnummer', lukkes. Er filnummeret ikke heltalligt, foretages en afrunding.

'#' og 'FILE' kan benyttes i flæng efter behag. I programudskrifter konverteres '#' til 'FILE'.

Eksempler

```
CLOSE #3
CLOSE FILE 0
CLOSE
```

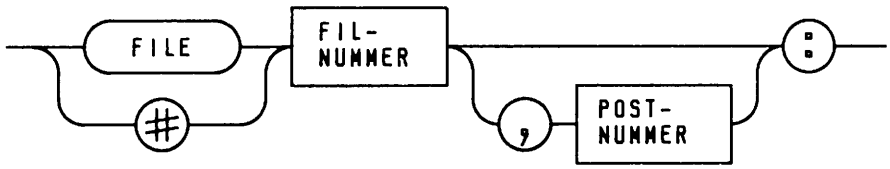
7.11 PRINT-FILE (USING)

PRINT-FILE (USING)-sætningen benyttes til at udskrive data på ASCII-form på en fil.

Syntaks

PRINT-FILE (USING)-sætningens opbygning er identisk med PRINT--sætningens respektive PRINT USING-sætningens, bortset fra, at 'PRINT' skal efterfølges af en fil-identifikation med følgende opbygning:

FILIDENTIFIKATION:



FILNUMMER, POSTNUMMER:

Syntaksen for PRINT (USING)-sætningen kan ses i afsnittene 5.5.3 og 5.5.4.

Udførelse

Angivelse af 'postnummer' har tilknytning til anvendelse af filer med direkte (random) tilgang, hvilket igen er knyttet til disketter. Anvendelse af filer med direkte tilgang vil blive beskrevet senere i afsnit 7.15.

Udførelse af en PRINT-FILE-sætning er helt analog med udførelse af en PRINT-sætning, bortset fra, at data udskrives på en fil i stedet for en skærm eller printer. Data udskrives på filen, som ved åbningen er blevet tildelt nummeret 'filnummer'. Er 'filnummer' ikke heltallig, foretages en afrunding.

'#' og 'FILE' kan benyttes i flæng efter behag. I programudskrifter konverteres '#' til 'FILE'.

Der henvises til afsnittene 5.5.3 og 5.5.4, hvor henholdsvis PRINT-USING-sætningen og PRINT-sætningen er beskrevet.

Kommentarer

1. Filen skal være åbnet til skrivning (WRITE).
   Ved direkte tilgang skal den dog være åbnet til læsning/-skrivning (RANDOM; se iøvrigt afsnit 7.15).

```
0010 ZONE:=0
0020 OPEN FILE 1, "DATAFIL", WRITE
0030 FOR I:=1 TO 10 DO
0040   FOR J:=1 TO 5 DO
0050     PRINT FILE 1: J,
0060   NEXT J
0070   PRINT FILE 1:
0080 NEXT I
0090 CLOSE FILE 1
0100 END
```

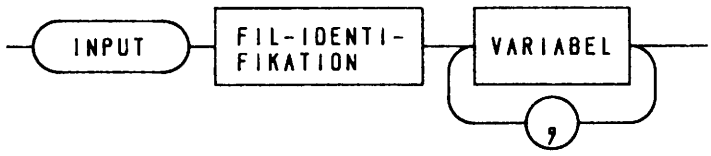
Udførelse af ovenstående program bevirker, at en fil med navnet DATAFIL dannes og åbnes til skrivning. Herefter udskrives 10 "linier", hver indeholdende cifrene 1, 2, 3, 4 og 5 skrevet umiddelbart efter hinanden.

7.12 INPUT-FILE

INPUT-FILE-sætningen benyttes til at indlæse data på ASCII-form, altså karakterform, fra en fil. (Normalt fra en fil, der er skrevet med PRINT FILE...(USING)).

Syntaks

INPUT-FILE-sætningen har følgende opbygning:



Fil-identifikation: Se forrige afsnit.

Udførelse

Med hensyn til betydningen af 'postnummer' i 'fil-identifikation' henvises til afsnit 7.15 om RANDOM-filer.

Udførelse af en INPUT-file-sætning er helt analog med udførelse af en INPUT-sætning bortset fra, at data læses fra en fil og ikke fra tastaturet. Data læses fra filen, som ved åbningen ikke er blevet tildelt nummeret 'filnummer'. Er 'filnummer' ikke heltallig, foretages en afrunding.

Læsning af en fil foregår sekventielt "linie" for "linie" startende med filens første "linie". Når en INPUT-FILE-sætning mødes under programudførelsen, gennemløbes listen af variable i sætningen fra venstre mod højre, og de tilsvarende værdier fra filens aktuelle "linie" tildeles.
'#' og 'FILE' kan benyttes i flæng i fil-identifikationen. I programudskrifter konverteres '#' til 'FILE'.

Kommentarer

1. Filen skal være åbnet til læsning (READ) eller til læsning-/skrivning (RANDOM).

Se kommentarerne til INPUT-sætningen i afsnit 5.5.5.

```
0010 OPEN FILE 1, "DATAFIL", READ
0020 FOR I:=1 TO 10 DO
0030   INPUT FILE 1: TAL
0040   PRINT TAL
0050 NEXT I
0060 CLOSE FILE 1
0070 END
```

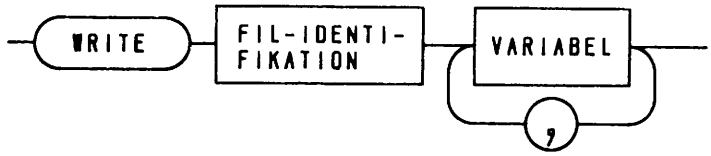
Udføres ovenstående program efter, at programmet i afsnit 7.11 er blevet udført, fås en udskrift bestående af 10 linier, hver indeholdende tallet 12345.

7.13 WRITE FILE

WRITE FILE-sætningen benyttes til at udskrive data på binær form i en fil.

Syntaks

WRITE FILE-sætningen har følgende form:



Variabelnavn kan angive enhver variabel bortset fra ZONE. Man kan altså både bruge simple, indicerede- og arrayvariable, og disse kan være af typen reel, heltal eller streng. Læg mærke til, at man ikke som ved PRINT... kan udskrive egentlige udtryk. Til gengæld kan man udskrive et helt array ad gangen.

Udførelse

Værdierne af de 'Variabelnavne', der er i sætningen, udskrives i en og samme post.

For serielle filer skrives der i næste post.

For RANDOM-filer skrives der i den ved NUM 2 udpegede post.

Værdierne udskrives i rækkefølge i binær form inden for posten. Heltal fylder 2 bytes (karakterer).

Reelle tal fylder 4 bytes ved 7-cifret præcision og 8 bytes ved 13-cifret præcision.

Strenge fylder hver 2 bytes + strengens længde i karakterer.

Hvis der altså udskrives A#, B$, hvor A# er en simpel heltalsvariabel og B$ er et 2 dimensionalt strengarray med 2*3 elementer, vil der blive udskrevet værdien af

A=B$(1,1) B$(1,2) B$(1,3) B$(2,1) B$(2,2) og B$(2,3)

i rækkefølge.

Hvis elementerne har længden 3, vil det fylde:

2 + 6 x (2+3) = 38 bytes

Ved en senere READ FILE-sætning på analog form vil man kunne indlæse værdierne igen i samme rækkefølge.

Ved PRINT FILE-sætninger gælder de samme regler for hvor meget, det fylder. Man skal blot huske, at PRINT altid udskriver karakterer.

Kommentarer

Den binære form er normalt hurtigere og mere pladsbesparende end karakterform.

Det kan være en fordel, hvor der er tale om større numeriske datamængder.

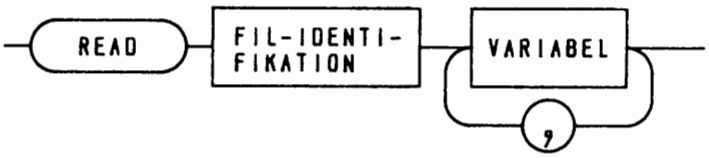
```
0010 DIM B$(2,3) OF 3
0020 MAT B$:="ABC"
0030 OPEN FILE 1, "DATA1", WRITE
0040 FOR I:=1 TO 4 DO
0050   WRITE FILE 1: I, B$
0060 NEXT I
0070 CLOSE FILE 1
0080 END
```

Dette program vil udskrive 4 poster, hver indeholdende et tal og 6 udgaver af "ABC".

7.14 READ FILE

READ FILE-sætningen bruges til at indlæse data på binær form fra en fil, der normalt vil være udskrevet med WRITE FILE.

Syntaks



Udførelse

Posten læses, og værdierne tildeles i rækkefølge til de anførte variable.

Eksempel

```
0010 DIM B$(6) OF 3
0020 OPEN FILE 0, "DATA1", READ
0030 FOR I:=1 TO 4 DO
0040   READ FILE 0: A, B$
0050   PRINT A;
0060   FOR X:=1 TO 6 DO
0070     PRINT B$(X);
0080   NEXT X
0085   PRINT
0090 NEXT I
```

```
0100 CLOSE FILE 0
0110 END
```

Hvis "DATA1" er den fil, der blev skrevet med eksemplet under WRITE FILE, vil udførelse af programmet give følgende resultat:

Værdierne i post nr. 2 er anbragt i A# og B$, d.v.s. A#=5, og de 6 elementer i B$ indeholder "ABC" på de første 3 pladser.

7.15 FILER AF TYPEN RANDOM

Disse kan kun anvendes med disketteversioner af COMET'en.

Kommentarer

1. Før brugen af en sådan fil skal den åbnes med en 'OPEN...RANDOM, NUM' sætning.
   'NUM' er et numerisk udtryk, der efter evt. afrunding angiver størrelsen i bytes af en post. Den nødvendige størrelse af en post kan udregnes efter de regler, som er angiver under WRITE FILE.

2. Der kan i flæng læses og skrives til en RANDOM-fil ved anvendelse af READ FILE..., INPUT FILE..., WRITE FILE... eller PRINT FILE...(USING).

3. Ved læsning/skrivning i en RANDOM-fil skal der altid angives postnummer efter filnummeret og adskilt fra dette med et komma.

4. Første gang filen åbnes, bliver den skabt automatisk. Den kan åbnes og lukkes igen og igen, men skal altid have samme post-længde.

Eksempel

```
0010 OPEN FILE 7, "RAN", RANDOM , 30
0020 DIM A$(2,3) OF 3
0030 DIM B$(3,2) OF 3
0040 MAT A$:="C80"
0050 WRITE FILE 7, 17: A$
0060 READ FILE 7, 17: B$
0070 CLOSE FILE 7
0080 PRINT B$(2,2)
0090 END
```

Dette program vil overføre indholdet af A$ til den (anderledes dimensionerede) B$ og udskrive værdien "C80" på skærmen.

7.16 EOF

EOF er en systemvariabel, der kan bruges ved læsning af serielle filer til at afgøre, om man har læst sidste post i filen.

EOF er en heltals-vektorvariabel med 10 elementer.

EOF(n) sættes = 0 ved åbning af fil nr. n.

Når sidste post i filen er læst, sættes EOF(n) = 1 (eller 'sand').

Eksempel

```
0010 OPEN FILE 2, "DATA2", READ
0020 WHILE NOT EOF(2) DO
0030   INPUT FILE 2: TAL
0040   PRINT TAL;
0050 ENDWHILE
0060 CLOSE FILE 2
0070 END
```

Dette program vil indlæse samtlige poster (i dette tilfælde reelle tal) fra filen, DATA2, og udskrive dem efter hinanden på skærmen.

7.17 BRUG AF FILER I CP/M FORMAT

Der findes to "switche" ved hjælp af hvilke, man kan bestemme, om de af COMAL-80 producerede diskettefiler skal være i COMAL-80 format eller CP/M format. Det har betydning, hvis man i COMAL-80 ønsker at udveksle data via diskette med andre CP/M programmer end COMAL-80 fortolkeren.

Hvis man anfører filnavne som beskrevet i COMAL-80 manualen oprettes disse ved skrivning i COMAL-80 format. Tilsvarende forventes det, at filer anført på denne måde i forbindelse med læsning, tidligere er oprettet i COMAL-80 format.

Ønskes en fil oprettet og skrevet i CP/M format, anføres efter filnavnet:

/C for ASCII-filer
/C/B for binære filer

Dette gælder såvel programfiler som sekventielle datafiler; d.v.s. filer åbnet ved OPEN .... WRITE. For RANDOM-filer er det ikke påkrævet at anvende switchene.

En fil oprettet som ovenfor beskrevet kan også behandles af programmer, der ikke er skrevet i COMAL-80; f.eks. CP/M programmer eller andre assembler programmer.

Ønsker man i COMAL-80 at anvende (læse) en fil oprettet i CP/M format (inklusive filer skrevet udenfor COMAL-80), skal man anføre switche i forbindelse med filnavnene på samme måde, som beskrevet ovenfor.

EKSEMPLER

a. Ønsker man at gemme et COMAL-80 program ved hjælp af LIST i CP/M format under navnet MITPROG, skrives:

```
LIST MITPROG/C   (LIST opretter en ASCII-fil)
```

Programmet indlæses således: ENTER MITPROG/C

b. Ønsker man i stedet at gemme ovenstående program ved hjælp af SAVE, men stadig i CP/M format, skrives:

```
SAVE MITPROG/C/B  (SAVE opretter en binær fil)
```

Programmet indlæses således: LOAD MITPROG/C/B

c. Switchene har størst betydning i forbindelse med datafiler.

Hvis man i et COMAL-80 program skal udskrive data til en fil ved navn KATALOG i CP/M format, åbnes denne fil ved:

(i) OPEN FILE 0, "KATALOG/C", WRITE

hvis man ønsker at skrive til filen med PRINT FILE,

eller

(ii) OPEN FILE 0, "KATALOG/C/B", WRITE

hvis man ønsker at skrive til filen med WRITE FILE.

d. Skal man læse fra filen c.i, åbnes denne ved:

```
OPEN FILE 0, "KATALOG/C", READ
```

og læsning sker ved INPUT FILE.

Skal man læse fra filen c.ii, åbnes med:

```
OPEN FILE 0, "KATALOG/C/B"
```

og READ FILE anvendes til læsning.

Til slut et resume over de programsætninger i forbindelse med hvilke, man kan benytte switchene:

Nøgleord	Switch	Senere brug	Filtype
LIST	/C	ENTER	programfil ASCII-format (.CML)
ENTER	/C	RUN el.lign.	programfil ASCII-format (.CML)
SAVE	/C/B	LOAD	programfil binær format (.CSB)
LOAD	/C/B	RUN el.lign.	programfil binær format (.CSB)
OPEN	/C	PRINT FILE	datafil ASCII-format (.DAT)
OPEN	/C	INPUT FILE	datafil ASCII-format (.DAT)
OPEN	/C/B	WRITE FILE	datafil binær format (.DAT)
OPEN	/C/B	READ FILE	datafil binær format (.DAT)

En sekvens, hvor man først skriver til en binær datafil i CP/M format, og dernæst læser den samme fil, kan være opbygget som følger:

```
0010 DIM TEKST$(100) OF 20
0020 // Her tildeles værdier til TEKST$
0030 OPEN FILE 0, "KATALOG/C/B", WRITE
0040 FOR POSTNR:=1 TO 100 DO
0050   WRITE FILE 0: TEKST$(POSTNR)
0060 NEXT POSTNR
0070 CLOSE FILE 0
0080 //
0090 OPEN FILE 0, "KATALOG/C/B", READ
0100 FOR POSTNR:=1 TO 100 DO
0110   READ FILE 0: TEKST$(POSTNR)
0120 NEXT POSTNR
0130 CLOSE FILE 0
0140 END
```

7.18 NOGLE RÅD TIL BEGYNDERE

* Opstår der tvivl, om der skulle være åbne datakanaler, kan 2 fremgangsmåder benyttes:

  A: Kommandoen CLOSE lukker alle åbne kanaler - men der er jo ikke sikkert, at dette er ønskeligt.

  B: Kommandoen RELEASE fortæller brugeren, **hvilke** åbne kanaler, der findes. Brugeren kan derefter foretage det fornødne.

* Benyt som kommando eller i en programsætning instruktionen

  INIT enhed$

  hver gang en diskette udskiftes. Er man ikke sikker på, om den aktuelle enhed er DK0 eller DK1, kan man få oplysningen med instruktionen GETUNIT.

* Dersom man ved læsning i en RANDOM-fil ikke kan huske postlængden (blokstørrelsen), kan følgende program give svaret:

```
0010 DIM FILNAVN$ OF 12
0020 INPUT "Filnavn : ": FILNAVN$
0030 // Her kan evt tilføjes diverse kontroller
0040 OPEN FILE 0, FILNAVN$, READ
0050 READ FILE 0: BLOK#
0060 CLOSE FILE 0
0070 //
0080 PRINT "Blokstørrelsen for ";FILNAVN$;" er ";BLOK#
0090 END
```

8 Systemkommandoer

INDHOLDSFORTEGNELSE KAPITEL 8 - SYSTEMKOMMANDOER

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

8.1 KOMMANDOER OG DERES ANVENDELSE

Hver kommandolinie afsluttes med at aktivere RETURN-tasten. Kommando mærket med * kan ikke indgå i programsætninger.

KOMMANDO	BESKRIVELSE
AUTO	Kan benyttes på flere måder:
	- AUTO — Første linienummer er 0010, og de næste numre forøges med 10.
	- AUTO nnnn — Første linienr. er nnnn, og de næste numre forøges med 10.
	- AUTO nnnn,n — Første linienr. er nnnn, og de næste numre forøges med n.
CALL nnnnn	Starter afvikling af rutine i maskinkode fra adresse nnnnn (decimalt).
CAT	Kan benyttes på følgende måder:
	- CAT — Udskriver katalog (directory) fra default drev.
	- CAT <enhed> — Udskriver katalog fra valgt <enhed>.
	- CAT *.<type> — Udskriver kun filer af valgt <type>.
	- CAT D??A.* — Udskriver kun filer, hvori det første af 4 tegn er D, og det sidste er A. Alle typer (extensions) medtages.
	- CAT ...,LP: — Sender katalog til printer. Man kan efter hver af ovenstående eksempler tilføje ,LP:
CLEAR	Sletter skærm og anbringer cursor (markør) i øverste venstre hjørne.
CLOSE	Benyttes i forskellige situationer:
	- CLOSE — Lukker alle åbne datafiler.
	- CLOSE <filnr> — Lukker for <filnr>.

KOMMANDO	BESKRIVELSE
CON nnnn	Fortsætter afviklingen af et program fra linie nnnn.
CURSOR	Kan benyttes som i programsætninger.
DEL	Benyttes til at slette linienumre:
	- DEL <nnnn> Sletter linie nnnn
	- DEL <nnnn>, Sletter fra og med linie nnnn og rest af program
	- DEL ,<nnnn> Sletter fra start til og med linie nnnn
	- DEL <fra>,<til> Sletter alle linier fra og med <fra> til og med <til>
DELETE	Benyttes til at slette (eksisterende) filer:
	- DELETE <navn.type> Sletter denne fil på default enhed
	- DELETE <enhed:navn.type> Sletter denne fil fra valgt enhed
	Alle filer kan vælges som under CAT.
EDIT	Benyttes til at editere en eller flere programlinier.
	- EDIT Alle linier kan editeres
	- EDIT <nnnn> Linie nnnn kan editeres
	- EDIT <fra>,<til> Alle linienumre fra og med <fra> til og med <til> editeres
	- EDIT ,<nnnn> Fra første linie til nnnn kan editeres
	- EDIT <nnnn>, Fra linie <nnnn> og rest af program kan editeres
	BEMÆRK: Også selve linienummeret kan ændres. Dog vil det gamle linienummer bevares.

KOMMANDO	BESKRIVELSE
ENTER	Indlæser en fil i ASCII-format (gemt med kommandoen LIST). Dersom filen er indlæst (lagret på baggrundslageret) ved kommandoen: LIST <filnavn/c> skal kommandoen hedde: ENTER <filnavn/c> Hvis filen har typen 'CML' er det ikke nødvendigt at inkludere denne i filnavnet.
ESC	Er ikke en egentlig kommando, men en tast.
IDENTIFIERLOWER	Kan være 0 (store) eller 1 (små) bogstaver i variabelnavne. Vælges små bogstaver, skrives IDENTIFIERLOWER:=1
INDENTION	Angiver antal positioners indrykning i udskrifter. Eks.: INDENTION:=4
INIT	Benyttes til at 'resette' disksystemet, så man kan udskrive data til nyindsat diskette. Man kan også benytte kommandoen INIT <enhed>. Eks.: INIT INIT DK1:
KEYWORDLOWER	Kan med samme værdier som ved IDENTIFIERLOWER påvirke nøgleord.
LIST	Benyttes i følgende sammenhænge: 1. LAGRING AF PROGRAM I ASCII-format: Ved at benytte LIST <filnavn> i stedet for SAVE <filnavn> kan det lade sig gøre at overføre programmer fra en COMAL-80 version til en anden. Vedrørende indlæsning af "LISTede" programmer se ENTER.

KOMMANDO BESKRIVELSE

LIST forts. Kommando: LIST <enhed:filnavn>

<filnavn> udlæses til den valgte enhed, men ikke i CP/M format. For at opnå CP/M format, skal kommandoen være:

LIST <enhed:filnavn/c>

Udelades <enhed>, udskrives til defaultenheden. Hvis der ikke specificeres nogen type på filnavnet, vil filen få type CML.

Vælges CP/M formatet, vil filen f.eks. kunne behandles i et tekstbehandlingssystem/Editor.

2. Listning af del af eller hele programmet.

- LIST	Alle linier listes fra start med 23 linier ad gangen ved tryk på en vilkårlig tast. Trykkes i stedet RETURN, tilføjes kun en linie ad gangen.
- LIST <nnnn>	Listning af linie nnnn
- LIST <fra>,<til>	Alle linienumre fra og med <fra> til og med <til> kan listes.
- LIST ,<nnnn>	Fra første linie til nnnn listes.
- LIST <nnnn>,	Fra linie <nnnn> og rest af program listes.
- ... LP:	Udskrift på printer. Dersom ovenstående kommandoer afsluttes med 'LP:' vil udskriften sendes til printeren.

I de situationer, hvor en 'listning' ønskes sendt til en printer, vil fortolkeren meddele, om printeren er tilsluttet.

LOAD Indlæser et program på binær form fra baggrundslager. Hvis der ikke specificeres nogen type i filnavnet, vil CSB vælges.

LOGOFF Afbryder aktiviteten af LOGON.

KOMMANDO	BESKRIVELSE
LOGON <filnavn>	Bevirker, at alt, der udskrives på skærm, udlæses til <filnavn>.
PAGE	Bevirker, at printeren udfører en såkaldt 'form-feed', d.v.s. sender papiret til 'første linie på næste side'. Det forudsættes, at man ikke manuelt har ført papiret frem eller tilbage.
PAGELENGTH	Angiver sidelængden i antal linier på printer.
PAGEWIDTH	Angiver sidebredden i antal tegn på printer.
PEEK/POKE	Direkte tilgang til lageret. Se i øvrigt afsnit 6.5.
PRINT	Kan i en kommandolinie benyttes således: PRINT "Dette er en tekst" eller PRINT <variabel> Forudsat variablen under en kørsel har fået tildelt en værdi.
QUIT	Er kommandoen, hvorved man fra COMAL-80 returnerer til CP/M.
RELEASE	Anvendes til at undersøge, om der findes åbne filer på en enhed.
RENAME	Efterfølges af <enhed>:<eksist.typ> , <enhed>:<nyfil.typ> Herved omdøbes filen <eksist.typ> til det navn, der svarer til <nyfil.typ>. Dersom <enhed> ikke angives, forudsættes det, at filerne ligger på defaultenheden.
RENUM	Kan anvendes på flere måder: - RENUM Alle linienumre organiseres med spring på 10 startende med linienummer 0010. - RENUM nnnn Første linienummer er nnnn, og der springes med 10.

KOMMANDO	BESKRIVELSE	
RENUM forts.	- RENUM nnnn,x	Første linienummer er nnnn, og der springes med x.
	- RENUM aa:aa,c	Linienummer aa flyttes til c, dersom en programlinie med nummer c ikke findes i forvejen.
	- RENUM a:b,c,d	Linierne fra og med a til og med b flyttes, så det nye første linienummer bliver c, og springet skal være med d. Resummeringen udføres kun, dersom eksisterende linienumre i det interval ikke overskrives.
RENUMBER	Svarer til RENUM.	
RUN	Kan bruges på 2 måder:	
	- RUN	Starter eksekvering af program.
	- RUN nnnn	Starter eksekvering af program fra linie nnnn. Dersom der i denne del optræder variable, som ikke har fået tildelt en værdi, stopper programmet med fejlmelding om ikke defineret/-definerede variable.
SIZE	Angiver følgende:	
	Bogstaverne nn er symbol for den decimale angivelse af antal bytes.	
	Pladsforbrug: nn "Extensions": nn Program: nn Variable: nn Tilbage: nn	
UNIT	Benyttes til at udvælge det drev, som skal være "default".	
ZONE	Angiver bredde i skrivefelt.	

8.2 SPECIALANVENDELSE UNDER KOMMANDO MODE

Der er intet i vejen for, at man i en kommandolinie tildeler værdier til en eller flere simple numeriske variable. Man kan herefter benytte de aritmetiske og logiske funktioner samt visse systemfunktioner til beregninger.

Forekommer der eksempelvis i en programlinie instruktionen STOP, kan man tildele en eksisterende variabel af en hvilken som helst type en ny værdi, hvorefter man kan anvende kommandoen CON.

Tilsvarende kan man ændre værdien af definerede variable, dersom man kan afbryde programafviklingen ved at aktivere ESC-tasten (TRAP ESC- ikke aktiv). Det vil her ofte være det rimeligste at fortsætte programafviklingen med kommandoen CON.

8.3 UDDYBENDE BESKRIVELSE AF UDVALGTE KOMMANDOER

8.3.1 IND- OG UDLÆSNING AF PROGRAMMER

Et COMAL-80 program kan lagres som enten en ASCII-fil eller en såkaldt BINÆR-fil. Binære filer fylder ikke så meget på disketten som ASCII-filer. Men COMAL-80's 13-cifres version kan ikke indlæse en binær fil, der er dannet med 7-cifres versionen, hvorimod det ikke er noget problem, hvis filen er i ASCII-format. Dersom man vil sikre sig, at nyttige programmer kan indlæses af tidligere og kommende COMAL-80 versioner, vil der i de fleste tilfælde være mulighed for at udnytte det meste af de således lagrede programmer.

Såfremt programmerne kun skal være tilgængelige for forskellige udgaver af COMAL-80, kan man ved lagring af et program benytte:

LIST <programnavn>	:	Typebetegnelsen .CML tilføjes automatisk, og programmet er lagret i det format, som kaldes ASCII-format.
SAVE <programnavn>	:	Typebetegnelsen .CSB tilføjes automatisk, og programmet er lagret i det format, som kaldes BINÆRT format.
ENTER <programnavn>	:	Skal benyttes, dersom typebetegnelsen er .CML
LOAD <programnavn>	:	Skal benyttes, dersom typebetegnelsen er .CSB

Såfremt man (af grunde, vi ikke her skal komme ind på) skulle foretrække programmer eller filer lagret i CP/M format, er der mulighed herfor:

LIST <programnavn>/C	Indlæses ved ENTER <programnavn>/C
SAVE <programnavn>/C/B	Indlæses ved ENTER <programnavn>/C/B

Ovenstående fremgangsmåde gælder også for sekventielle datafiler, d.v.s. filer, der er åbnet ved instruktionen OPEN...WRITE.

Tilsvarende hensyn er ikke påkrævet i forbindelse med radomiserede datafiler. Se om filbehandling i kapitel 7.

Dersom brugeren SAVEr eller LISTer et program med et navn, som

eksisterer i forvejen, får brugeren dette oplyst, idet fortolkeren giver brugeren mulighed for at annullere kommandoen ved at trykke ESC, hvorimod RETURN vil bevirke overskrivning af denne fil.

I forbindelse med udlæsning til datamediet bør man sikre sig på flere måder:

- Systemet skal være bekendt med katalogblokken (Directory), hvilket ikke er tilfældet, dersom man har monteret en ny diskette uden at give kommandoen INIT.

Eksempel

```
&p
```

9 Diverse tabeller

INDHOLDSFORTEGNELSE KAPITEL 9 - DIVERSE TABELLER

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

9. DIVERSE TABELLER

9.1 STIKORDSREGISTER

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

International Computers Limited a/s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk

9.2 RESERVEREDE NØGLEORD (ALFABETISK)

Dette er ord, som ifølge deres indhold ville kunne bruges som normale identifikatorer, men som er reserveret til andre formål.

ABS
AND
ATN
AUTO
BSTR
BVAL
CALL
CASE
CAT
CHAIN
CHR
CLEAR
CLOSE
CLOSED
CON
COS
CURSOR
DATA
DEF
DEL
DELETE
DIM
DIV
DO
DOWNTO
EDIT
ELIF
ELSE
END
ENDCASE
ENDDEF
ENDIF
ENDLOOP
ENDPROC
ENDWHILE
ENTER
EOD
EOF
ERR
ERRTEXT
ESC
EXEC
EXP
FALSE
FILE
FOR
FRAC
GETUNIT
GLOBAL
GOSUB
GOTO
IF
IN
INIT
INP
INPUT
INT
IVAL
LEN
LET
LIST
LOAD
LOG
LOOP
MAT
MOD
NEW
NOT
OF
ON
OPEN
OR
ORD
OTHERWISE
OUT
OUTPUT
PAGE
PEEK
POKE
POS
PRINT
PROC
QUIT
RANDOM
RANDOMIZE
READ
REF
RELEASE
REM
RENAME
RENUM
RENUMBER
REPEAT
RESTORE
RETURN
RND
ROUND
RUN
SAVE
SELECT
SGN
SIN
SIZE
SPC
SQR
STEP
STOP
STR
TAB
TAN
THEN
TO
TRAP
TRUE
TRUNC
UNIT
UNTIL
USING
VAL
VARPTR
WHEN
WHILE
WRITE
ZONE

Bemærk, at ord, der blot indeholder reserverede ord, ikke er reserverede; f.eks. FILE2, INDDATA eller FINIS.

9.3 RESERVEREDE ORD (RUBRICERET EFTER BRUG I HENHOLDSVIS SÆTNINGER OG KOMMANDOER

(Navne på standardfunktioner er ikke medtaget her)

Bruges kun i sætninger		Bruges som kommando i sætnings-format	Forskelligt format for sætninger og kommandoer	Bruges kun som kommando
AND	IN	CLEAR	CAT	AUTO
CALL	INPUT	CLOSE	DELETE	CON
CASE	LABEL	CURSOR	FORMAT	DEL
CHAIN	LOOP	(ERR)	GETUNIT	EDIT
CLOSED	MOD	(ESC)	INIT	ENTER
DATA	NEXT	(FILE)	RELEASE	LIST
DEF	NOT	LET	RENAME	LOAD
DIM	OF	MAT	UNIT	NEW
DIV	ON	OPEN		RENUM
DO	OR	(OUTPUT)		RENUMBER
DOWNTO	OTHERWISE	PAGE		RUN
ELIF	OUT	POKE		SAVE
ELSE	PEEK	PRINT		SIZE
END	PROC	QUIT		
ENDCASE	READ	(RANDOM)		
ENDDEF	REF	RANDOM		
ENDIF	REM	RANDOMIZE		
ENDLOOP	REPEAT	(READ)		
ENDPROC	RESTORE	SELECT		
ENDWHILE	RETURN	(TAB)		
EXEC	STEP	TRAP		
EXIT	STOP	(USING)		
FALSE	TAB	(WRITE)		
FN...	THEN	(ZONE)		
FOR	TO			
GLOBAL	TRUE			
GOSUB	UNTIL			
GOTO	WHEN			
IF	WHILE			
	WRITE			
	ZONE			

9.4 TABEL OVER ASCII-KARAKTERER

Decimal- og Hexadecimalværdier af ASCII-tegn

Talværdi: Dec.	Hex.	ASCII kode:	Talværdi: Dec.	Hex.	ASCII kode:	Talværdi: Dec.	Hex.	ASCII kode:	Talværdi: Dec.	Hex.	ASCII kode:
0	00	NUL	32	20	SPACE	64	40		96	60	
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	£	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	´	71	47	G	103	67	g
8	08	BS 1)	40	28	(	72	48	H	104	68	h
9	09	HT 2)	41	29	)	73	49	I	105	69	i
10	0A	LF 3)	42	2A	*	74	4A	J	106	6A	j
11	0B	VT 4)	43	2B	+	75	4B	K	107	6B	k
12	0C	FF 5)	44	2C	,	76	4C	L	108	6C	l
13	0D	CR 6)	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC 7)	59	3B	;	91	5B	Æ	123	7B	æ
28	1C	FS	60	3C	<	92	5C	Ø	124	7C	ø
29	1D	GS	61	3D	=	93	5D	Å	125	7D	å
30	1E	RS	62	3E	>	94	5E	^	126	7E	
31	1F	US 8)	63	3F	?	95	5F	_	127	7F	DEL 9)

Taster på tastatur: 1) Pil til venstre (<==)/<BS>
2) <TAB>
3) Pil ned
4) Pil op
5) Pil til højre (==>)
6) <RETURN>
7) <ESC>
8) <INS>
9) <DEL>

9.5 FEJLTEKSTER

Listen inkluderer også enkelte andre systemtekster.

```
 1 : Lagerplads opbrugt
 2 : Syntaksfejl
 3 : Overløb
 4 : Ikke S/# her
 5 : Kun for strenge
 6 : Fejl i kommando
 7 : Ikke flere nye navne !
 8 : Ikke afsluttet streng
 9 : Ulovligt tegn
10 : Ulovligt tegn
11 : Ulovligt linienummer
12 : For lang linie
13 : Variabel forventet
14 : ')' forventet
15 : Typekonflikt
16 : For kompliceret udtryk
17 : '(' forventet
18 : Typekonflikt i parameter
19 : Har ingen parametre
20 : Forkert type
21 : ',' forventet
22 : TAB ikke tilladt her
23 : Operand forventet
24 : Konstant forventet
25 : ':' forventet
26 : Funktion ikke tilladt her
27 : :=/:+/:-/= brugt forkert
28 : :=/:+/:- forventet
29 : ':' ikke tilladt her
30 : 'FILE' forventet
31 : Linieslut her ?
32 : Ukendt I/U-enhed
33 : Et navn forventet
34 : Se manualen
35 : 'OF' forventet
36 : Ikke strengfunktion
37 : Linienummer forventet
38 : GOTO/GOSUB forventet
39 : Ikke efter 'THEN'
40 : Se manualen
41 : Tabel ikke tilladt
42 : TO/DOWNTO forventet
43 : READ/WRITE/APPEND/RANDOM forventet
44 : Fra >= til
45 : Linieslut forventet
46 : Programsætning forventet
47 : Kommando forventet
48 : Fejl i programstruktur
```

Fejltekster fortsat ...

49 : Typekonflikt
50 : Fejl i programstruktur
51 : Dobbeltdefineret
52 : Typekonflikt i RETURN-sætning
53 : Navnekonflikt med PROC/FUNC
54 : FOR-NEXT dybde
55 : Ukendt linienummer
56 : RESTORE: kun til DATA-sætning
57 : IF uden ENDIF
58 : CASE uden ENDCASE
59 : PROC uden ENDPROC
60 : FUNC uden ENDFUNC
61 : REPEAT uden UNTIL
62 : WHILE uden ENDWHILE
63 : FOR uden NEXT
64 : Ukendt PROC/FUNC/LABEL
65 : For kompliceret programstruktur
66 : Fejl ved udskrivning på log
67 : Indexfejl
68 : Ulovligt postnummer
69 : Ikke delstreng her
70 : For få indices
71 : For mange indices
72 : Ikke flere data
73 : Fejl i tildeling til delstreng
74 : Kun for arrays
75 : Fejl i USING-strengen
76 : Ulovlig TAB-værdi
77 : Variablen findes allerede
78 : Kan ikke returnere
79 : Kan ikke returnere (ulovlig GOTO/GOSUB)
80 : CASE-værdi findes ikke
81 : STEP = 0
82 : SYSTEMFEJL
83 : SYSTEMFEJL
84 : Ude af definitionsområdet
85 : For lang
86 : Overløb
87 : Udefineret variabel
88 : For lang
89 : Ikke kald af FUNC som kommando
90 : Indexfejl
91 : Typekonflikt i parameter
92 : For mange parametre
93 : For få parametre
94 : Division med 0
95 : SYSTEMFEJL
96 : Typekonflikt

Fejltekster fortsat ...

```
 97 : For lang linie
 98 : Ikke nu
 99 : Fejl i NEXT
100 : ':' ikke tilladt her
101 : Ingen linier med angivet nr.
102 : Umuligt
103 : Umuligt
104 : Umuligt
105 : Auto overløb
106 : Ikke nu
107 : SAVEd under uforligelig COMAL-version
108 : Tabeller skal have REF
109 : Parameteren skal være en variabel
110 : Parameteren har forkert dimension
111 : EXIT uden LOOP
112 : LOOP uden ENDLOOP
113 : Umuligt
114 : Umuligt
115 : Ikke nu
116 : PROC/FUNC skal være både kaldt og CLOSED
117 : Delstreng af FUNC-kald ikke tilladt
118 : Kan ikke kaldes fra en CLOSED PROC/FUNC
119 : Skal returnere med 'RETURN'
120 : Fejl i brug af "extension"
121 : Fejl i brug af "extension"
122 : Fejl i brug af "extension"
123 : Fejl i brug af "extension"
124 : Fejl i brug af "extension"
125 : Fejl i brug af "extension"
126 : Fejl i brug af "extension"
127 : Fejl i brug af "extension"
128 : Fejl i brug af "extension"
129 : Fejl i brug af "extension"
130 : Allerede eksisterende "extension"
131 : Manglende "extension"
132 : Bruger en "extension" forkert
133 : For mange "extensions"
134 : Ulovlig værdi for en systemvariabel
135 : For mange variable i RECEIVE
136 : Typekonflikt
137 : Fejl i initialiseringsfil
138 : Kanalen er allerede åben
139 : Kanalen er ikke åben
140 : Ulovligt kanalnummer
141 : Ukendt I/U-enhed
142 : Ukendt I/U-enhed
143 : Fejl i filnavn
144 : SYSTEMFEJL
```

Fejltekster fortsat ...

145 : SYSTEMFEJL
146 : SYSTEMFEJL
147 : Filtype ikke tilladt her
148 : Umuligt
149 : Umuligt
150 : Fejl i initialiseringsfil
151 : Kanalen er allerede åben
152 : Kanalen er ikke åben
153 : Ulovligt kanalnummer
154 : Ukendt I/U-enhed
155 : Ukendt I/U-enhed
156 : Fejl i filnavn
157 : SYSTEMFEJL
158 : SYSTEMFEJL
159 : SYSTEMFEJL
160 : Filtype ikke tilladt her
161 : Umuligt
162 : Umuligt
163 : SYSTEMFEJL
164 : Kan ikke skrive
165 : Kan ikke læse
166 : Allerede åben i en anden måde
167 : Filen er i brug
168 : SYSTEMFEJL
169 : Der kan ikke åbnes flere diskfiler nu
170 : Filen findes ikke
171 : SYSTEMFEJL
172 : SYSTEMFEJL
173 : SYSTEMFEJL
174 : Umuligt: en fil er åben
175 : SYSTEMFEJL
176 : Simpel I/U-enhed
177 : SYSTEMFEJL
178 : SYSTEMFEJL
179 : SYSTEMFEJL
180 : Filkataloget er fuldt
181 : Disken eller filen er fuld
182 : SYSTEMFEJL
183 : Forkert brug af filen
184 : "End-Of-File"
185 : Fejl ved lukning
186 : SYSTEMFEJL
187 : Forkert længde
188 : SYSTEMFEJL
189 : Læsefejl
190 : SYSTEMFEJL
191 : SYSTEMFEJL
192 : SYSTEMFEJL

Fejltekster fortsat ...

193 : SYSTEMFEJL
194 : SYSTEMFEJL
195 : SYSTEMFEJL
196 : SYSTEMFEJL
197 : SYSTEMFEJL
198 : SYSTEMFEJL
199 : SYSTEMFEJL
200 : Fejl i initialiseringsfil
201 : Kanalen er allerede åben
202 : Kanalen er ikke åben
203 : Ulovligt kanalnummer
204 : Ukendt I/U-enhed
205 : Ukendt I/U-enhed
206 : Fejl i filnavn
207 : SYSTEMFEJL
208 : SYSTEMFEJL
209 : SYSTEMFEJL
210 : Filtype ikke tilladt her
211 : Umuligt
212 : Umuligt
213 : SYSTEMFEJL
214 : Kan ikke skrive
215 : Kan ikke læse
216 : Allerede åben i en anden måde
217 : Filen er i brug
218 : SYSTEMFEJL
219 : Der kan ikke åbnes flere diskfiler nu
220 : Filen findes ikke
221 : SYSTEMFEJL
222 : SYSTEMFEJL
223 : SYSTEMFEJL
224 : Umuligt: en fil er åben
225 : SYSTEMFEJL
226 : Simpel I/U-enhed
227 : SYSTEMFEJL
228 : SYSTEMFEJL
229 : SYSTEMFEJL
230 : Filkataloget er fuldt
231 : Disken eller filen er fuld
232 : SYSTEMFEJL
233 : Forkert brug af filen
234 : "End-Of-File"
235 : Fejl ved lukning
236 : SYSTEMFEJL
237 : Forkert længde
238 : SYSTEMFEJL
239 : Læsefejl
240 : SYSTEMFEJL

Fejltekster fortsat ...

241 : SYSTEMFEJL
242 : SYSTEMFEJL
243 : SYSTEMFEJL
244 : SYSTEMFEJL
245 : SYSTEMFEJL
246 : SYSTEMFEJL
247 : SYSTEMFEJL
248 : SYSTEMFEJL
249 : SYSTEMFEJL
250 : SYSTEMFEJL
251 : SYSTEMFEJL
252 : SYSTEMFEJL
253 : SYSTEMFEJL
254 : SYSTEMFEJL
255 : SYSTEMFEJL
256 : SYSTEMFEJL
257 : SYSTEMFEJL
258 : Blokken er overskredet
259 : Ulovlig bloklængde
260 : Det er ikke en RANDOM-fil
261 : Forkert bloklængde
262 : Filen findes allerede
263 : SYSTEMFEJL
264 : SYSTEMFEJL
265 : Fejl i filnavn
266 : Forskellige I/U-enheder er angivet
267 : SYSTEMFEJL
268 : SYSTEMFEJL
269 : SYSTEMFEJL
270 : SYSTEMFEJL
271 : SYSTEMFEJL
272 : SYSTEMFEJL
273 : SYSTEMFEJL
274 : SYSTEMFEJL
275 : SYSTEMFEJL
276 : SYSTEMFEJL
277 : SYSTEMFEJL
278 : SYSTEMFEJL
279 : SYSTEMFEJL
280 : SYSTEMFEJL
281 : SYSTEMFEJL
282 : SYSTEMFEJL
283 : SYSTEMFEJL

10 Private notater

INDHOLDSFORTEGNELSE KAPITEL 10 - PRIVATE NOTATER SAMT MATERIALE I STØRRE FORMAT END A4

Afsnit Side

International Computers Limited a s

Hovedkontor: Klampenborgvej 232, 2800 Lyngby. Tlf. 02-88 94 88. Giro 5 45 30 11
Reg. nr. 40572. Telex 39147 iclcph dk. Telegram Computel, København

Jylland og Fyn: Romancevej 9,8700 Horsens. Telefon 05-62 75 88
Teknisk service og uddannelsesafdeling:
Bådehavnsgade 10, 2450 København SV. Telefon 01-16 55 88
Telex 22451 iclts dk