Min utrolige

AMSTRAD

Min utrolige

AMSTRAD

AF

MOGENS LARSEN

2. oplag, 1984

INDHOLDSFORTEGNELSE!

FORORD.

Denne bog er skrevet, for at du som AMSTRAD bruger kan få den fulde glæde, ved brugen af denne utrolige computer.
Den er skrevet således, at du kan udnytte din computer helt, uden at have kendskab til EDB.
AMSTRAD er ikke kun til spil, du kan med denne bog selv lære at programmere, du kan også købe professionelle programmer, så som tekstbehandling, kalkulation og undervisningsprogrammer.
Du kan også købe andre programmeringssprog til din AMSTRAD end lige BASIC.
Vi håber, med denne bog at kunne hjælpe dig og dine venner ind i nogle nye og helt utrolige oplevelser.
Held og lykke med din nye AMSTRAD.

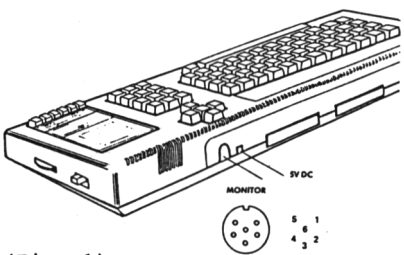
Mogens Garbenfeldt
Direktør

1.1 OPSTILLING.

Amstrad CPC464 leveres med flg. skærmmuligheder:

1) Amstrad GT64	- grøn monocrom skærm
2) Amstrad CTM640	- farvemonitor
3) Amstrad MP1	- modulator for tilslutning til farve- eller S/H-TV apparat (indstilles på UHF-kanal 36)

Sørg for, at monitor/modulator og tangentbords afbrydere står i afbrudt tilstand (på monitor afbryderknap i yderste stilling og på tangentbord på OFF). Tilslut kabler fra monitor/modulator til tangentbordet, henholdsvis 5V DC og MONITOR (fig. 1). Tænd for monitor og for tangentbordet, hvorefter skærmen vil vise billedet side F1.2 i den engelske manual.



(Fig. 1).

Af hensyn til computeren er det vigtigt, at strømmen på monitor ell. modulator afbrydes, når

1) monitor/modulator tilsluttes
2) printer/diskettedrev tilsluttes
3) iøvrigt alle ydre tilslutninger skiftes

Vær iøvrigt opmærksom på, at hvis computeren kun afbrydes på tangentbordet, vil monitor/TV få sort skærm, men der er stadig spænding på skærmen, så afbryd også denne, hvis computeren ikke skal bruges i et længere tidsrum.

Monitorerne er på højre side forsynet med en drejeknap til KONTRAST-indstilling (indstilles til billedet har en passende lys/kontrastvirkning) og på bagsiden en knap til indstilling af VERTIKALT-HOLD (V-HOLD), der indstilles så billedet ikke ruller.

Modulatoren er forsynet med antennestik til alm. farve- eller sort-hvid TV-apparat, der tilsluttes i fjernsynets antennebøsning. Fjernsynet omstilles til UHF (se fjernsynets brugsanvisning) og indstilles på kanal 36 - tuningsknappen justeres til billedet står klart på skærmen. Farver (på S/H-apparat som gråtoner), lys og kontrast indstilles på TV-apparatet (drej ned for lyden).

1.2 BRUG AF BÅNDOPTAGER.

Luk kassettelågen op ved at trykke på (STOP/EJECT) og indsæt kassetten med A-siden opad (fyldt spole til venstre). Luk lågen og spol evt. båndet tilbage (REW) og tryk på STOP, så snart båndoptageren standser. Tryk ned på den grønne CTRL taste og tryk samtidig på den lille blå ENTER taste, der sidder ved talblokken (CTRL står for CONTROL) og skærmen vil vise

```
RUN"
Press PLAY then any key:
```

Tryk nu på båndoptagerens PLAY og derefter en tilfældig taste, hvorefter programmet indlæses i computeren i løbet af ca. 5 min.

Kort efter start vil skærmen udskrive

```
Loading WELCOME 1 block 1
```

Under indlæsningen vil "block"-nummeret skifte til 2, 3 op til 11, hvorefter programmet starter automatisk, når det er indlæst. Betragt det et par gange (det starter automatisk forfra, når det er kørt igennem).
Programmet stoppes ved at trykke på CTRL, SHIFT og den røde ESC taste samtidig (ESC står for ESCAPE). Vær forsigtig med denne ordre, idet den nulstiller computeren totalt - den bør kun være sidste mulighed for at standse et program. Tryk på båndoptagerens STOP knap 2 gange, lågen åbnes og kassetten kan vendes. Gentag indlæsningsproceduren, og skærmen udlæser

```
Loading WELCOME 2 block 1
```

Følg de instruktioner, der gives på skærmen til tastaturøvelserne og tryk IKKE på båndoptagerens STOP knap, idet computeren automatisk fortsætter indlæsning, når den får besked herpå.

Også andre købte programmer kan indlæses på denne måde (følg den brugsanvisning, der følger med programmet).
Andre programmer, herunder dine egne, kan indlæses ved at skrive

```
load ""  (og tryk på en af de blå ENTER taster)
```

hvorefter skærmen vil udlæse

```
Press PLAY then any key:
```

og indlæsningsproceduren er nu den samme som ved RUN". Computeren vil denne gang dog ikke starte programmet automatisk, men på skærmen udskrive

```
Ready
```

Nu taster du

```
run        (og tryk på en blå ENTER taste)
```

og programmet vil køre (det udfører de instruktioner, der er i det pågældende program).

Hvis programmet er navngivet, kan det i stedet indlæses med

load "programtitel" (og tryk ENTER)

eller hvis det skal starte kørslen automatisk, ved at taste

run "" (ENTER).

Bemærk, at hvis der allerede har ligget et program i computerens hukommelse, vil det blive slettet, når det nye program indlæses.
Hvis det program, du søger (og har navngivet), ikke er det første på båndet, vil skærmen udskrive flg. efterhånden som den finder andre programmer

Found (anden titel) block 1

Computeren indlæser ikke dette program, men fortsætter med at søge til den har fundet det ønskede program, hvorefter dette indlæses som tidligere beskrevet.

Vil du vide, hvad der ligger på et af dine bånd, kan det ske ved hjælp af CAT-kommandoen. Se afsnit 8.1 - BASIC NØGLEORD.

1.3 UDLÆSNING.

Et program, der ligger i computerens hukommelse kan udlæses/gemmes (eng.: save) på kassettebåndoptageren. Det sker med ordren

save "program titel" (ENTER)

hvorefter skærmen udlæser

Press REC and PLAY then any key

dvs. tryk på REC og PLAY knapperne samtidig og derefter en tilfældig tast, hvorefter der udlæses

Saving (program titel) block 1

Når hele programmet er udlæst (block numrene skifter på tilsvarende måde som under indlæsning) svarer skærmen med

Ready

Bemærk, at købte programmer er beskyttet ved indlæsningen, og ikke kan kopieres på denne måde. Du kan selv beskytte dine programmer.

Hvis du har en tekst, du vil gemme som ASCII-karakterer (en tekst-fil) kan det ske med kommandoen

SAVE "programnavn",A

hvorefter der er karakterernes talværdier, der bliver bliver gemt.

Det er navnlig store tekststrenge fra wordprocessorer, der bliver gemt på denne måde.

Som omtalt, kan du selv beskytte dine programmer. Det gøres med

```
SAVE "programnavn",P
```

Programmer, der er SAVEd på denne måde kan kun indlæses igen med RUN" eller CHAIN kommandoen, og de kan ikke listes. Det er derfor også væsentligt for dig, at du gemmer en ikke beskyttet kopi hvis du senere finder ud af, at noget i programmet burde være anderledes.

Hvis du ønsker at gemme hele blokke af informationer i binær notation (dvs. på den form som det ligger i computerens hukommelse) kan det gøres med

```
SAVE "programnavn",B,startadress,længde(,startadresse)
```

hvor startadressen er det sted i RAM, hvor den binære blok starter, hvor langt det er - og er det et maskinkodeprogram kan startadressen lægges ind, så programmet selvstarter efter indlæsning. På denne måde kan man f.eks. gemme et skærmbillede. Det gøres ikke på helt samme måde, da vi ikke er interesserede i at få meldingerne fra kassetteoprationerne med på vort skærmbillede. Indtast et af de grafiske programmer i afsnit 3.5 og tilføj en ny linie sidst i programmet (husk et linienummer, der er højere end det, der står nederst i programmet):

```
(linienr.) SAVE "!programnavn",B,&C000,16384
```

hvor udråbstegnet før programnavnet får kassetteoperationsmeddelelserne undertrykt, B er parametret fra før,&C000 er skærmens begyndelsesadresse og 16384 er længden af skærmhukommelsen.

Der kan udlæses med to hastigheder, 1000 og 2000 baud (databits pr. sekund). Ved opstart er hastigheden 1000 - en hastighed, der sikrer en god dataoverførsel og som absolut bør anvendes til sikkerhedskopier. De kopier man arbejder med kan udmærket overføres ved 2000. Der omstilles mellem hastighederne med

```
SPEED WRITE heltal
```

hvor heltal er 0 for 1000 baud og 1 for 2000 baud. Ved indlæsningen fra båndoptageren sker hastighedsomskiftningen under soft-ware kontrol.

1.4 BÅNDOPTAGERFEJL.

Det kan forekomme, at der sker læsefejl på båndoptageren og skærmen udskriver

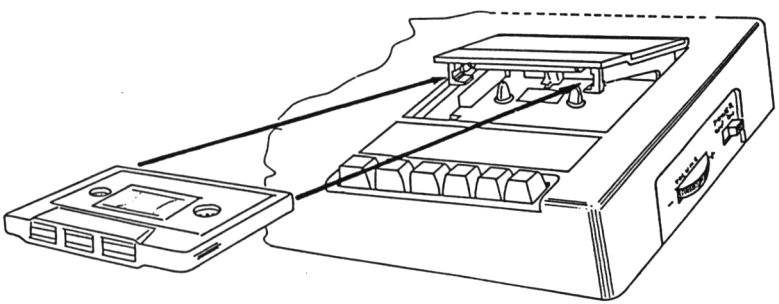
```
Read error
```

Skulle det ske, forsøg igen at indlæse båndet. Normalt skulle der ikke være problemer med bånd, der optages og gengives på samme maskine.

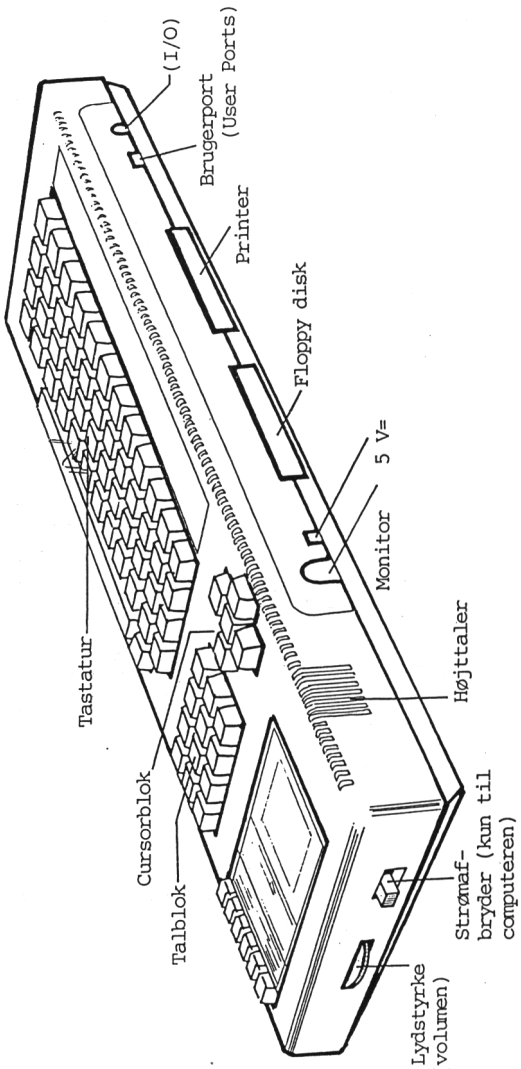
Anvend kun bånd af en virkelig god kvalitet, og det er klogt at vælge dem forholdsvis korte (C12 ell. lign.) - ikke fordi de kvalitetsmæssigt er bedre, men fordi det er langt hurtigere at finde frem til et ønsket program.

Hvis fejlen opstår medens der arbejdes med SPEED WRITE 1, så prøv at skifte baud-hastighed og forsøg om ikke optagelsen og gengivelsen så lykkes.

Read error a - der er målt en bit-værdi (0 eller 1), der er for "lang" - en optagefejl.
Read error b - data indlæst forkert fra båndet.
Read error d - data-blokken indeholder mere end 2K byte data.



(Fig. 2) Kassettebåndoptageren. Båndet kan beskyttes ved at brække den lille tap på bagsiden af (til venstre, når den side der skal beskyttes, vender opad).



(Fig. 3) Tilslutninger på Amstrad CPC 464.

2.1 GRUNDLÆGGENDE PROGRAMMERING.

Når du tænder for computeren, er den parat til direkte brug - du kan taste direkte ind (også regneudtryk) den er i direct mode - i modsætning til under programkørsel, hvor funktionerne er forudbestemt af programmet (vi skal følge programmets instruktioner).
Som du allerede nu har set, bruger vi en af de blå ENTER taster, hver gang vi vil give en besked til computeren. ENTER udfører to funktioner - den giver ordre til udførsel af den ordre, vi har skrevet og den skifter til en ny linie.
Du kan hele tiden se, hvor på skærmen, du skriver. Cursoren (markør) er den lille firkant der følger dine indtastninger på skærmen. Den kan flyttes til hvilken som helst position på skærmen med cursor-pilene, der sidder over talblokken.
Du skal altså bruge ENTER hver gang der indlæses. Prøv nu at taste

new (ENTER)

og du ser at efter et øjeblik kommer ordet Ready frem igen. Vi har "tømt" lageret for programinstruktioner og er nu klar til at taste nye informationer ind.
Tast nu flg. ind (du får store bogstaver ved at trykke på en af de grønne SHIFT taster - trykker du på den grønne CAPS LOCK vil computeren udelukkende skrive med store bogstaver):

print "Amstrad CPC 464 Computer" (ENTER)

og du vil se at skærmen udskriver

Amstrad CPC 464 Computer

Det fremtræder uden kommandoen (print) og gåseøjnene. Du kan på samme måde skrive andre ord i gåseøjnene og få dem skrevet ud.
Vi har her givet computeren en direkte ordre. Prøv at taste ENTER igen - du ser der sker kun et linieskift. Computeren kan altså ikke "huske" ordren, men det kan vi få den til.
Ved at forsyne ordren med et tal først (linienummer) kan vi lægge ordren fast ind i hukommelsen. Tast det flg. ind med de viste mellemrum (bemærk at nul sidder ved siden af 9 (ni) på det store tastatur og på talblokken - begge steder som 0 (i det hele taget må der ikke anvendes bogstavet O som nul eller bogstavet lille l for ettallet, 1)):

10 print "Amstrad CPC 464 Computer" (ENTER)

Denne gang udskrives sætningen ikke på skærmen. Vi har lagret den til senere brug. Nu kan du taste

run (ENTER)

og teksten udskrives på skærmen:

Amstrad CPC 464 Computer

Taster du nu run igen, vil teksten atter bliver udskrevet.

Ønsker du at tilføje noget til den programstump, vi har skrevet, kan du taste

list (ENTER)

og skærmen svarer med at udlæse

```
10 PRINT "Amstrad CPC 464 Computer"
```

Du ser, at ordren print er blevet ændret til PRINT . Det er noget styresystemet i Amstrad foretager med alle ordre-navne der lægges i hukommelsen som programmer. Du kan selv indtaste ordrene med store bogstaver, hvis du foretrækker det - computeren skal nok finde ud af det. Du kan også nøjes med at skrive et spørgsmålstegn (?) i stedet for PRINT - det bliver også ændret.

Det er nødvendigt at lave de viste mellemrum, da computeren ellers ikke kan skelne de enkelte ordrer fra den øvrige tekst.

Fra nu af vil vi ikke mere skrive (ENTER) efter hver indlæsningsordre. Det er underforstået, at du trykker ENTER, hver gang du ønsker en kommando/ordre udført.

2.2 FEJLREDIGERING.

Tastaturet har to taster til direkte fejlretning, medens du indtaster. Den grønne DEL og til venstre for den en grå CLR. DEL sletter det bogstav, der står umiddelbart til venstre for cursoren, mens CLR sletter det bogstav, der står under cursoren.
Indtast flg. fejlbehæftede sætning uden at trykke ENTER:

```
10 priint "Amstrad CPC 464 Computer§
```

Cursor står umiddelbart til højre for §-tegnet, der skulle have været ". Tryk på DEL - § slettes og du kan nu skrive " . Brug derefter cursortasten med pilen mod venstre og bring cursoren hen over et i i print og tryk på CLR - bogstavet slettes og ordet rykkes automatisk sammen.
Har du glemt et mellemrum, flyttes cursoren hen på det sted, hvor mellemrummet skal være, hvorefter der trykkes på mellemrumstangenten.

Prøv at indtaste ovenstående sætning med fejl igen og tryk på ENTER denne gang. Computeren svarer igen med

```
Syntax error in 10
10 priint "Amstrad CPC 464 Computer§
```

Brug cursor-pilen for at bringe cursor hen over fejlen, og tast CLR over begge fejl og indsæt tast ", og derefter ENTER og linien bliver accepteret.
Afsn. 8.3 indeholder en komplet fortegnelse over BASIC'ens fejlmeldinger.

Indtast nu flg. korte program:

```
10 print "Amstrad CPC 464 Computer"
20 print "er din garand for"
30 print
40 print "Rigtig god fornojelse"
```

Tast run og du vil se, at en print-linie uden efterfølgende tekst giver et alm. linieskift med en blank linie:

```
Amstrad CPC 464 Computer
er din garand for

Rigtig god fornojelse
```

Nu burde vi faktisk have haft lidt mere med, så tast list så vi får programmet frem igen:

```
10 PRINT "Amstrad CPC 464 Computer"
20 PRINT "er din garand for"
30 PRINT
40 PRINT "Rigtig god fornojelse"

Ready
```

Nu indtaster du

```
15 print "(et engelsj produkt)"
```

og run , hvorpå skærmen udlæser

```
Amstrad CPC 464 Computer
(et engelsj produkt)
er din garand for

Rigtig god fornojelse
```

Du ser, at computeren har lagt den nye linie ind på det rigtige sted. Det er uvæsentligt for computeren i hvilken rækkefølge den får instruktionerne - den lægger selv linierne ind på den rette plads efter linienumrene, hvad du kan se ved at liste programmet med ordren list .
Har du listet programmet, og synes at det ser sjusket ud med et ulige linienr. kan du taste renum , og derefter list når skærmen udskriver Ready , vil programmet have fået nye linienumre startende ved 10 og med spring på 10. Prøv!

Som du formentlig har bemærket, er engelsk stavet forkert, men computeren har ikke gjort os opmærksomme på det, da fejlen forekommer i en printsætning. Vi skal have ændret fejlen, og til det brug har Amstrad 3 muligheder:

NY LINIE: Du kan skrive linien helt om. Når du trykker på ENTER vil den nye linie automatisk "overskrive" den gamle og blive indsat på dens plads.

EDIT: Du kan kalde linien frem frem til redigering ved at taste

```
EDIT (linienummer)
```

Skærmen udskriver nu

```
20 PRINT "(et engelsj produkt)
```

Før cursoren ind over "j" og tast k , der nu indsættes på rette plads (og derefter ENTER).

Prøv at taste run og vi opdager nu, at der er en stavefejl i linie 40. Det kan vi rette med

COPY: Hold en af de grønne SHIFT taster nede og tryk på cursorpilen, der peger opad indtil den står over den linie, der skal rettes. Du vil se, at der nu er 2 cursorer - den ene står nederst, hvor du startede fra, og den anden står nu over den linie der skal kopieres. Slip SHIFT tasten og tryk på den grønne COPY taste (mellem cursorpilene) indtil den står over det bogstav hvor fejlen er.
Du vil se, at indtil nu er linien blevet kopieret nøjagtig som den var ved den nederste cursor.
Nu taster du ø - dette skrives af den nederste cursor (men ikke den øverste). Tryk igen på COPY tasten, indtil resten af linien er kopieret. Tast enter og den rettede nederste linie lægges ind i hukommelsen og kopi-cursoren forsvinder.
Foruden fejlretning kan COPY-funktionen bruges til at flette en ny programlinie sammen af en eller flere linier, der allerede er skrevet.
Men der mangler en linie for at få sætningen til at hænge rigtigt sammen. Skriv

```
30 PRINT "mange fornøjelige timer"
```

Hvis du nu lister programmet, vil du se, at den nye linie har overskrevet den gamle. Det vil altid ske, hvis der lægges en linie ind med et nummer, der eksisterer i forvejen. Ved at taste et linienummer uden efterfølgende ordrer, vil det slette en li- nie med samme nummer.

Tast nu CLS (clear screen - "rens" skærmen) og skærmen sletter hvad der er skrevet på den indtil nu.
Vil du se dit program igen, kan du taste LIST, hvorefter programmet bliver listet på skærmen.

2.3 YDERLIGERE PROGRAMMERING.

Det følgende program, der kan udskrive primtallene fra 1 til 100 vil vise en række grundlæggende programmeringsfunktioner på Amstrad CPC 464, og vil senere blive modificeret til yderligere funktioner:

```
10 FOR x=1 TO 100 STEP 1
20 a = INT(x/2)
30 b = 2*a
40 IF b=x THEN GOTO 150
50 b = 1
60 a = INT(SQR (x))
70 c = b+2
80 b = c
90 IF b > a THEN GOTO 140
100 c = INT(x/b)
110 d = b*c
120 IF d = x THEN GOTO 150 ELSE GOTO 70
130 REM udskrift
140 PRINT x;" ";
150 NEXT x
160 PRINT "Slut på primtal."
```

Indtast programmet, kør det (RUN), se hvordan udskriften bliver og LIST det derefter.

Nogle af de regneudtryk, CPC 464 kan behandle, indgår i dette program. De alm. 4 regningsarter er addition (+), subtraktion (-), multiplikation (*) og division (/).
SQR (square root) beregner kvadratroden af tallet (variablen), der står i parantesen.
INT (integer) tager heltallet af det tal (variabel) der følger efter i parantes, dvs. den bortkaster evt. decimaltal.

Linie 10 opstiller en FOR-NEXT løkke (en tæller). Programmet bliver løbet igennem et antal specificeret gange - her startende med værdien 1 og op til 100 . Step-værdien angiver hvor stort et spring, der skal foretages hver gang. Her er den sat til 1 (og behøver ikke at blive skrevet, når værdien er 1 - det vil altid være underforstået). x er en variabel, der her anvendes til at gemme tælleværdien.
Linie 150 tildeler x en den næste værdi. Hvis den nye værdi er inden for det område, der er opstillet i linie 10, gennemløbes programmet endnu en gang, ellers går det videre til linie 160.

Linie 20 giver variablen a en værdi, beregnet ud fra den værdi x har i det pågældende gennemløb af programmet. I andre versioner af BASIC er det nødvendigt at skrive "LET a = ..." - dette er underforstået i den BASIC-version, der ligger i CPC 464. Variablen består her af et bogstav, men kan bestå af flere, evt. hele ord på op til 40 karakterer. Linien "tilskriver" variablen a værdien af regneudtrykket INT(x divideret med 2).
Linie 40 er en betinget hop-sætning. Ved hjælp af ordren IF undersøges det, om variablen b er lig med variablen x . Er det tilfældet, hoppes (GOTO) til linie 150 - er variablerne forskellige går programudførelsen videre med linie 50.
Linie 120 undersøger ligeledes en betingelse, men her er sætningen udvidet med ELSE. Dvs. "hvis d er lig med x så gå til 150 ellers (ELSE) gå til 70".
Linie 130 er en REM (remark - bemærkning) sætning. Denne kan indsættes overalt i programmet. REM fortæller computeren, at den ikke skal tage hensyn til hvad der står på resten af linien. Det kan være praktisk med en oplysning om, hvad der foretages i det efterfølgende programafsnit.
Linie 140 udskriver værdien af x , hvis den aktuelle værdi i programmet er beregnet til at være et primtal. Semicolon er anvendt til at formattere udskriften. Når ";" anvendes, vil tallene blive udskrevet umiddelbart efter hinanden, hvorfor det er nødvendigt at indsætte et mellemrum ved hjælp af gåseøjnene. Hvis du udskifter linie 40 med

```
40 PRINT x,
```

vil du se en ny formattering. Det skyldes at komma deler efter en forudbestemt tabulering, der ligger på tværs af skærmen med en indbyrdes afstand af 13 karakterer.

Prøv at ændre TO-værdien i linie 10 til f.eks. 1000 og kør programmet igen. Prøv derefter at slette GOTO i linierne 40, 90 og 120 - du vil opdage, at disse heller ikke er nødvendige, for at computeren skal forstå, hvad du mener. Det kan dog kun lade sig gøre i sammensætninger som vist. Står det alene på en linie, kan det ikke udelades (se iøvrigt 8.1 BASIC NØGLEORD).

2.4 INPUT.

I primtalsprogrammet indlagde vi tællervariablen i programmet. Vi kan imidlertid også indlæse tallet fra tastaturet ved at lave en ny linie 5 og ændre linie 10:

```
5 INPUT tal
10 FOR x = 1 TO tal
```

(vi husker stepværdien ikke er nødvendig her). Tast RUN og skærmen udskriver et spørgsmålstegn - du indtaster det tal, hvortil du ønsker primtallene udskrevet (f.eks. 750) og trykker på ENTER, hvorefter programmet kører som før.

Denne procedure gælder for tal-variabler. Du kan imidlertid også indlæse en streng-variabel, dvs. med alle bogstaver og tegn, CPC 464 råder over (§%A7Ü?) samt de tegn, du selv måtte lægge ind (se afsnit 2.5).

Dette gøres ved at tildele variabel-navnet et dollartegn ($) efter navnet, som det fremgår af flg. program:

```
10 REM modulustest
20 z=0:x=0:e=0
30 B$="432765.4320"
40 INPUT "Indtast CPR-nummer paa formen
          DDMMAA-XXXK ", A$
50 FOR i = 1 TO 11
60 IF i = 7 THEN 90
70 z = VAL (MID$(A$,i,1)):e = VAL (MID$(B$,i,1))
80 e = e*z:x = x+e
90 NEXT i
100 c = 11-(x-(INT(x/11)*11))
110 IF c>9 THEN 130
120 GOTO 140
130 c=0
140 PRINT A$,"Kontrolciffer = ";c
```

Prøv at køre programmet, og indtast et CPR-nummer når computeren beder om det.
Inden vi gennemgår programmet, så lad os se, hvorledes kontrolcifferet beregnes (efter modulus-11 metoden). Det foregår ved at hver enkelt ciffer i personnummeret (indeholdt i A$) multipliceres med med en vægt (indeholdt i B$). De fremkomne tal adderes derefter. Hvis f.eks. personnummeret er 260449-1447 giver det:

CPR:	2	6	0	4	4	9	1	4	4	7	
	*	*	*	*	*	*	*	*	*	*	
vægt:	4	3	2	7	6	5	4	3	2	0	
=	8	+18	+ 0	+28	+24	+45	+ 4	+12	+ 8	+ 0	= 147

De to cifre, der skal multipliceres, findes i A$ og B$ i linie 70. Variablen z tildeles værdien af udtrykket på højre side af lighedstegnet. Da vi ikke kan regne direkte på et streng-udtryk, indsættes først VAL (value - værdi), der omdanner resultatet af strengen, så resultatet kan beregnes.
MID$ er en strengoperator, der angiver, at vi ønsker at arbejde med den streng, der en specificeret i parantes - de efterfølgende variabler fortæller hvilket bogstav i strengen, vi arbejder

på ("i" - fra FOR - NEXT funktionen) og "1", at vi ønsker en karakter behandlet af gangen.
I linie 80 multipliceres de to fremkomne værdier og de summes løbende op i x .
For at finde kontrolcifret divideres resultatet af x (147) med 11 (= 13.3636364). Vi bruger heltalsdelen (INT) = 13 som multipliceres med 11 (= 143). Derefter trækkes 143 fra 147 (= 4) og ved at trække 4 fra 11 (modulus-tallet) fremkommer kontrol-cifret 7.

Bemærk også, at der kan skrives flere kommandoer på samme linie, når disse adskilles med kolon (:).

Til strengbehandlingen er der yderligere tilknyttet to operatorer. Se LEFT$ og RIGHT$ i 8.1 BASIC NØGLEORD.

Når programmet har kørt en modulustest, standser det med ordene Ready. Det kan undgås ved f.eks. at indlægge flg. linier sidst i programmet:

```
150 PRINT:INPUT "Skal der foretages flere test? (J/N) ",
svar$
160 IF svar$="J" THEN 20 ELSE IF svar$="N" THEN END
170 GOTO 150
```

Navnet på en strengvariabel kan ligeledes indeholde 40 karakterer. Vi tester svar$ og begynder i linie 20 hvis der svares J og slutter (END), hvis der svares N . Er der ikke svaret med et af de to bogstaver, udføres linie 170 og spørgsmålet stilles igen.

Ønsker du at afbryde programudførelsen, kan du trykke på den røde ESC taste. Ved et tryk standses udførelsen, indtil der trykkes på en af de øvrige taster. Trykker du i stedet på ESC endnu en gang, udskriver skærmen

```
Break in (linie nr.)
Ready
```

og programmet standser. Hvis du ombestemmer dig, kan det startes igen med ordren CONT (continue - fortsæt), der også virker hvis en programlinie har indeholdt STOP eller END. Hvis du taster ESC i en INPUTlinie, udskrives *BREAK*.

Du har formentlig bemærket, at tastaturet mangler et dansk karaktersæt . Det kan du selv lave, og det er omtalt i næste afsnit.

2.5 KARAKTERSÆTTET.

Ca. halvdelen af Amstrad CPC 464's karaktersæt kan skrives direkte fra tangentbordet. Computeren indeholder imidlertid endnu flere karakterer (se siderne 21 - 24), der kan udskrives med flg. program:

```
10 FOR n=32 TO 255
20 PRINT CHR$(n);
30 NEXT n
```

Karaktererne fra 0 til 31 tjener som controlkarakterer f.eks. i forbindelse med CTRL tasten. Der er imidlertid tilknyttet symboler til disse tegn også. De kan udskrives med programmet:

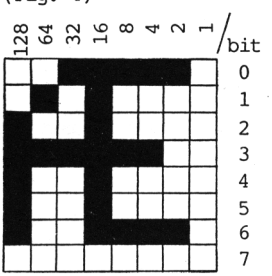
```
10 FOR n=0 TO 31
20 PRINT CHR$(1)+CHR$(n);
30 NEXT n
```

Lagde du mærke til "klokken" under programkørslen. Den svarer til PRINT CHR$(7), der vil aktivere klokken under programkontrol evt. indlagt i en print-sætning:

```
40 PRINT "Amstrad ßG CPC 464 ßG Computer"
```

hvor ßG betyder nedtrykning af CTRL og G samtidig (se afsn. 8.2 - KONTROLKODER).

(Fig. 4)



Samtlige symboler i området 32 til 255 kan omdefineres til eget brug med kommandoerne SYMBOL AFTER og SYMBOL.
Vi vil bruge det til at skabe et dansk karaktersæt på tasterne , og . Det laveste symbol-nummer, vi skal bruge, er 91, hvorfor SYMBOL AFTER sættes til denne værdi - 1. Ved hjælp af SYMBOL indlæses først symbol-/karakternummer og derefter talværdierne for de enkelte vandrette linier i symbolnettet her ved siden af for bogstavet Æ .
Indtast flg. program og kør det:

```
10 SYMBOL AFTER 90
20 SYMBOL 91,0,0,236,18,124,144,110,0
30 SYMBOL 123,62,80,144,252,144,144,158,0
40 SYMBOL 93,48,0,120,12,124,204,118,0
50 SYMBOL 125,24,36,126,102,102,126,102,0
60 SYMBOL 92,0,2,124,206,214,230,124,128
70 SYMBOL 96,58,108,206,214,230,108,184,0
```

Prøv nu at trykke på de nævnte taster, og du er forsynet med et dansk karaktersæt, der kan indlægges i dine egne programmer eller sågar indlægges i Amsoft wordprocessorer, hvor du lægger karaktersættet ind først og derefter indlæser wordprocessoren på vanlig vis med CTRL og ENTER tasten i talblokken.
I forbindelse med anvendelse af andre prntere end Amstrad DMP 80 kan det være nødvendigt at flytte nogle af de danske karakterer til andre taster (symbol/karakterkoder) - tal med din forhandler herom, hvis du er i tvivl.

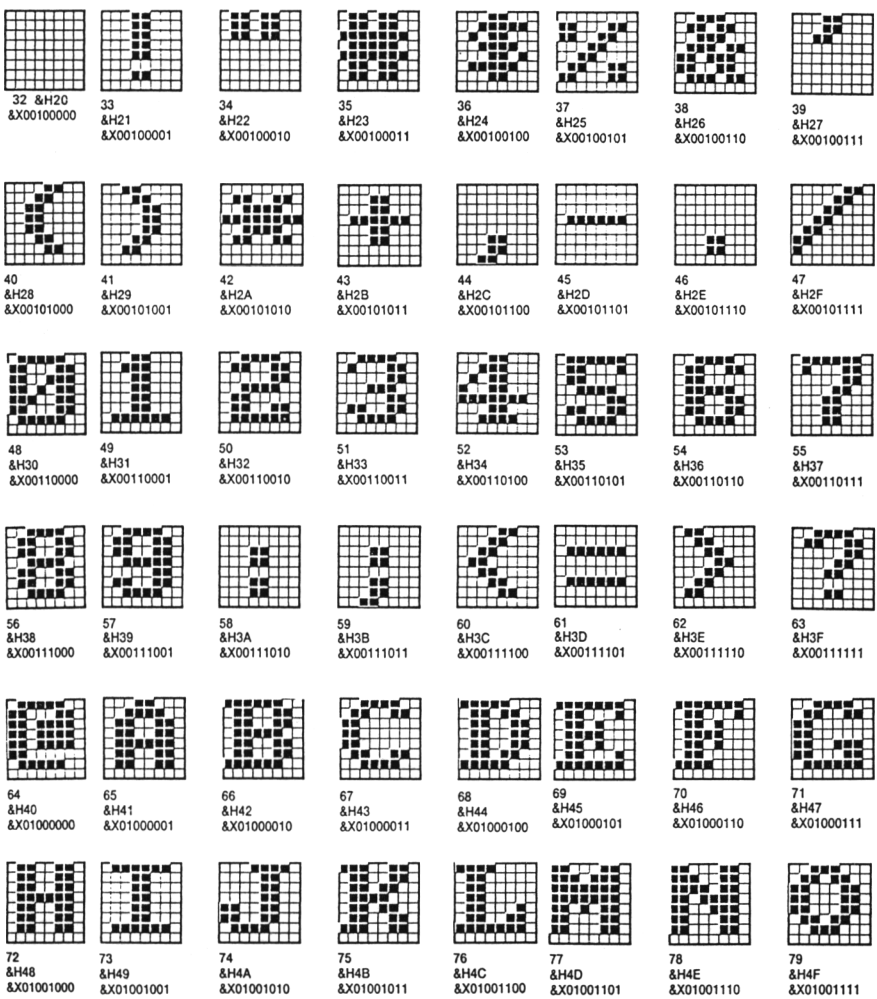
Når du selv skal definere nye tegn, er det lettest at gøre i et tilsvarende net. Du finder først ud af, hvilken karakter du ønsker at benytte (siderne 21-24) og skriver dens decimaltal umiddelbart efter SYMBOL. Nu udfylder du de felter, hvor tegnet skal fremtræde i - det giver nogle hvide felter (slukkede, defineret ved 0) og sorte felter (tændte, defineret ved 1). Ved at addere tallene der står over nettet hvor der er et sort felt, finder du

for hver vandrette linie et decimaltal, der skrives på programlinien efter hinanden, adskilt med komma.
For bogstavet Æ (linie 30 - symbol 123), giver den øverste linie

```
0*128+0*64+1*32+1*16+1*8+1*4+1*2+0*1 = 62
```

De øvrige linier beregnes på lignende måde.

Se også afsn. 7.1 - TALSYSTEMER.



Siderne 21-24 viser karaktersættes opbygning.


80
&H50
&X01010000


81
&H51
&X01010001


82
&H52
&X01010010


83
&H53
&X01010011


84
&H54
&X01010100


85
&H55
&X01010101


86
&H56
&X01010110


87
&H57
&X01010111


88
&H58
&X01011000


89
&H59
&X01011001


90
&H5A
&X01011010


91
&H5B
&X01011011


92
&H5C
&X01011100


93
&H5D
&X01011101


94
&H5E
&X01011110


95
&H5F
&X01011111


96
&H60
&X01100000


97
&H61
&X01100001


98
&H62
&X01100010


99
&H63
&X01100011


100
&H64
&X01100100


101
&H65
&X01100101

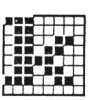

102
&H66
&X01100110


103
&H67
&X01100111


104
&H68
&X01101000


105
&H69
&X01101001


106
&H6A
&X01101010

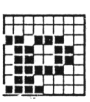

107
&H6B
&X01101011


108
&H6C
&X01101100


109
&H6D
&X01101101

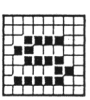

110
&H6E
&X01101110

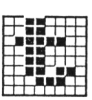

111
&H6F
&X01101111


112
&H70
&X01110000


113
&H71
&X01110001


114
&H72
&X01110010


115
&H73
&X01110011


116
&H74
&X01110100


117
&H75
&X01110101


118
&H76
&X01110110


119
&H77
&X01110111

120
&H78
&X01111000

121
&H79
&X01111001

122
&H7A
&X01111010

123
&H7B
&X01111011

124
&H7C
&X01111100

125
&H7D
&X01111101

126
&H7E
&X01111110

127
&H7F
&X01111111

128
&H80
&X10000000

129
&H81
&X10000001

130
&H82
&X10000010

131
&H83
&X10000011

132
&H84
&X10000100

133
&H85
&X10000101

134
&H86
&X10000110

135
&H87
&X10000111

136
&H88
&X10001000

137
&H89
&X10001001

138
&H8A
&X10001010

139
&H8B
&X10001011

140
&H8C
&X10001100

141
&H8D
&X10001101

142
&H8E
&X10001110

143
&H8F
&X10001111

144
&H90
&X10010000

145
&H91
&X10010001

146
&H92
&X10010010

147
&H93
&X10010011

148
&H94
&X10010100

149
&H95
&X10010101

150
&H96
&X10010110

151
&H97
&X10010111

152
&H98
&X10011000

153
&H99
&X10011001

154
&H9A
&X10011010

155
&H9B
&X10011011

156
&H9C
&X10011100

157
&H9D
&X10011101

158
&H9E
&X10011110

159
&H9F
&X10011111

160
&HA0
&X10100000

161
&HA1
&X10100001

162
&HA2
&X10100010

163
&HA3
&X10100011

164
&HA4
&X10100100

165
&HA5
&X10100101

166
&HA6
&X10100110

167
&HA7
&X10100111

168
&HA8
&X10101000

169
&HA9
&X10101001

170
&HAA
&X10101010

171
&HAB
&X10101011

172
&HAC
&X10101100

173
&HAD
&X10101101

174
&HAE
&X10101110

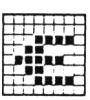
175
&HAF
&X10101111

176
&HB0
&X10110000

177
&HB1
&X10110001

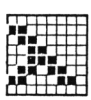
178
&HB2
&X10110010

179
&HB3
&X10110011

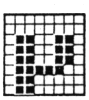


180
&HB4
&X10110100

181
&HB5
&X10110101



182
&HB6
&X10110110



183
&HB7
&X10110111

184
&HB8
&X10111000

185
&HB9
&X10111001

186
&HBA
&X10111010

187
&HBB
&X10111011

188
&HBC
&X10111100

189
&HBD
&X10111101

190
&HBE
&X10111110

191
&HBF
&X10111111

192
&HC0
&X11000000

193
&HC1
&X11000001

194
&HC2
&X11000010

195
&HC3
&X11000011

196
&HC4
&X11000100

197
&HC5
&X11000101

198
&HC6
&X11000110

199
&HC7
&X11000111

200
&HC8
&X11001000

201
&HC9
&X11001001

202
&HCA
&X11001010

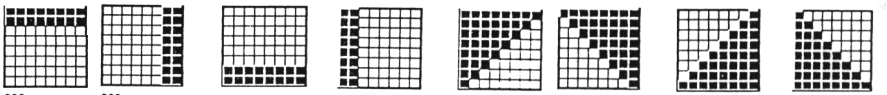
203
&HCB
&X11001011

204
&HCC
&X11001100

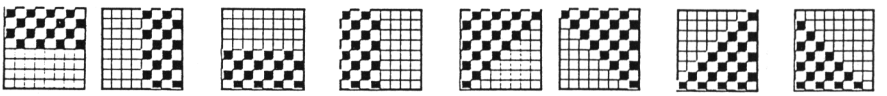
205
&HCD
&X11001101

206
&HCE
&X11001110

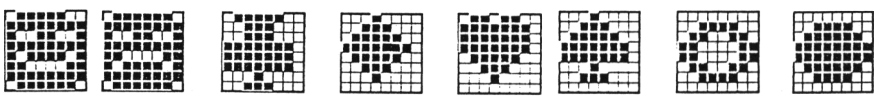
207
&HCF
&X11001111



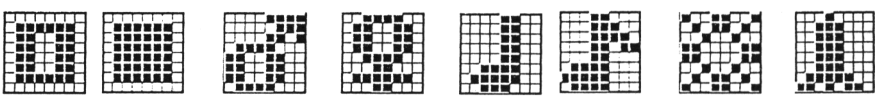
208 &HD0 &X11010000 | 209 &HD1 &X11010001 | 210 &HD2 &X11010010 | 211 &HD3 &X11010011 | 212 &HD4 &X11010100 | 213 &HD5 &X11010101 | 214 &HD6 &X11010110 | 215 &HD7 &X11010111



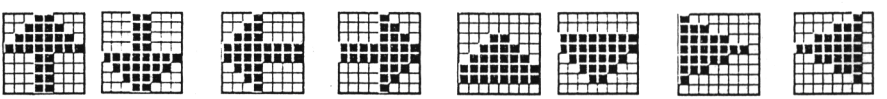
216 &HD8 &X11011000 | 217 &HD9 &X11011001 | 218 &HDA &X11011010 | 219 &HDB &X11011011 | 220 &HDC &X11011100 | 221 &HDD &X11011101 | 222 &HDE &X11011110 | 223 &HDF &X11011111



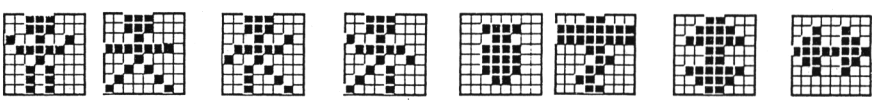
224 &HE0 &X11100000 | 225 &HE1 &X11100001 | 226 &HE2 &X11100010 | 227 &HE3 &X11100011 | 228 &HE4 &X11100100 | 229 &HE5 &X11100101 | 230 &HE6 &X11100110 | 231 &HE7 &X11100111



232 &HE8 &X11101000 | 233 &HE9 &X11101001 | 234 &HEA &X11101010 | 235 &HEB &X11101011 | 236 &HEC &X11101100 | 237 &HED &X11101101 | 238 &HEE &X11101110 | 239 &HEF &X11101111



240 &HF0 &X11110000 | 241 &HF1 &X11110001 | 242 &HF2 &X11110010 | 243 &HF3 &X11110011 | 244 &HF4 &X11110100 | 245 &HF5 &X11110101 | 246 &HF6 &X11110110 | 247 &HF7 &X11110111



248 &HF8 &X11111000 | 249 &HF9 &X11111001 | 250 &HFA &X11111010 | 251 &HFB &X11111011 | 252 &HFC &X11111100 | 253 &HFD &X11111101 | 254 &HFE &X11111110 | 255 &HFF &X11111111

Bemærk:

En del printere har lille ø liggende i ROM, som karakter CHR$ (124). Denne karakter er på CPC464 tilknyttet ¦ , der senere skal anvendes til kald af ydre ROM-moduler. Lægges ø her, vil det ikke have betydning for word-processoren, men vil CHR$(124) blive anvendt i datasætninger, kommer den i konflikt med computerens styresystem og "fjerner" den næste karakter i datasætningen.

Mellemrum er CHR$(32).

SHIFT LOCK alene giver udelukkende store bogstaver. Trykkes samtidig CTRL og SHIFT LOCK skiftes der også til de øverste funktioner på de øvrige taster.

3.1 SKÆRMFORMATER.

Amstrad CPC 464 er forsynet med 3 skærmformater. Når du tænder for den, starter den op i MODE 1 (skærmformat 1):

MODE 0 - 20 kolonner (karakterer pr. linie)
MODE 1 - 40 kolonner
MODE 2 - 80 kolonner

For opløsningen på grafikskærmen, se afsn. 3.4 - GRAFIK.

MODE (nummer) kan indtastes direkte fra tastaturet, men kan også indlægges som programlinier, hvilket vil sige, at dit program kan skifte mellem de forskellige skærmformater under udførelsen.

Afprøv flg. programstump for at se skærmformaterne:

```
10 FOR n=0 TO 2
20 MODE n
30 LOCATE 12,12
40 PRINT "AMSTRAD"
50 FOR i=1 TO 1000:NEXT i
60 NEXT n
```

Linie 10 opstiller en tæller for gennemløb af skærmene. Linie 30 placerer cursoren i den ønskede PRINT-position og linie 50 er en "tom" tæller, der er sat op for at give en passende pause, så vi kan nå at se placeringen af teksten og skærmformatet.

LOCATE ordren placerer cursoren på et bestemt sted på skærmen, bestemt ved

LOCATE kolonne nr.,linie nr.

Den engelske manual indeholder indeholder planlægningsskemaer for alle 3 MODEs i appendix VI, side 2-4.

3.2 VINDUER.

Før vi begynder på at definere vinduer (WINDOWs), indtaster vi en lille rutine, der ændrer den lille ENTER-taste i talblokken til funktionstaste. Ved at gøre det, kan vi altid vende tilbage til vores udgangspunkt, hvis vi ikke kan "styre" vinduerne.

```
KEY 139,"MODE 2:PAPER 0:INK 1,0:INK 0,9:LIST"+CHR$(13)
```

der definerer, hvilken taste, vi ønsker at bruge, "hvad der skal udføres" og en kontrol-karakter, der får cursoren til at placere sig i øverste venstre hjørne.

Vi prøver her at definere 2 vinduer, det ene med samme papirfarve som den øvrige skærm. Indtast

```
10 INK 1,0:PEN 1
20 MODE 1
30 WINDOW #0,10,30,7,18
```

```
40 PAPER 2:CLS
50 WINDOW #1,2,6,2,9
60 PAPER 0:CLS
```

Når du har kørt programmet, så prøv at liste det med henholdsvis LIST #0 og LIST #1.
Vil du slette skærmbilledet i et vindue, sker det med CLS #0 eller #1.
Du kan skifte om på, hvilke vinduer du vil arbejde i med ordren

```
WINDOW SWAP vindue,vindue
```

- prøv f.eks. at tilføje

```
70 PRINT "MIN AMSTRAD ER"
80 WINDOW SWAP 0,1
90 PRINT "FAKTISK HELT NY"
```

3.3 FARVER.

CPC 464 har 27 forskellige farver at vælge imellem. På Amstrad GT 64 - monokrom skærm vil farverne fremstå som forskellige nuancer af grønt (på S/H TV som gråtoner). Alle 27 farver er til rådighed, men de kan ikke alle bruges samtidig på skærmen. Ud af de 27 farver giver de forskellige skærmformater mulighed for samtidig anvendelse af flg. antal farver:

```
MODE 0 - op til 16
MODE 1 - op til 4
MODE 2 - op til 2
```

Du kan skifte farverne på:

```
BORDER - farven på kanten af skærmbilledet    (1 - blå)
PAPER  - den baggrundsfarve,
         der skrives på ("papiret")           (1 - blå)
PEN    - den farve,
         der skrives med (karaktererne)       (24 - lys gul)
```

Tallene i paratesen angiver farven i henhold til farveskemaet for de 27 farver:

INK nr.	COLOUR/INK	INK nr.	COLOUR/INK
0	sort	14	pastelblå
1	blå	15	orange
2	lys blå	16	lyserød
3	rød	17	lys magenta
4	magenta	18	lysegrøn
5	mauve	19	søgrøn
6	lys rød	20	lys cyan
7	lilla	21	lime grøn
8	lys magenta	22	pastelgrøn
9	grøn	23	pastel cyan
10	cyan	24	lysegul
11	skyblå	25	pastelgul
12	gul	26	lys hvid
13	hvid		

Før du begynder med farverne, så indtast linien fra foregående afsnit, hvor vi definerede KEY 139.

Prøv at skifte BORDERfarven først (i MODE 1). For at få farverne til at skifte er det tillige nødvendigt med en CLS ordre, så tast

```
BORDER 3:CLS
```

og kanten skifter til rød (jvnf. ovenstående skema). Brug PAPER og INK på tilsvarende måde. Du oplever ganske givet, at farverne ikke opfører sig helt, som du havde forventet. Alle 3 formater hænger nemlig sammen i brug af penne og blæk. Kan du ikke finde tilbage til læselige farver, så tast CTRL, SHIFT og ESC samtidig for at nulstille computeren.

Som nævnt er der 16 penne, der hver især kan styre en INK (der bruges også en PEN til PAPER. Det er derfor nødvendigt med et nyt skema, der viser sammenhængen mellem PAPER/PEN og nummeret på den anvendte farve, når der tændes (nulstilles):

PAPER/PEN nr.	Mode 0	Mode 1	Mode 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	*)	20	1
15	**)	6	24

*) Flashing 1,24 - **) Flashing 16,11 (flash = blink)

Når computeren startes op, bruges PEN 1 - i skemaet herover ses det at svare til INK 24 i alle 3 MODEs. Ved at kigge i det foregående skema, ses det, at INK 24 svarer til lysegul.
Indtast nu

```
PAPER 2:CLS
```

Det er ikke videre tydeligt at læse, så fortsæt med at taste

```
PEN 3
```

Herved skifter vores skrift til rød, der er det nuværende indhold i PEN 3.
Vi vil nu skifte farver i pennene. Vi bruger pen 2 til at give PAPER 2. Den farve vil vi nu skifte til sort med kommandoen

```
INK 2,0
```

hvor 2 som nævnt er den PAPER-farve, vi har i pennen, og 0 er den farve, vi ønsker udskiftet med (sort). Nu skal vi have skiftet farven i PEN 3 (fra rød til hvid). Det gøres med

```
INK 3,26
```

hvorefter du taster CLS for at få papirfarven til at skifte.
Altså: vælg først hvilken PEN eller PAPER du vil anvende - derefter "tilegner du en farve til den valgte pen.

Du kan få farverne til at blinke (flash), som du så ved skemaet. Det kan ske i mode 0 ved at vælge PEN 14 (blinker sort/lysegul) eller PEN 15 (blinker pink/skyblå). Afprøv det med

```
MODE 0:PEN 15
```

hvorefter ordet Ready blinker med de farver, PEN 15 er tilegnet. Vil du også have baggrunden til at blinke, kan det gøres med

```
PAPER 14:CLS
```

Det kan også lade sig gøre med andre farver. Ved at tilføje endnu et tal til henholdsvis BORDER og INK, kan de forskellige farver bringes til at blinke, f.eks. kanten mellem rød og grøn:

```
BORDER 3,9
```

eller brug f.eks. PEN 1 til at blinke med rødt og hvidt:

```
INK 1,3,13
```

Endelig kan også papiret bringes til at blinke med

```
INK 0,4,10
```

Med kommandoen SPEED INK kan man selv bestemme, hvilken hastighed, farverne skal blinke med (pas på øjnene!).

Ved hjælp af en af de kontrolkarakterer vi allerede har været lidt inde på, kan vi gøre farverne transparante.
Du har sikert bemærket, at cursoren er transparant, når du har kørt den hen over karaktererne. Prøv at indtaste flg. program, der tillige laver lidt grafik:

```
 5 CLS
10 INK 0,14
15 INK 1,24
20 ORIGIN 320,200
30 FOR n=1 TO 200
40 DRAW 15+n*SIN(n*PI/2),(SIN(PI/2))*n*SIN(n)
50 NEXT n
60 INK 2,0
70 PEN 2
80 LOCATE 12,12
90 PRINT "AMSTRAD COMPUTER"
```

Linie 10 tilegner INK 14 til PEN 0 (papirfarven). Linie 20 tilegner INK 24 til PEN 1 (pennen, der tegner grafikken). Linie 60 tilegner INK 0 til PEN 2, og i linie 70 skifter vi fra PEN 1, der var i brug til grafikken, til PEN 2 for at udskrive teksten

i linie 90. Hvis vi ikke havde skiftet PEN, men kun farve i PEN 1, ville grafikken også have skiftet til den nye farve, idet farven følger den PEN, der er skrevet med.
Lagde du mærke til, at teksten i linie 90 skrev hen over grafikken, og slettede den til fordel for papirfarven.
Det kan undgås ved at gøre farven i PEN 2 transparant ved tilføjelse af 2 linier:

```
75 PRINT CHR$(22)+CHR$(1)
100 PRINT CHR$(22)+CHR$(0)
```

hvor linie 75 har kontrolkoden, der "tænder" for transparanten og linie 100 "slukker".
Pas på ved editering, hvis du ikke har fået slukket, idet du ellers kan lægge karaktererne oven i hinanden.

3.4 LYD.

Lyden generes af en indbygget højttaler, der justeres over styrkeknappen, der er anbragt på højre side at tastaturet. Lyden kan også overspilles til stereoanlægget eller aflyttes over hovedtelefoner med et lille stereo-jack stik, der tilsluttes I/O-stikket på computerens bagside.

Lydkommandoen SOUND har 7 parametre, af hvilke de to første skal anvendes - resten er valgfrie. Den komplette lydkommando er:

```
SOUND G,H,I,J,K,L,M
```

hvor

G - er kanal status, der altid skal bruges, idet denne henviser til hvilke(-n) af kanalerne, der anvendes, altså en reference. Status angives som heltal, men når det konverteres til binær nototaion, angiver de aktive bits i byten følgende kommandoer:

Decimal	Bit	Kommando
1	0	sender lyden til kanal A
2	1	sender lyden til kanal B
4	2	sender lyden til kanal C
8	3	spiller sammen med kanal A
16	4	spiller sammen med kanal B
32	5	spiller sammen med kanal C
64	6	hold
128	7	flush

hvor bit 0 er den mindst betydende bit og bit 7 er den mest betydende (se afsnit 7.1), hvilket betyder

```
  2 = send den følgende lyd til kanal A
  5 = send den følgende lyd til kanal A og C
 98 = 64 + 32 + 2
    = send den følgende lyd til kanal B, spil sammen med
      kanal C og hold
```

Det er vigtigt at huske, at sammenspil (rendezvous) mellem to eller tre kanaler kræver kontrol af kanalstatus (flag) på hver af kanalerne. Hold på hvilken som helst kanal, standser udførelsen af den kommando og fastlåser køen (queue) bag den indtil den løsnes med en RELEASE kommando (eller er flush'ed af en senere Sound kommando).
Når flush-bit'en aktiveres af en kanal vil lyden blive sendt igennem med flush og efterlader lydkøen tom og hovedkanalen inaktiv.

H - Tone perioden er også obligatorisk og har en værdi i området 0 til 4095.
Opstiller den periode, der fastsætter frekvensen af den lyd, der skal spilles. Frekvensen kan beregnes på formlen

```
frekvens = 125000/periode
```

hvis der anvendes 0, sættes der ingen frekvens, og toneperioden er så mest velegnet til "støjkanal" (se appendix VII i den engelske manual, hvor frekvenser og perioder er opgivet med en procentskala for hvor stor afvigelsen er fra den "ægte" tone.

I - Duration (varighed) sættes til 20, hvis den ikke opgives, og kan ligge i området -32768 til +32767. Enheden er 1/100 sekund og sat til 20 vil det sige en varighed på 0.20 sekund når værdien er positiv. Er den nul bestemmes den af den lyd envelope, der er sat. Er den negativ giver den positive værdi af dette tal det antal gange, envelopen skal gentages.

J - Volume (lydstyrke) er et heltal i området 0 til 15 (0 til 7, hvor der ikke er specificeret en lyd envelope). Specificeres lydstyrken ikke sættes den til 12 (4 hvis ingen envelope).
Udgør en startværdi der kan ændres af lyd envelopen hvis den er specificeret.
Sat til nul er lydstyrken afbrudt!!!

K - Volume envelope (lyd env.) er et heltal i o området 0 til 15 eller 0 hvis den ikke er specificeret. Værdien, der anvendes er en envelope, der er predefineret med ENV kommandoen.
Er volume envelopen sat til 0 kan dette ikke ændres af ENV kommandoen.

L - Tone envelope har værdier i området 0 til 15 og sættes til 0 hvis den udelades. Værdien er predifeneret med ENT kommandoen. Er tone envelopen sat til 0 kan dette ikke ændres af ENT kommandoen.

M - Støjperiode. Specificerer den støj, der skal tilsættes lyden. Hvis den ikke er specificeret i området 0 til 15, sættes den automatisk til nul. Bemærk at der kun kan sættes en en støjperiode, hvilket vil sige at alle kanaler der er specificeret for støj, modtager den samme støj.

Bemærk at frekvensen måles i hertz (Hz) og at kammertonen A (internationalt A) ligger i området oktav 0 og er på 440 Hz.

Værdierne kan kalkuleres ud fra følgende formler:

```
Frekvens=440*(2↑(oktav+(10-n)/12))
```

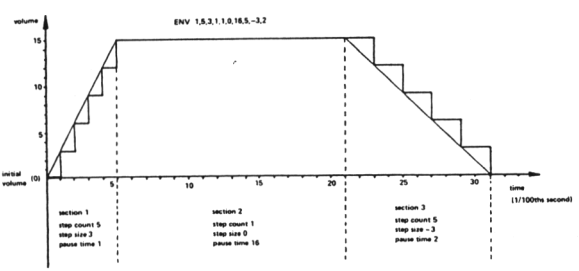
```
Periode =ROUND(125000/frekvens)
```

hvor n=1 for C, 2 for C#, 3 for D etc. Se appendix VII i den engelske manual.

Så må det være tid til lidt musik. Her er 3 rutiner, lagt ind i en programlistning. Prøv at skille det ad, og ved hjælp af en FOR/NEXT løkke lade dem køre lidt længere.

```
6 ENV 6,15,-1,1
7 ENV 5,15,-1,10
8 ENT 1,30,10,1
10 READ h,i:IF h=9999 THEN END
20 SOUND 1,h,i,15
30 GOTO 10
100 READ h,i,0,2:IF h=9999 THEN END
110 SOUND 1,h,i,0,2
120 GOTO 100
200 READ h:IF h=9999 THEN END
210 SOUND 4,h,20,15,1
220 GOTO 200
1000 DATA 478,50,319,50,358,13,379,13,42
6,13,239,60,319,60,358,13,379,13,426,13,
239,60,319,60,358,15,379,15,358,15,426,7
0,9999,9999
1410 DATA 1911,120,1911,80,1911,40,1911,
120,1607,80,1703,40,1703,80,1911,40,1911
,80,2025,40,1911,120,9999,9999
2020 DATA 60,53,47,45,60,0,45,47,45,40,5
3,0,53,47,45,36,40,40,45,45,47,53,47,60,
9999
```

Volume envelope (ENV) giver liv til tonen ved at danne et forhold mellem lydstyrke (volume) og tiden (time). Forholdet kan med fordel indtegnes på et stykke millimeterpapir (se den eng. manual appendix VII). Lav en inddeling på akserne som vist i fig. 3 - en enhed på x-aksen svarer til 1/100 sekund.



(Fig. 5) Eksempel på volume envelope.

Kurven inddeles i sektioner. Der kan være op til 5, der skal være en vandret linie. Inddel derefter hver sektion i trin (step count - trin-tæller), der tildeles en værdi kaldet pausetid.
Trinene vil desuden have en stigning eller et fald i lydstyrke kaldet trinstørrelse (step size). Hvis den ikke er specificeret i en sektion vil den foregående blive bragt videre.

at begyndelses- og slutvolumen på en tone er nul, hvorfor den grundlæggende volumen i SOUND kommandoen bør være nul, hvorefter al kontrol af lydstyrken overlades til volume envelope funktionen.
Kommandoen formuleres således:

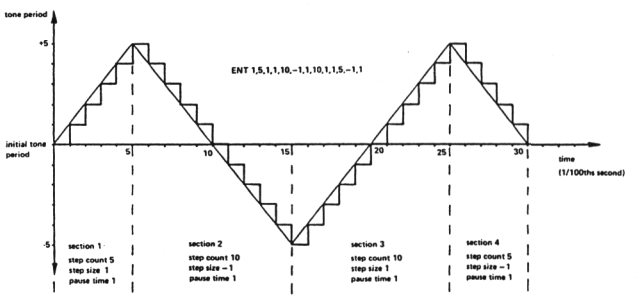
```
ENV N,P1,Q1,R1,P2,Q2,R2,P3,Q3,R3,P4,Q4,R4,P5,Q5,R5
```

hvor

```
N       : Envelope nummer           i området    1...15
P1,..,P5: Step Count (sektion 1-5) i området    0...127
Q1,..,Q5: Step size  (sektion 1-5) i området -128...+127
R1,..,R5: Pause tid  (sektion 1-5) i området    0...255
```

Det anbefales at anvende alle sektioner, men ønskede sektioner kan dog udelades.

Tone envelope (ENT) er af samme type konstruktion som volume envelopen, men danner ganske små variationer i tonens frekvens i et forhold mellem tone-periode og tiden - en form for vibrato. Det ses af fig. 6, at der er tale om en tilnærmet sinuskurve.



(Fig. 6) Den tilstræbte sinuskurve i tone envelopen.

Forskellen på ENV og ENT er at tone envelopen ikke har nogen indvirkning på længden af en tone, der er defineret tidligere, enten i SOUND eller ENT kommandoen. Hvis ENT afslutter før tonen er afsluttet kaldes den konstante tone i SOUND kommandoen automatisk op og forsætter til afslutningen af den fastsatte tid. På den anden siden droppes de resterende trin i ENT, hvis tiden for tonen overskrides.
For at gentage en tone, anvendes et negativt envelope nummer. Bemærk, at ENT ikke kaldes fra SOUND kommandoen med et negativt tal.

Kommandoen formuleres:

```
ENT S,T1,V1,W1,T2,V2,W2,T3,V3,W3,T4,V4,W4,T5,V5,W5
```

hvor

```
S       : Envelope nummer           i området    1...15
T1,..,T5: Step Count (sektion 1-5) i området    0...239
V1,..,V5: Step Size  (sektion 1-5) i området -128...+127
W1,..,W5: Pause tid  (sektion 1-5) i området    0...255
```

Pause tiden beregnes i 1/100 sekunder.
Når et envelope nummer er defineret, overskrives tidligere værdier.
Hvis der specificeres en envelope uden [illegible] bestemt, vil
let. ligere definerede envelope med samme nummer blive 0-stil

3.5 GRAFIK.

Den mest enkle form for grafik opnås med en PRINT CHR$
eksemplet under CALL i BASIC NØGLEORD.
Grafikskærmen har en opløsning på 640 punkter vandret (
talt - x-aksen) og 400 punkter lodret (vertikalt - y akse
x,y koordinaterne har nul-punkt (0,0) i nederste venstre
hvor den grafiske cursor befinder sig ved opstart. I mo
til koordinaterne, der bruges til karaktererne, æn
grafiske koordinater sig ikke, når der skiftes skærmforma

```
10 FOR n=2 TO 0 step -1
20 MODE n
30 PLOT 320,200
40 FOR i=1 TO 1000:NEXT i
50 NEXT n
```

Som du ser, flytter punktet sig ikke, men det bliver stør
opløsningen bliver mindre. Opløsningen i de 3 skærmformat

```
MODE 0   -   160 x 200 punkter
MODE 1   -   320 x 200 punkter
MODE 2   -   640 x 200 punkter
```

Prøv på lignende måde at indtaste

```
10 CLS
20 FOR y=10 TO 100 STEP 10
30   FOR x=1 TO 200
40     PLOT 190+y+100*SIN(x),120+y+100*COS(x)
50   NEXT
60 NEXT
```

Bemærk udeladelsen af variablen efter NEXT. Den er ikke
dig, men hvis der er mange FOR-NEXT løkker, kan programm
tigt blive uoverskueligt. Ved at lave den viste indryk
programmet dog laves mere overskueligt.
Prøv først at ændre PLOT-kommandoen i linie 40 til
derefter indsætte en ny linie:

```
15 DEG
```

og derefter prøve begge kommandoer igen.

Med kommandoen ORIGIN kan du placere den grafiske cursor,
skærmen, du måtte ønske det - du kan altså selv bestemm
dine tegninger skal ligge. Prøv at indsætte

```
5 ORIGIN 100,50
```

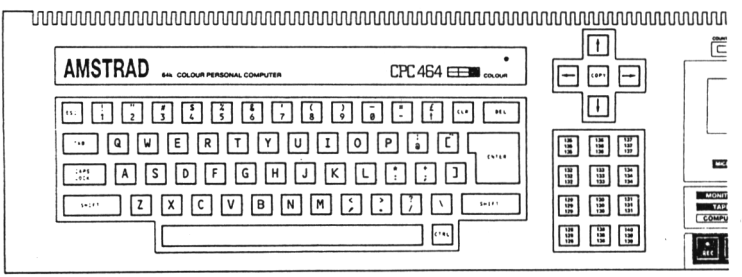
i ovenstående program.

Dette er det grundlæggende i grafikken. DRAW-kommandoen trækker en lige streg fra den sidste position, den grafiske cursor befandt sig i (cursoren står der, hvorfra den næste grafiske operation vil begynde), men er som nævnt ikke synlig. Med kommandoerne

PRINT XPOS og PRINT YPOS

vil du kunne får udskrevet, hvor respektive x og y koordinaten befinder sig.

Du skal være opmærksom på, at hvis der skal udskrives tekst efter grafikken, vil teksten blive skrevet nederst på skærmen, og få denne til at rulle opad incl. den grafiske skærm. Dette kan undgås ved at formattere udskriften, dvs. placere den alm. cursor et passende sted på skærmen med LOCATE, og evt. anvende transparante farver.



(Fig. 7) De definerbare taster, deres startværdi og placering.

exp char	value	ascii
128	0	30
129	1	31
130	2	32
131	3	33
132	4	34
133	5	35
134	6	36
135	7	37
136	8	38
137	9	39
138	.	2E
139	[enter]	0D
140	run"	52,55,4E,22,0D

4.1 MERE BASIC.

Indtast det følgende program, nøjagtig som det er opstillet. Det er meget væsentligt i PRINT-sætningerne, der til tilpasset skærmformat, MODE 1. Du kan selv tilpasse teksten til de andre skærmformater.

```
100 REM Program for nummerologisk karak-
       terbedømmelse.
110 GOSUB 690:MODE 1
120 CLS:GOSUB 670
130 s=0:b=0
140 INPUT "Dit navn";a$
150 n$=UPPER$(a$)
160 FOR n=1 TO LEN (n$)
170 IF MID$(n$,n,1)=" " THEN  s=s+1
180 b=b+ASC(MID$(n$,n,1))
190 NEXT n
200 z=b-s*32-((LEN(n$)-s)*64)
210 b=INT(z/100)
220 c=INT((z-b*100)/10
230 d=INT(z-b*100-c*10)
240 e=b+c+d
250 IF e<=9 THEN 300
260 f=INT(e/10)
270 g=e-f*10
280 e=f+g
290 GOTO 250
300 PRINT:PRINT
310 PRINT "Subjektets nummer er: ";e
320 PRINT "Dine hovedkarakteristika er:"
:PRINT:PRINT
330 ON e GOSUB 350,380,410,440,470,500,5
30,560,590
340 GOTO 620
350 PRINT "I bedste fald er du enormt en
ergisk, mensom regel er du kontrær på en
 meget ke- delig måde."
360 PRINT "Du er individualist med udpræ
get enspo- ret tankegang og et massivt e
go. Du er  samtidig en bemærkelsesværdig
 og uudhol-delig person."
370 RETURN
380 PRINT "Du har udprægede feminine kar
akteristikaog er meget reserveret og vil
lig til at gå på akkord."
390 PRINT "Du kan virke meget stille, me
n du er al-tid parat til at smede ænker
. Du kan  være meget sød og du har fakt
isk en ro- lig side."
400 RETURN
410 PRINT "Du har en livlig karakter, me
n det er  måske fordi du er meget livli
g. Du har  held med, hvad du foretager d
ig."
420 PRINT "Du er skapsinding, intellektu
el og vit- tig. Du taler med charme og e
r en virke-lig vinder (Bvadr!)
430 RETURN
440 PRINT "Du er både uheldig, sløv og i
```

```
deforladt -foruden du er selvhævdende og
respektløs."
450 PRINT "Der er lidt af en 'samfundsst
øtte' i digmen det er meget afhængigt af
 din egen  medvirken."
460 RETURN
470 PRINT "Du er en god (om end ikke tro
fast) sen- gepartner. Du er attraktiv og
 energisk  og har mange interesser og ta
lenter."
480 PRINT "På grund af din utålmodighed
kan du værefarlig at kende, men sjov at
betragte."
490 RETURN
500 PRINT "Du er et ideal af et harmonis
k hjemme-  menneske med et roligt gemyt.
 Du er fairmen meget langsom til at angr
e."
510 PRINT "Du er en god ægtefælle, en sa
mvittig-  hedsfuld forældre og en loyal
 ven. Du ersjov at være sammen med - hvi
s du kan  klare de andres sludren."
520 RETURN
530 PRINT "Du er både hemmelighedsfuld,
reserveret,diciplineret - og arrogant. D
u optræder mystisk, ofte skjulende noget
."
540 PRINT "Du kan være ulykkelig, verden
sfjern el- ler desillussioneret. Du er o
fte bitter,ringeagtende og sarkastisk."
550 RETURN
560 PRINT "Du er fast besluttet på succe
s i livet. Du er rå, smalsporet men vind
er og taber'stort'."
570 PRINT "Du er af og til gerrig og meg
et materia-listisk indstillet."
580 RETURN
590 PRINT "Du er en viis rådgiver, spiri
tuel leder,efterforsker og sandhedslærer
. Idealist i ordets største betydning."
600 PRINT "Du er stærkt lidenskabelig og
 besat af  en stædig vilje. Du er impuls
iv, roman- tisk og i det hele taget en b
mærkelses- værdig personlighed."
610 RETURN
620 PRINT:PRINT
630 INPUT "Vil du gøre flere studier (J/
N) ";s$
640 IF s$="J" THEN 650 ELSE IF s$="N" TH
EN END ELSE 660
650 PRINT "Ja, ja da!":FOR n=1 TO 500:NE
XT:GOTO 120
660 PRINT:PRINT "Kender du ikke forskel
på STORE og små bogstaver?":GOTO 620
670 PRINT "Du indtaster dit fulde navn,
og på dettegrundlag udskriver AMSTRAD CP
C 464 en  karakterbeskrivelse af dig."
680 PRINT:PRINT:RETURN
690 RESTORE
700 SYMBOL AFTER 90
```

```
710 FOR i= 1 TO 6
720 READ sy,k,l,m,n,o,p,q,r
730 SYMBOL sy,k,l,m,n,o,p,q,r
740 NEXT i
750 RETURN
760 DATA 91,0,0,236,18,124,144,110,0
770 DATA 123,62,80,144,252,144,144,158,0
780 DATA 93,48,0,120,12,124,204,118,0
790 DATA 125,24,36,126,102,102,126,102,0
800 DATA 92,0,2,124,206,214,230,124,128
810 DATA 96,58,108,206,214,230,108,184,0
```

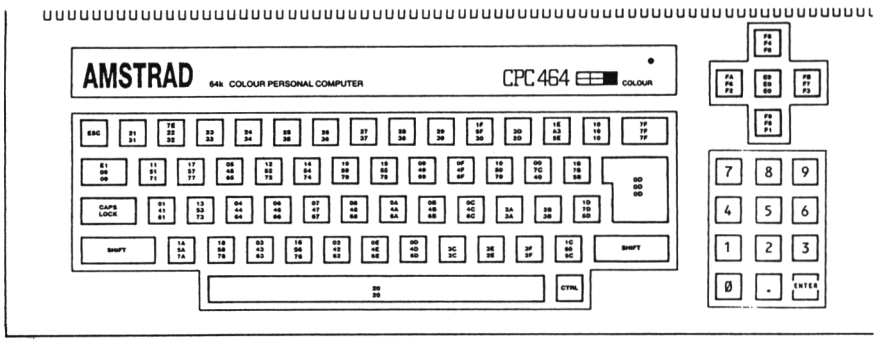
Programmet præsenterer brugen af subrutiner, der kan anvendes når en bestemt operation skal anvendes flere gange, eller som i dette program, hvor det ville være meget upraktisk med PRINT sætningerne midt i programmet.
Ligeledes er det danske karaktersæt lagt i en subrutine, der udføres første gang programmet bliver kørt (GOSUB 690 i linie 110). En GOSUB kommando skal altid efterfølges af RETURN (findes til denne rutine i linie 750) når subrutinen er udført, hvorefter programudførelsen hopper tilbage og udfører den kommando, der følger umiddelbart efter GOSUB ordren (her skift af skærmformat). I linie 330 findes en variant af subrutinen. Variablen e antager i programforløbet skiftende værdier mellem 1 og 9. Denne værdi er afgørende for, hvilken subrutine, der vælges ud af de 9 mulige rutiner. Hvis f.eks. e=3 anvendes subrutinen i linie 410. Variablens værdi er altså afgørende for hvor langt der tælles mod højre. Også her skal der afsluttes med RETURN.

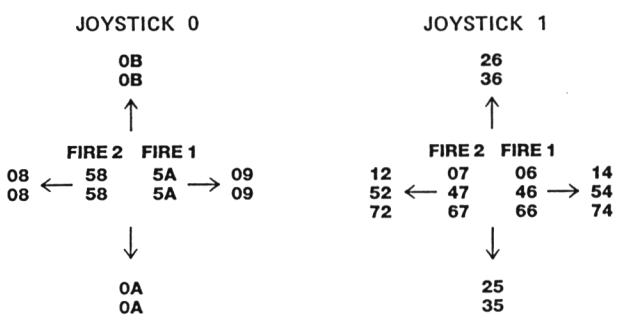
Funktionen UPPER$(strengudtryk) i linie 150 konverterer alle små bogstaver til store (den ændrer ikke på bogstaver, der er skrevet med stort. Den modsvares af LOWER$(strengudtryk) der konverterer til små bogstaver.

Det danske karaktersæt indlæses på en anden måde end vi har set tidligere, idet vi har lagt talkonstanterne ud i DATA-sætninger (linie 760 - 810), adskilt med kommaer. De kunne udmærket have ligget i den samme DATA-sætning, men det hjælper på overskueligheden på denne måde. For hvert gennemløb af FOR-NEXT løkken læses (READ) de variabler, der er tilknyttet READ-sætningen, hvorefter de "tilegnes" symbolkommandoen i linie 730.
RESTORE i linie 690 "vækker" DATA-sætningerne, da de ellers kun ville kunne læses en gang.
DATA-sætninger indeholder kun nummeriske data her, men kan også læse strengvariabler når blot variablen tillægges $ i READ-sætningen, og nummeriske- og strengvariabler kan blandes efter behag.

Programmet bygger iøvrigt på en ide om at tillægge bogstaverne en talværdi (A=1,B=2 etc.) som vi finder tværsummen af, indtil vi har en værdi, der ligger mellem 1 0g 9. F.eks. er "Kim" = 11 +9+13 = 33 og 3+3 = 6.
Programmets hjerte er linierne 130 - 290. Testpersonens navn INPUTtes i a$ (inclusiv mellemrum, men uden bindestreger) og konverteres til store bogstaver for kun at skulle beregne på det ene karaktersæt. Linie 170 tæller, hvor mange mellemrum der er i variablen s . Linie 180 adderer ASCII-værdien af bogstaverne i variablen b (værdien findes ved hjælp af funktionen ASC(strengudtryk)). ASCII-værdierne for store bogstaver begynder

ved 65, hvorfor værdien af ASC() fratrækkes 64 for at få A=1, og summen af disse tal gemmes i variablen z . Linierne 210 - 240 uddrager de enkelte cifre af z og adderer dem igen, indtil vi har et ciffer mellem 1 og 9.





(Fig. 8) ASCII karaktererne og deres placeringer, samt værdierne for joy-stikkene.

5.1 SPECIALFUNKTIONER.

Amstrad CPC 464 indeholder det, der kaldes for et "real time" ur, og nogle funktioner der kan styre subrutiner afhængig af den tid, der er forløbet.
"Uret" er quartz-styret og kontrollerer, hvad der sker i computeren, så som at scanne tangentbord, skærm, skifte mellem ROM og RAM og styre Z80A processoren.

Der findes 4 timere (tidsstilbare funktioner), der hver især kan få en subrutine tilknyttet. Den første kommando hertil er:

AFTER %tal (,%tal) GOSUB linienummer

Det første heltal angiver, hvor lang tid, der skal gå, før subrutinen skal kaldes og beregnes i enheder på 1/50 sekund.
Det andet heltal specificerer, hvilken af de 4 timere, der skal anvendes. Det skal være en værdi mellem 0 og 3 (hvis denne værdi ikke opgives, formoder computeren, at det drejer sig om timer 0.

Når den specificerede tid er gået, bliver subrutinen kaldt automatisk nøjagtig som om at der var mødt en subrutine-linie i programmet på dette tidspunkt. Når subrutinen slutter (der skal anvendes en normal RETURN kommando), vender det tilbage til det punkt i hovedprogrammet, hvor det blev afbrudt (interrupted).

Timerne har forskellige interrupt prioriteter, hvor timer 3 har den højeste, og timer 0 den laveste.
AFTER kommandoer kan opstilles hele tiden i programmet (eller fornyes om nødvendigt). Timerne er de samme, som anvendes af EVERY kommandoen, så en AFTER kommando vil overskrive en tidligere skrevet EVERY og omvendt.

```
10 MODE 1:x=0
20 AFTER 45 GOSUB 100
30 AFTER 100,1 GOSUB 200
40 PRINT "AMSTRAD"
50 WHILE x<100
60 LOCATE #1,30,1:PRINT #1,x:x=x+1
70 WEND
80 END
100 PRINT "CPC 464"
110 RETURN
120 PRINT "Copyright"
130 RETURN
```

Bemærk brugen af to streams (vinduer - begge har her hele skærmen som område) for at tillade udskrift fra hovedprogrammet (linie 50-80) og interrupt subrutinerne samtidig. Efter den tid, der er specificeret i kommandolinierne udfører programmet subrutinerne.

Med EVERY kommandoen er det muligt at få BASIC'en til at kalde en subrutine i faste intervaller. Betingelserne for brugen af kommandoen er de samme som for after, og lyder:

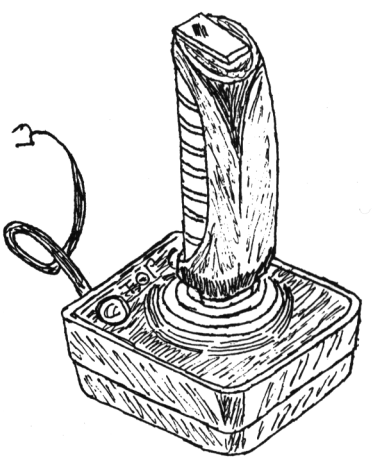
EVERY %tal (,%tal) GOSUB linienummer

I det følgende program anvendes DI (Disable Interrupt - afbryd interrupt) for at evt. andre interrupt-rutiner ikke skal gribe

forstyrrende ind, mens den øjeblikkelige rutine udføres og derefter anvendes EI (Enable Interrupt - tillad interrupt). Dette sætter en pause på den højere prioriteret timer 1, så timer 0 kan afslutte sine funktioner.

```
10 MODE 1:x=0
20 p1=0:EVERY 10 GOSUB 100
30 p2=0:EVERY 12,1 GOSUB 200
40 PRINT "AMSOFT"
50 WHILE x<200
60 LOCATE #1,30,1:PRINT #1,x:x=x+1
70 WEND
80 LOCATE 1,20:END
100 DI:PEN p1:LOCATE 1,2:PRINT "tilbehør":EI
105 IF p1=0 THEN p1=1 ELSE p1=0
110 RETURN
200 PEN p2:LOCATE 1,3:PRINT "og software"
205 IF p2=2 THEN p2=3 ELSE p2=2
210 RETURN
```

Funktionen REMAIN kan anvendes i forbindelse med de to kommandoer. Funktionen returnerer den tilbageværende tid for en af de 4 timere og afbryder en pågældende, eller returnerer 0, hvis timeren allerede er afbrudt.



6.1 TILSLUTNINGER.

Der kan tilsluttes joystik til CPC 464 til brug for spil med denne facilitet (med indbygget "fire"-knap (skyd). Joystikket, model JY1 tilsluttes på på computeren i det 9-polede stik mærket USER PORTS (I/O) (I/O står for In/Out - ind/ud), og hvis det er nødvendigt, kan der tilsluttes endnu et stik i soklen på det første joystik. Skal porten bruges til andre formål, er stikforbindelserne vist i appendix V i den eng. manual.
Joystikkene behandles som en del af tangentbordet og kan "kopieres" på tangenterne med INKEY$ og INKEY. Bemærk, at hvis der kun er en "fire"-knap på dit joystik vil det formentlig være "Fire 2" i Amstrad terminogien.
Funktionen JOY (n) kan kontrollere joystikket direkte. JOY(0) for det første og JOY(1) for det andet. Funktionen returnerer et bit signifikant resultat, som indikerer hvorledes kontakterne i joy-stikket står under det seneste tangentbords-scan, der foretages 50 gange i sekundet.
Joy-stikket returnerer værdier som følger, hvor KEY er værdien, der skal bruges i en INKEY funktion og spejl er den tilsvarende tangentbordsværdi.

1. stik	JOY(0)	KEY	2. stik	JOY(1)	KEY	Spejl
op	Bit 0	72	op	Bit 0	48	6
ned	Bit 1	73	ned	Bit 1	49	5
venstre	Bit 2	74	venstre	Bit 2	50	R
højre	BIT 3	75	højre	Bit 3	51	T
fire 2	Bit 4	76	fire 2	Bit 4	52	G
fire 1	BIT 5	77	fire 1	Bit 5	53	F

Når joystik 2 er tilsluttet kan computeren ikke "se" forskel på tangentbord og joy-stik. Det er usansynligt at der vil ske fejl, så tangentbordet skulle kunne bruges som substitut for stik 2.
Ben 9 på stikket i porten accepterer standard joy-stik af andre fabrikater, men giver ikke mulighed for tilslutning af stik 2.
Skriver du selv spil til dette brug, bør du give brugeren en mulighed for at vælge mellem stik og tangenter (brug COPY-knappen som "fire".

Printer-tilslutningen har interface efter 'Centronics' industristandard. Kablet er forsynet med 2 stik, hvoraf det der monteres i computeren har 2 "fingre" færre end printer-stikket.
Det er ikke nødvendigt med specielle kommandoer for at starte printeren op, og der udprintes direkte med kommandoerne LIST #8 og PRINT #8.
Mange printere vil automatisk lave vognretur, når slutningen på print-linien er nået. Det kan sættes med kommandoen WIDTH, der har en startværdi på 132, men kan f.eks. ændres til 40, 36 eller 80. Hvis den sættes til 255 vil BASIC ikke foretage vognreturen, men overlade denne helt til printeren. BASIC indeholder en tæller for printerpositionen, der kan anvendes med POS funktionen:

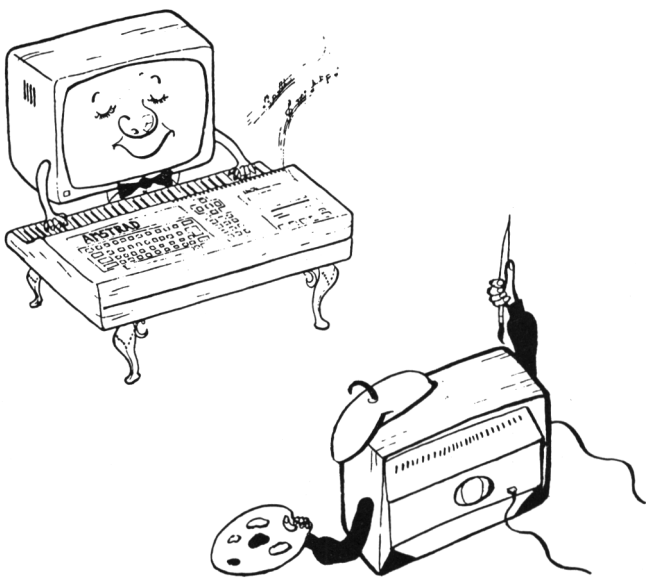
```
IF POS(#8)50 GOTO 100
```

CPC foretager et linieskift CHR$(10) og en vognretur CHR$(13) vedslutningen af linien. Printeren vil almindeligvis indeholde en omskifter for indstilling af den nødvendige form for indlæsning.

Der vil komme udvidelses ROM-moduler i 16K blokkes format. De vil optage de øverste 16K, hvor BASIC befinder sig, og det vil blive muligt at tilslutte op til 240 ekstra blokke.

De forskellige streams har flg. funktioner:

Stream #	Indlæsning	Udlæsning
0-7	Tangentbord	Skærm
8	Tangentbord	Printer
9	Kassette	Kassette



7.1 TALSYSTEMER.

Almindeligvis regner vi i decimaltals systemet, 10-tals systemet. Computeren arbejder imidlertid med andre talnotationer.

Opbygningen af 10-talssystemet sker med to typer elementer:

1) Grundtal: 10
2) Cifre: 0,1,2,3,4,5,6,7,8,9

Inden for databehandling anvendes:

- det binære talsystem (grundtal: 2)
- det hexadecimale talsystem (grundtal: 16)
- det oktale talsystem (grundtal: 8)

Det binære talsystem anvendes internt af computeren, hvilket vil sige at alle informationer i copmputeren optræder binært (1 eller 0 - tændt eller slukket).
Det binære talsystem har elementerne:

1) Grundtal: 2
2) Cifre: 0,1

Et tal, der binært er repræsenteret ved 1101 kan omregnes til decimaltal ved

$$1*2\uparrow3+1*2\uparrow2+0*2\uparrow1+1*2\uparrow0 = 8+4+0+1 = 13$$

eller generelt for en 8 bits notation som den anvendes i CPC 464

$2\uparrow7$	$2\uparrow6$	$2\uparrow5$	$2\uparrow4$	$2\uparrow3$	$2\uparrow2$	$2\uparrow1$	$2\uparrow0$	Værdien
* +	* +	* +	* +	* +	* +	* +	*	=af 8 bits
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	(1 byte)

hvor bit 7 er den mest betydende og bit 0 den mindst betydende.

8 bits = 1 byte
1 Kbyte = 1024 bits

Hvilket vil sige, når der er tale om 64K, er der (1024*64)bits = 65536 bits til rådighed.

Det oktale system har

1) grundtal: 8
2) cifrene: 0,1,2,3,4,5,6,7

og beregnes på lignende måde som det binære, blot er det grundtallet 8 der opløftes i tilsvarende potenser efter antal cifre i tallet.

Det samme gør sig gælden for det hexadecimale system, der har

1) grundtal: 16
2) cifrene: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

hvor bogstaverne fra A mod højre tillægges værdierne 10, 11, 12, 13, 14 og 15.

Her er det grundtallet 16, der opløftes i en potens.

Når der summes op i hex-notation foregår det således:

0 1 2 3 ... 9 A B C D E F 10 (udtales en - nul) 12 (en - to)...1E 1F 20 21...2F 30 (30 = 3*16↑1+0*16↑0).

Iøvrigt kan du anvende CPC 464's indbyggede BIN$() og HEX$() funktioner.

7.2 MATEMATISKE FUNKTIONER.

Når der tændes for computeren er den parat til direkte indlæsning, og kan bruges direkte som kalkulator, eller regnefunktionerne kan lægges ind i programmerne.
Bemærk at decimalskilletegnet er punktum (.).

De 4 almindelige regnearter:

```
?3+5          ?9-5          ?4*3          ?12/4
 8             4             12            3

              ?4-9                        ?12/0
              -5                          11 Division by zero
```

Addition Subtraktion Multiplika. Division

Kvadratrod:

```
?SQR(25)
 5
```

Potensopløftning:

```
?3↑3   (3*3*3)        ?3↑13
 9                     1594323 (antal kombinationer
                               på tipskuponen)
```

Kubikrod:

```
?27↑(1/3)
 3
```

E-notation:

500 kan også skrives som 3*10↑2 - i notation som 5E2
0.03 kan også skrives som 3*10↑-2 - i notation som 5E-2

Der kan anvendes stort eller lille E (e).

7.3 LOGISKE OPERATORER.

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

To halvdele af et logisk udtryk er argumenter, og udtrykket ser således ud:

```
argument (LOGISK OPERATOR argument)
```

hvor

```
argument (er) NOT argument
              ... NOT nummerisk udtryk
              ... NOT relativt udtryk
              ... NOT (logisk udtryk)
```

begge argumenter for en logisk operator forceres til at arbejde på heltal (%tal), og resulterer i Error 6 hvis argumentet ikke passer ind i heltalsområdet.

Operatorerne arbejder på den binære repræsentation, hvilket kan give resultatforskelle, selv om decimaltallene ser ens ud. Deres effekt på hver bit er

```
AND  resulterer i 0 med mindre begge argument bits er 1
OR   resulterer i 1 med mindre begge argument bits er 0
XOR  resulterer i 1 med mindre begge argument bits er ens
```

And er den mest almindeligt forekomne logiske operator og den betyder ikke "addere".

```
PRINT 10 AND 10
 10

PRINT 10 AND 12
 8

PRINT 10 AND 1000
 8
```

De sidste to resultater skyldes, den binære repræsentation:

```
      1010
1111101000
```

Operationen undersøger hver korresponderende bit parvis, og hvor bittene i øverste og nederste række er 1, er svaret 1:

```
0000001000
```

som konverteret til decimalsystemet giver resultatet 8.
AND undersøger hvornår to tilstande er til stede samtidig.

OR arbejder på at resultatet er 1, med mindre begge bits fra argumenterne er 0, i hvilket tilfælde resultatet er 1. Med samme taleksempel:

```
PRINT 1000 OR 10
 1002
```

eller i binær repræsentation

```
      1010
1111101000
```

hvilket giver resultatet:

```
1111100010
```

Endelig producerer XOR (eXclusive OR) et sandt resultat, så længe begge argumenter er forskellige.

Sammenhængen mellem de 3 operatorer kan stilles op skematisk:

argument A	1	0	1	0
Argument B	0	1	1	0
AND resultat	0	0	1	0
OR resultat	1	1	1	0
XOR resultat	1	1	0	0

8.1 BASIC NØGLEORD.

Det følgende afsnit er en oversigt over de nøgleord (keywords), der indgår i Amstrad BASIC. Nøgleord er reserverede udtryk, der anvendes af operativsystemet i computeren, og kan ikke anvendes som variabelnavne, men kan indgå som en del af et variabelnavn. Oversigten, der er i alfabetisk orden, viser

NØGLEORDET
Syntax/funktion
Beskrivelse
Eksempler
Beslægtede nøgleord

Der benyttes flg. notation:

& eller H	- prefix for hexadecimal konstant (se 7.1)
&X	- prefix for binær konstant
#	- prefix for en stream
:	- skilletegn for flere operationer på samme linie

Datatyper:

Datastrenge (kendetegnet ved $) kan have en længde fra 0 (strengen er tom - A$="") til 255 karakterer. Datastrenge kan forlænges med + (A$=A$+B$), hvis den samlede længde ikke overstiger 255 karakterer.

Nummeriske data (0,1...9) kan være reelle tal(real - dvs. decimaltal) eller heltal (INTeger - dvs. ingen decimaler).
De reelle tal ligger i området +-1.7E+38 med den mindste værdi over nul på ca. 2.9E-39, og beregnes med lidt mere end 9 cifres nøjagtighed. Reelle tal kendetegnes ved et foranstillet ! .
Heltallene ligger i området -32768 til 32767, og kendetegnes ved et foranstillet % (%tal).
Et nummerisk udtryk er en hvilken som helst operation, der giver en talværdi som løsning på en regneoperation.

Et stream udtryk defineres ved #(nummerisk værdi) og henviser til bestemte funktioner på skærmen, printer eller kassettebåndoptager.

Nøgleordene er enten KOMMANDOer (commands), der udføres direkte eller FUNKTIONer (functions), dvs. operationer der har et efterfølgende argument (evt. variabel) i parantes.

Nøgleordene kan skrive enten i store eller små bogstaver. I den efterfølgende liste er de alle skrevet med stort for overskuelighedens skyld. Hvis de skrives med småt, vil BASIC'en i CPC 464 selv ændre nøgleordene til stort, når programmet LISTes.

Nøgleordene skal være omgivet af et mellemrum på hver side, da disse reserverede ord kan indgå i variabelnavne, som f.eks. RUNNAME.

ABS

```
ABS(nummerisk udtryk)

PRINT ABS (-260.44)
 260.44
```

Funktion: Returnerer den absolutte værdi af et nummerisk udtryk, hvilket vil sige at negative tal returneres som positive.

Se også: SGN

AFTER

```
AFTER %tal(,%tal) GOSUB linienr.

AFTER 360,3 GOSUB 1110
```

Kommando: Udfør en given subrutine efter forløbet af en vis periode. Det første %tal angiver hvor ofte subrutinen skal kaldes i enheder på 0.02 sekund og det andet %tal (0, 1, 2 ell. 3) angiver hvilken af de 4 timere, der skal bruges.

Se også: EVERY, REMAIN

ASC

```
ASC(strengudtryk)

PRINT ASC("M")
 77
```

Funktion: Tager den nummeriske værdi af den første karakter i en streng (på ASCII-karaktererne).

Se også: CHR$

ATN

```
PRINT ATN(nummerisk udtryk)

PRINT ATN(1.5)
 0.982793723
```

Funktion: Beregner arc-tangens af det opgivne udtryk. Resultatet er et reelt tal i RADianer i området PI/2 til PI/2.

Se også: SIN, COS, TAN, DEG, RAD

AUTO

```
AUTO linienr.,spring
```

```
AUTO 100,25
```

Kommando: Danner automatisk det næste linienummer efter der er trykket på ENTER. Hvis linienr. og spring ikke defineres startes automatisk med linie 10 og der springes 10. Der kan brydes ud af AUTO ved at trykke på ESC. Skal der fortsættes med AUTO skrives linienr. på den første nye linie, og hvilket spring, der skal tages.
Hvis der dannes et nyt linienummer, der allerede eksisterer, indsætter BASIC'en en stjerne (*) efter det generede linienr. for at henledes opmærksomheden herpå.

BIN$

```
BIN$(positivt %tal(,%tal))

PRINT BIN$(77,8)
 01001101
```

Funktion: Returnerer en streng med binære cifre der repræsenterer værdien af det positive %tal. Der udfyldes med foranstillede nuller til det antal cifre der angives ved %tal (=16).

Se også: HEX$, STR$

BORDER

```
BORDER farve,farve

BORDER 3,9
```

Kommando: Skifter farve på skærmkanten. Hvis der kun opgives 1 farve, skiftes til denne. Opgives to farver, vil kanten skifte mellem disse to i blink, hvis hastighed kan styres af SPEED INK. Udtrykket farve er et %tal i området 0 til 26.

Se også: SPEED INK

CALL

```
CALL adresse,parametre

10 CLS
20 FOR n=1 to 39:FOR i=39 TO 1 STEP -1
30 LOCATE n,18
40 CALL &BD19
50 PRINT " ";CHR$(250)
60 LOCATE i,20
70 CALL &BD19
80 PRINT CHR$(251);" "
90 Next i:Next n
100 GOTO 5
```

Kommando: Kalder en maskinkoderutine fra BASIC. Rutinen kan være

i ROM, eller selvkonstrueret rutine i RAM. &BD19 kalder en rutine der venter på den næste scanning billedrørets elektroner foretager og gør bevægelse mere glidende. Programmet kan forbedres ved at indlægge nogle FOR-NEXT pauserutiner.

Se også: UNT

CAT

CAT

ARKIVNAVN Block nr. Flag Ok

Kommando: Får båndoptageren til at katalogisere indholdet på et bånd uden at læse programmerne ind. Filnavn er navnet, programmet er SAVEd under. Flag indikerer hvilken slags fil, der er tale om:

$ - et ubeskyttet BASIC program
% - et beskyttet BASIC program
& - en ubeskyttet binær fil
' - en beskyttet binær fil
* - en ubeskyttet ASCII tekst fil (kan ikke beskyttes)

Andre karakterer kan forekomme, hvis filen ikke er produceret af et BASIC program.

Se også: LOAD, RUN, SAVE

CHAIN

CHAIN filnavn,linie nr.

CHAIN "MOLINT", 300

Kommando: Læser et program ind fra kassette, der erstatter det eksisterende program. Hvis linienr. opgives, vil programmet starte udførelsen fra denne linie, ellers fra laveste linienr.

Se også: CHAIN MERGE

CHAIN MERGE

CHAIN MERGE filnavn,linie nr.,DELETE linienr. område

CHAIN MERGE "MOLINT",300

Kommando: Lægger et program ind i forlængelse af et eksisterende der ikke må have samme linienumre. Linie nr. angiver hvilken linie programmet starter udførelsen fra. Hvis dette ikke opgives, starter programmet fra laveste linienummer. Hvis DELETE angives, vil linierne i området blive slettet.
Hvis der ikke opgives noget filnavn, vil CHAIN MERGE indlæse det første program det møder.
CHAIN MERGE påvirker ikke variabler der ligger i hukommelsen,

men brugerfunktioner og åbne filer undertrykkes. ON ERROR GOTO "lukkes", RESTORE generes og DEFINT, DEFREAL og DEFSTR nulstilles, medens alle aktive FOR, WHILE og GOSUB "glemmes".
En beskyttet fil kan ikke merges.

Se også: DELETE, LOAD, MERGE

CHR$

```
CHR$(%tal)

PRINT CHR$(77)
M
```

Funktion: Konverterer en nummerisk værdi i området 32 til 255 til en tilsvarende karakter (Appendix III i eng. manual).

Se også: ASC, LEFT$, RIGHT$, MID$, STR$

CINT

```
CINT(nummerisk udtryk)

10 A=1576.3313
20 PRINT CINT(A)
RUN
 1576
```

Funktion: Afrunder den opgivne værdi til nærmeste %tal i området -32768 til 32767.

Se også: CREAL, INT, FIX, ROUND, UNT

CLEAR

```
CLEAR
```

Kommando: Nulstiller alle variabler og filer.

CLG

```
CLG maskeret INK

CLG
```

Kommando: "Renser" den grafiske skærm.

Se også: CLS, ORIGIN

CLOSEIN

```
CLOSEIN

CLOSEIN
```

Kommando: Lukker kassetteindlæsningsfilen.

Se også: OPENIN, CLOSEOUT

CLOSEOUT

```
CLOSEOUT

CLOSEOUT
```

Kommando: Lukker kassetteudlæsningsfilen.

Se også: OPENOUT, CLOSEIN

CLS

```
CLS #stream

CLS
```

Kommando: "Renser" et givent skærmvindue til dets papirfarve. Hvis der ikke opgives #stream forudsættes #0.

Se også: CLG

CONT

```
CONT

CONT
```

Kommando: Fortsætter programudførelsen efter *BREAK*, STOP eller END, så længe der ikke er foretaget rettelser i programmet - der kan dog gives direkte ordrer til computeren.

COS

```
COS(nummerisk udtryk)

PRINT COS(10)
 -0.839071529
```

eller

```
DEG:PRINT COS(10)
  0.984807753
```

Funktion: Beregner cosinus af den opgive værdi. Resultatet udlæses i RADianer, hvis ikke der gives kommandoen DEG.

Se også: SIN, TAN, ATN, DEG, RAD

CREAL

```
10 DEFINT n
20 n=1576.73319
30 d=n/110.37
40 PRINT d,
50 PRINT CREAL(n)
60 PRINT n/1504.62
RUN
14.288303     1577
1.04810517
```

Funktion: Konverterer et heltal (her defineret ved DEFINT n) til et decimaltal i modsætning til funktionen INT.

Se også CINT, DEFINT

DATA

```
DATA konstant, konstant,.....,konstant

10 DATA 25,56,Molint,256,1020,jæger
```

Kommando: Definerer konstante data til brug i programforløbet. Anvendes, hvor der ofte kan være brug for at indlæse forskellige datasæt i den samme operation. DATA kan placeres hvor som helst i programmet.

Se også: READ, RESTORE

DEF FN

```
DEF FN navn,(parametre)=generelt udtryk

10 DEF FN a(x,y)=190+y+100*SIN(x)
```

Kommando: Beregninger, der ofte går igen i programmet, kan defineres i DEFine FuNction. De fungerer på samme måde som indbyggede funktioner som SIN og COS.
Definitionen bør ligge i begyndelsen af programmet og ikke i den del af programmet der udføres.
Parametrene viser hvilke variabler, der anvendes i funktionen.
Hver funktion defineres desuden med navn.

Funktionen kaldes med

```
FN navn,(parametre)
```

Se også: DEFINT, DEFREAL, DEFSTR, FN

DEFSTR

DEFtype variabel(-ler)

DEFINT K-O
DEFREAL
DEFSTR M,P-T

Kommando: Definerer nævnte variabler i hele programmet, hvor type er heltal (INTeger), decimaltal (REAL) eller en streng (STRing). Variablen sættes efter det første bogstav i variabelnavnet (store eller små bogstaver). Det er således ikke nødvendigt ved f.eks. variablen K hver gang at specificere den som K%.

Se også: LOAD, RUN, CHAIN, NEW, CLEAR

DEG

DEG

DEG

Kommando: Omstiller computeren til DEGrees (grader), hvor den bliver indtil den gives en af kommandoerne CLEAR eller RAD eller et nyt program indlæses.

Se også: RAD

DELETE

DELETE linienr. område

DELETE 10-400

Kommando: Fjerner det specificerede område fra hukommelsen. Bruges kommandoen ved en fejltagelse, er der ikke andet at gøre, end at taste linierne ind igen, så vær forsigtig.

Se også: CHAIN MERGE, NEW

DI

DI

Kommando: Disable Interrupt (BREAK er også et interrupt, men bliver ikke påvirket af denne rutine) - afbryd interrupt indtil det reetableres med EI eller af RETURN ved en interrupt GOSUB rutine.

Se også: EI, TAG

DIM

```
DIM variabel(),variabel(),...,variabel()
```

```
10 DIM ML$(5),TA(5)
20 CLS
30 FOR n=1 TO 5
40 INPUT "Indlæs et navn";ML$(n)
50 TA(n)=n
60 NEXT
70 FOR n=1 TO 5:PRINT TA(n),ML$(n)
80 PRINT, "Til lykke med din AMSTRAD":NEXT
90 FOR n=5 TO 1 STEP -1:PRINT TA(n),ML$(n)
```

Kommando: DIMensionerer plads for et område (array) i hukommelsen og angiver hvor stort et område (antal variabler), der skal være plads til. Hvis dette ikke ikke specificeres, opstiller BASIC et grundfelt på 10.

Der kan også opsættes matrixer (dvs. områder på flere dimensioner), f.eks.

```
DIM omr(5,5,5)
```

Tænk på den første variabel som en tæller i en tabel - den næste variabel.

Et DIMensioneret må ikke ændres i størrelse under programudførelsen, men indholdet kan slettes med ERASE. Størrelsen af områderne er kun begrænset af den mængde hukommelse, der er til rådighed.

Se også ERASE

DRAW

```
DRAW x-koordinat,y-koordinat,(maskeret INK)
```

```
DRAW 200,300,13
```

Kommando: Trækker en linie til et abolut punkt, der er defineret på den grafiske skærm. Dvs. 200 punkter ud ad x-aksen og 300 punkter op ad y aksen med (maskeret) INK 13.
Koordinat punktet forbliver uændret i de 3 skærmformater.

Se også: DRAWR, MOVE, MOVER, ORIGIN, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS

DRAWR

```
DRAWR x-start,y-start,(maskeret INK)
```

```
DRAWR 200,300,13
```

Kommando: Trækker en linie fra den grafiske cursors nuværende position til en relativ position i forhold til nuværende posi-

tion.

Se også DRAW, MOVE, MOVER, ORIGIN, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS

EDIT

EDIT linienummer

EDIT 310

KOMMANDO: Kald en programlinie frem til redigering (se afsn. 2.2).

Se også: LIST

EI

EI

EI

Kommando: Reetablerer interrupt, der har været undertrykt af en DI kommando.

Se også DI

END

END

END

Kommando: Afslutningen på et program. END er underforstået i BASIC, når sidste programlinie er udført. END lukker alle kassette filer og omstillet computeren til direkte indtastning. Evt. SOUND kommandoer vil fortsætte til lydkommandoen er afsluttet. Ligger END midt i et program, kan programudførelæsen fortsættes med CONT.

Se også CONT, STOP

ENT

ENT envelope nummer (,envelope sektioner,...,...)

```
10 ENT 1,200,2,30
20 SOUND 1,284,1500,4,0,1
```

Kommando: Medens en lyd bliver genereret, er det muligt at variere dens tone (ENT = ENvelope Tone). En tone-envelope definerer hvorledes tonen skal varieres.

Envelope nummer er et %tal i området 1 til 15 og specificerer hvilken envelope, der skal bruges. Hvis envelope nummer er negativt gentages envelopen.

Der kan være op til 5 envelope sektioner, der hver kan tage formen - trintæller, trinspring,pause periode - eller toneperiode, pauseperiode.

Den første form beskriver en trinvis stigning i relation til den øjeblikkelige toneperiode. Den anden form specificerer en absolut tilstand for toneperioden.
Trintæller giver antallet af trin i denne envelope sektion og er et %tal i området 0 til 239.
Trinspring er det tal, der varierer længden af tone perioden på hvert trin i envelopen og er et %tal i området -128 til +127.
Pause periode angiver den tid, der ventes mellem hvert trin, og er beregnes i enheder på 0.01 sekund. Enhederne beskrives ved en værdi i området 0 til 255 (hvor 0 behandles som 256).
Toneperiode angiver giver en ny værdi for perioden og er et %tal i området 0 til 4095.

SOUND kommandoen sætter den grundlæggende toneperiode og kan specificere en af de 15 tone enveloper. Hvis der ikke angives nogen envelope, eller en envelope ikke er specificeret, vil tonen forblive konstant i hele lydperioden.

Se også: ENV, SOUND

ENV

ENV envelope nummer (, envelope sektioner,...,...)

```
10 ENV 1,100,2,20
20 SOUND 1,100,1000,4,1
```

Kommando: Medens en lyd genereres, er det muligt at variere dens lydstyrke (ENV = ENvelope Volume). En envelope sektion består af - trintæller,trinspring,pause periode - eller som - hardware envelope,envelope periode - der specifices ved

Trintæller der giver antal trin i sektionen, defineret i området 0 til 127.
Trinspring er tallet for hvor meget, amplituden (styrken) skal varieres på hvert trin i envelopen, og ligger i området -128 til +127 (%tal) - eller hvis trintælleren er sat til 0 er det den absolutte værdi amplituden sættes til.
Pause periode angiver pausetiden mellem trinene og er et %tal i området 0 til 255 (0 behandles som 256) og er beregnet 1/100 sekunds tidsenheder.
Hardware envelope er den værdi, der sættes ind i envelope formregistret (register 15,oktal - 13,decimal).
Envelope periode er den værdi, der lægges ind i envelope periode registrene (registrene 13 og 14,oktal - 11 og 12,decimal).
Harware enveloperne har ikke en tildelt pause periode. Der bør derfor beregnes en passende pause for næste trin - hvis denne ikke specificeres afsættes en 2 sekunders pause automatisk.

Se også ENT, SOUND

EOF

```
EOF

PRINT EOF
-1
```

Funktion: Tester om kassette indlæsningen er nået til End Of File (slut på fil). Svarer med -1 (sand) hvis filen er færdig, eller med 0 (falsk) (sand/falsk 0 se afsnit 7.3).

Se også: OPENIN

ERASE

```
ERASE variabel (,variabel,..,varabel)

ERASE ML$,TA
```

Når et dimensioneret område ikke mere er nødvendigt, kan det slettes med denne kommando og hukommelsen kan anvendes til andre formål.

Se også: DIM

ERR
ERL

```
ERR
ERL

10 CLS:DIM a$(5)
20 ON ERROR GOTO 1000
30 DATA Amstrad,CPC 464,Personlig,farve,
computer
40 FOR n=0 to 5:READ a$(n):NEXT
50 FOR n=0 to 4:PRINT a$(n);" ";:NEXT
60 END
999 REM Fejlrutine starter her
1000 IF ERR=4 THEN PRINT "Der er en ERRO
R EXHAUSTED fejl..."
1010 IF ERL<70 AND ERL>20 THEN PRINT"..
.i linierne 30-40!"
1020 END
```

Variabler: De bruges i fejlbehandlingsrutiner til at finde ud af hvilken fejlkode der er tale om og linienummeret med fejl. Se fejlkoderne i afsn. 8.3.

Se også: ON ERROR, ERROR

ERROR

```
ERROR %tals udtryk
```

ERROR 35

Kommando: Finder fejlmeddelelse med et givent nummer. Nummeret kan være et af de, der allerede benyttes af BASIC (afsn. 8.3), eller det kan være et højere nummer, der kan benyttes til egne fejlmeddelelser i forbindelse med ERR og ERL.

Se også: ERR, ERL, ON ERROR

EVERY

EVERY %tals udtryk (,%tals udtryk) GOSUB linienummer

EVERY 500,2 GOSUB 50

Kommando: CPC 464 har et real time ("ægte tid") ur indbygget. EVERY kommandoen giver mulighed for at kalde subrutiner med et forudbestemt tidsinterval. Der er 4 timere til rådighed, og de er udtrykt ved 0 til 3 (det andet %tals udtryk). Hver timer kan have en subrutine tilknyttet.
Tiden måles i 1/50 sekund - dvs. første %tal skal være 250, hvis subrutinen skal kaldes hver 3. sekund.

Se også: AFTER, REMAIN

EXP

EXP (nummerisk udtryk)

```
PRINT EXP(9.44062)
 12589.52
```

Funktion: Kalkulerer 'e' til nummerisk udtryk, hvor 'e' er 2.71828183 (den naturlige logaritme til 'e' er 1).

Se også: LOG

FIX

FIX(nummerisk udtryk)

```
PRINT FIX(9.8976)
 9
```

Funktion: I modsætning til CINT, fjerner FIX decimaler og afrunder mod 0.

Se også CINT, INT, ROUND

FOR

```
FOR variabel = startværdi TO slutværdi STEP værdi

FOR MANED=1 TO 12 STEP 2
```

Kommando: Opstiller en "tæller", der udfører den rutine, der ligger mellem FOR og NEXT et specificeret antal gange. FOR-variablen er en simpel, dvs. den må kun indeholde karaktererne A-Z. Hvis STEP værdien ikke opgives, sættes den automatisk til 1. En FOR-løkke skal altid afsluttes med NEXT

Se også: NEXT, WHILE

FN

```
FN navn(parametre)

10 DEF FN a(x,y)=190+y+100*SIN(x)
20 DEF FN b(x,y)=120+y+100*COS(x)
30 CLS
40 FOR y=10 TO 100 STEP 10
50 FOR x=1 TO 200
60 PLOT FN a(x,y),FN b(x,y)
70 NEXT:NEXT
```

Funktion: Udfører den funktion, der er defineret med kommandoen DEF FN.

Se også: DEF FN

FRE

```
FRE(nummerisk udtryk)

FRE(strengvariabel)

PRINT FRE(0)
 43533

PRINT FRE("")
 43533
```

Funktion: Returnerer oplysning om, hvor meget fri hukommelse der er tilbage, når BASIC har taget sin del. Formen FRE("") foretager en garbage collection inden udskrift, dvs. den fjerner områder i hukommelsen, der af en eller anden grund har har været i brug, men som ikke mere er nødvendige for programmet.
43533 er det antal bytes, der er fri ved opstart.

GOSUB

```
GOSUB linienummer
```

```
GOSUB 300
```

Kommando: Hopper til den specificerede linie og RETURNerer til næste instruktion (-slinie) efter den, hvorfra hoppet skete.

Se også RETURN

GOTO

```
GOTO linienummer

GOTO 300
```

Kommando: Hop til specificeret linie og forsæt programmet derfra.

HEX$

```
HEX$(positivt %tal)

PRINT HEX$(64000)
FA00
```

Funktion: Konverterer et positivt %tal til hexadecimal form (se afsn. 7.1). %tal skal ligge i området 0 til 65535.

Se også: BIN$, STR$

HIMEM

```
HIMEM

PRINT HIMEM
 43903
```

Variabel: Returnerer adressen på den højeste byte i hukommelsen, der anvendes af BASIC, og kan anvendes som nummerisk udtryk. Adressen 43903 er højeste ved opstart - kan ændres med MEMORY.

Se også FRE, MEMORY

IF

```
IF

IF logisk udtryk THEN udfør (ELSE udfør)

IF a>b THEN a=c ELSE a=d

IF logisk udtryk GOTO linienummer (ELSE udfør)

IF a>b GOTO 300 ELSE 600
```

Kommando: Anvendes til at teste et givet logisk udtryk og hvis udtrykket er sandt, udføres beskeden efter THEN eller der hoppes (GOTO) til linienummer. Er udtrykket falsk, udføres beskeden efter ELSE eller der hopppes til næste linie.
Udtrykkene kan flettes, det vil sige at der efter THEN og ELSE kan fortsættes med en ny IF-sætning.
Fortsæt kun med flere udtryk på samme linie, hvis der anvendes et "rent" IF-THEN udtryk, der kun udføres hvis det logiske ud- tryk er falsk.

Se også: AND, ELSE, GOTO, OR, THEN, WHILE

INK

```
INK nuværende ink,farve (,farve)

INK 0,1
```

Kommando: Afhængig af det skærmformat der anvendes (se afsn. 3.1) er der et antal INK til rådighed, og de kan ændres med INK-kommandoen (se afsn. 3.3).

Se også PEN, PAPER

INKEY

```
INKEY (%tal)

INKEY (%tal)=returværdi

10 CLS:IF INKEY(55)=32 THEN 30 ELSE 20
20 GOTO 10
30 PRINT "Du trykker på  SHIFT og V"
40 FOR n=1 TO 1000:NEXT:GOTO 10
```

Funktion: Bringer tangentbordet til at give en tilbagemelding om hvilken tangent, der trykkes på (tangentbordet bliver scannet (undersøgt) for nedtrykkede tangenter for hver 0.02 sekund). Kan fungere som substitut for JOY-stik til spil, idet man kan bruge tangenterne i stedet for JOY-stikket.
%tal er den nedtrykkede tangents identifikationsnummer (se fig. D forrest i bogen).
Funktionen kan bruges til at undersøge for svar af typen J/N, idet SHIFT tangenten ikke er nødvendig i en sådan situation.
I eksemplet herover vises det, hvorledes man forholder sig, hvis SHIFT tasten bruges (gælder også CTRL). Returværdien kan aflæses af skemaet herunder:

Returværdi	SHIFT	CTRL	Tangent
-1	?	?	Oppe
0	Oppe	Oppe	Nede
32	Nede	Oppe	Nede
128	Oppe	Nede	Nede
160	Nede	Nede	Nede

Se også: INPUT, INKEY$, JOY

INKEY$

```
INKEY$

10 CLS
20 PRINT "Har du flere spørgsmål (J/N)?"
30 ML$=INKEY$:IF ML$="" GOTO 30
40 IF ML$="j" OR ML$="J" GOTO 10 ELSE IF
 ML$="n" OR ML$="N" GOTO 50 ELSE GOTO 60
50 PRINT "Slut":END
60 PRINT "Der stod J eller N":FOR n=1 TO
 1000:NEXT:GOTO 10
```

Funktion: "Læser" en tangent fra tangentbordet for at foretage en aktivering af et program uden at trykke på ENTER for hvert svar (gælder kun for karakterer, der ligger direkte på tastatu- ret). Hvis der trykkes på en tast udføres den tilsvarende ordre, ellers fortsættes med returnering af en tom streng (linie 30) indtil der trykkes på den rigtige taste.

Se også: INPUT, INKEY

INP

```
INP(port nummer)

PRINT INP(&FF77)
 255
```

Funktion: Returnerer indlæsningsværdien fra den I/O port der er specificeret i (port nummer).

Se også: OUT, WAIT

INPUT

```
INPUT (# stream),"strengudtryk",variabel,...,variabel

INPUT #0,"Dit telefonnummer?",nr

INPUT (# stream),strengudtryk,variabel,...,variabel

10 CLS
20 INPUT "Skriv din fødselsdag som
          5,august,1954 - adskilt af komm
aer, som  vist ",a,b$,c
30 PRINT:PRINT "Du er født den:"
40 PRINT a;b$;c
```

Kommando: Indlæser data, evt. fra en specificeret STREAM.
Et semicolon efter INPUT undertrykker den CR (Carriage Return - "vognretur") der følger efter den linie man indlæser.
Et semicolon efter et strengudtryk får skærmen til at udskrive et spørgsmålstegn - et komma undertrykker ?. Hvis der indlæses fra en kassette-stream, bliver ? undertrykket, uanset om det

er specificeret eller ej.
Hvis du laver en indtastning der ikke svarer til variablen (et bogstav til en nummerisk variabel), vil BASIC'en svare med

```
?Redo from start
```

(sammen med den tekst du har specificeret i INPUT-sætningen (linien bliver behandlet forfra)).
Alle INPUT-sætninger skal efterfølges af ENTER.

Er det en kassette-STREAM, vil variablerne blive indlæst fra filen enkeltvis for hver enkelt opgivet variabel, og skal svare overens med variablerne i INPUT-sætningen, der er:
nummerisk variabel, der skelnes enten ved komma, CR, mellemrum eller EOF (End Of File).
Hele strengudtryk i gåseøjne kan kan også gemmes på denne måde. De identificeres ved Gåseøjnenes start og indlæses fortløbende indtil slut-gåseøjnene nås.

Se også: INKEY$, LINE INPUT, READ

INSTR

```
INSTR (%tal, strengudtryk,strengudtryk)

10 b$="Amstrad"
20 PRINT INSTR(2,b$,"strad")
```

Funktion: Returnerer oplysningen om, hvilken position det andet strengudtryk begynder ved ved i det første strengudtryk. Hvis %tal opgives, begynder søgningen ved denne position, eller fra strengens start.

Se også: LEFT$, MID$, RIGHT$

INT

```
INT(nummerisk udtryk)

10 PRINT INT(-1.934)
20 PRINT INT(4.65)
-2
 4
```

Funktion: Returnere heltalsværdien afrundet til nærmeste mindre %tal. Det samme som FIX for positive numre, men returnerer en mindre end FIX ved negative tal, der ikke allerede er %tal.

Se også: CINT, FIX, ROUND

JOY

```
JOY(%tal)
```

```
10 IF JOY(0)=8 GOTO 100
```

Funktion: Læser et bit-signifikant resultat (dvs. der testes på hver enkelt bit i en byte) fra et tilsluttet joystik, specificeret i %tal (enten 0 eller 1). Decimalværdien til højre for lighedstegnet ses i skemaet herunder:

Bit	Funktion	Decimalværdi
0	Op	1
1	Ned	2
2	Venstre	4
3	Højre	8
4	Skyd 2	16
5	Skyd 1	32

Se også: INKEY

KEY

```
KEY %tal,CHR$(n) + strengudtryk + CHR$(n)

KEY 140,"RUN" + CHR$(13)
```

Kommando: Tillægger de definerbare taster den funktion, der ønskes. Der er 32 definerbare taster i karaktersætområdet 128-159 (se eks. afsnit 3.2 og siderne 21-24). Når en af disse karakterer "læses" ekspanderes den til en streng på op til 32 karakterer - det samlede antal af ekspansionskarakterer kan dog ikke overskride 120.
Se eksemplet i afsn. 3.2.

Se også: KEY DEF

KEY DEF

```
KEY DEF tangent nummer,gentag (,normal,SHIFTed,ConTRoL)

KEY DEF 46,1,63
```

Kommando: (KEY DEFine) ændrer den værdi, der generes af hvilken som helst tangent (defineret siderne 21-24, fig. 8 tangentplacering). Ovenstående eksempel ændrer N-tangenten til at skrive ? (karakter 63). For normal funktion igen defineres

```
KEY DEF 46,1,110
```

hvor karakter 110 er lille n .

Se også KEY

LEFT$

```
LEFT$(strengudtryk,%tal)

10 CLS
20 a$="Amstrad"
30 b$=LEFT$(a$,3)
40 PRINT b$
```

Funktion: Returnerer det antal karakterer i den specificerede streng, der er angivet i %tal, regnet fra strengens venstre side. Er strengen mindre end det givne %tal, returneres hele strengen.

Se også: MID$, RIGHT$

LEN

```
LEN(strengudtryk)

a$="Amstrad CPC 464":PRINT LEN(a$)
 15
```

Funktion: Returnerer et %tal, som er længden på strengen (alle typer karakterer og mellemrum).

LET

```
LET n=25
```

Kommando: Forstået som: lad n være lig med 25. En reminisens fra BASIC'ens barndom, hvor enhver variabel skulle defineres på denne måde. Er ikke nødvendig i Amstrad's BASIC-version.

LINE INPUT

```
LINE INPUT (# stream),strengudtryk

LINE INPUT #0,A$

LINE INPUT (# stream),"strengudtryk";strengudtryk

LINE INPUT "Dit navn? ";N$
```

Kommando: Læser en hel linie fra den specificerede stream (§0, hvis ikke specificeret). Et semicolon efter LINE INPUT undertrykker ekko af CR (Carriage Return).

Se også: INKEY$, INPUT, READ

LIST

```
LIST(linieområde)(,# stream)

LIST
LIST 100
LIST 100-300
LIST -300
LIST 100-300,#8
```

Kommando: Lister specificerede linier (hvis intet angivet, hele programmet). #8 er printerstream'en.
Listningen kan standses med ESC. Et tryk holder pause i listningen (den fortsætter ved tryk på mellemrumstangenten) - to tryk BREAKer og returnerer computeren til direkte indlæsning og skærmen udskriver *BREAK*.

LOAD

```
LOAD filnavn (,startadresse)

LOAD "DANKAR"
```

Kommando: Indlæser et program fra kassette til hukommelsen.
Startadressen specificeres kun ved binære filer.
Se iøvrigt afsnit 1.2 og 1.3.

LOCATE

```
LOCATE (#stream,)x-koordinat,y-koordinat

10 MODE 1
20 LOCATE 12,13
30 PRINT "Amstrad CPC 464"
```

Kommando: Placerer cursoren på den specificerede position i forhold til øverste venstre hjørne (1,1). Er der opgivet en stream, er øverste venstre hjørne i hvert vindue (WINDOW) 1,1.

Se også WINDOW

LOG

```
LOG(nummerisk udtryk)

PRINT LOG(12589.52)
 9.44062
```

Funktion: Beregner den naturlige logaritme af nummerisk udtryk og returnerer et reelt tal selv om udtrykket er et %tal.

Se også: EXP, LOG10

LOG10

```
LOG10(nummerisk udtryk)

PRINT LOG10(9.44062)
 0.975000517
```

Funktion: Beregner 10-tals logaritmen til nummerisk udtryk og returnerer et reelt tal selv om udtrykket er et %tal.

Se også: EXP, LOG

LOWER$

```
LOWER$(strengudtryk)

a$="Amstrad CPC 464":PRINT LOWER$(a$)
amstrad cpc 464
```

Funktion: konverterer strengudtrykket til samme strengudtryk, men med små bogstaver. Anvendes, når der skal bruges små bogstaver i databehandlingen, men bogstaverne kommer som en blanding af store og små bogstaver.

Se også: UPPER$

MAX

```
MAX(nummerisk udtryk,...,nummerisk udtryk)

10 n=26
20 PRINT MAX(7,2,n,3,18,23)
```

Henter den største værdi fra den spcificerede liste af nummeriske variabler (udtryk).

Se også: MIN

MEMORY

```
MEMORY adresse i hukommelsen

MEMORY &88B8
```

Kommando: Omstiller (reset) BASIC'ens hukommelses (memory) parametre for at ændre hukommelsens størrelse. Maskinkodeprogrammer, der placeres over HIMEM slettes ikke af NEW og lignende ordrer.

Se også: FRE, HIMEM

MERGE

```
MERGE (filnavn)

MERGE "MOLINT"
```

Kommando: Fletter (MERGEr) et program fra kassette ind i et program, der allerede ligger i hukommelsen. Hvis der ikke er opgivet noget filnavn, bliver den første fil på båndet indlæst.
Hvis den første karakter i filnavnet er ! , bliver den vanlige skærmmeddelelse undertrykt.
Skal det program, der ligger i hukommelsen ikke overskrives af det der MERGEs, skal førstnævnte have lavere linienumre end det nye.
Beskyttede filer kan ikke MERGEs.

Se også CHAIN, CHAIN MERGE, LOAD

MID$

```
MID$(strengudtryk,%tal(,%tal))

a$="Amstrad":PRINT MID$(a$,2,3)
MST

a$="Amstrad":b$=MID$(a$,2,2):PRINT b$
MS
```

Kommando og funktion: MID$ specificerer en del af en streng (en under- eller sub-streng), der kan bruges enten som bestemmelse for en kommando eller som argument i en funktion.
Det første %tal angiver nummeret på den karakter i strengen, hvorfra udlæsningen skal starte og det andet %tal angiver hvor mange karakterer, der skal udlæses (hvis det udelades, udlæses resten af strengen).
Se anvendelsen i programmet i afsn. 4.1.

Se også LEFT$, RIGHT$

MIN

```
MIN(nummerisk udtryk,...,nummerisk udtryk)

10 n=26
20 PRINT MIN(7,2,n,3,18,23)
```

Kommando: Udskriver den mindste værdi af en række specificerede nummeriske udtryk.

Se også: MAX

MODE

MODE %tal

MODE 1

Kommando: Ændrer skærmformatet (MODE) og CLEARer skærmen til INK 0, som ikke nødvendigvis er den øjeblikkelige PAPER INK. Alle grafiske og tekst vinduer stilles om til hele skærmen, og tekst- og grafikcursor sættes i deres respektive startpositioner. Se afsnit 3.1.

Se også: ORIGIN, WINDOW

MOVE

MOVE x-koordinat,y-koordinat

MOVE 66,89

Kommando: Flytter den grafiske cursor til positionen der er specificeret ved de absolutte koordinater.

Se også: DRAW, DRAWR, MOVER, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS

MOVER

MOVER 66,89

Kommando: Flytter den grafiske cursor til en relativ position i forhold til de nuværende koordinater.

Se også: DRAW, DRAWR, MOVE, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS

NEW

NEW

NEW

Kommando: Sletter det øjeblikkelige program og variabler. Tangent (KEY) definitioner slettes, men skærmen slettes ikke. Næsten det nærmeste man kan komme en nulstilling af maskinen uden at slukke den. Brug af CALL 0, nulstiller den totalt, da computerens opstarts rutine ligger på denne adresse i ROM.

NEXT

NEXT variabel,...,variabel

FOR n=1 TO 1000:NEXT n

Kommando: Indikerer afslutningen på en FOR-løkke. Det er ikke nødvendigt at specificere variabelnavne.

Se også: FOR

ON BREAK GOSUB

```
ON BREAK GOSUB linienummer

10 ON BREAK GOSUB 40
20 PRINT "Programmet kører i øjeblikket"
30 GOTO 20
40 CLS
50 PRINT "2 tryk på ESC kalder subrutinen"
60 FOR n=1 TO 1000:NEXT
70 RETURN
```

Kommando: Kalder en subrutine, når ESC tasten nedtrykkes 2 gange.

Se også: ON BREAK STOP, RETURN

ON BREAK STOP

```
ON BREAK STOP

10 ON BREAK GOSUB 40
20 PRINT "Programmet kører i øjeblikket"
30 GOTO 20
40 CLS
50 PRINT "2 tryk på ESC kalder subrutinen"
60 FOR n=1 TO 1000:NEXT
70 ON BREAK STOP
80 RETURN
```

Kommando: Anvendes i en ON BREAK subrutine, undertrykker funktionen BREAK ordren, men har ellers ingen indflydelse. I programmet herover vil ON BREAK GOSUB kommandoen kun virke en gang.

Se også: ON BREAK GOSUB

ON ERROR GOTO

```
ON ERROR GOTO linienummer

10 ON ERROR GOTO 100
20 PRINT "VI bruger ERL og LIST for
at finde en    evt. fejl."
30 FOR n=1 TO 1000:NEXT
40 FOR n=1 TO 4:READ a:NEXT
50 DATA 45,67,28
60 GOTO 20
100 CLS: PRINT "Der er en fejl i linie "
;ERL:PRINT
110 LIST
```

Kommando: Hopper til en specificeret linie, når programmet opdager en fejl. Her er der for få variabler i datasætningen.

Se også: ERL, ERR, RESUME

ON GOSUB
ON GOTO

ON %tal GOSUB linienummer,...,linienummer

ON e GOSUB 350,380,410,440,470

ON %tal GOTO linienummer,...,linienummer

ON a GOTO 100,200,300,400

Kommando: Udfører en subrutine eller hopper til linienummer efter værdien af %tal. Er e f.eks. 4, udføres subrutinen linie 440- se afsn. 4.1

Se også: GOSUB, GOTO

ON SQ GOSUB

ON SQ (kanal) GOSUB linienummer

ON SQ (2) GOSUB 900

Kommando: Tillader et interrupt (afbrydelse) når der er en åbning i den givne Sound Queue (se SQ). Kanal er et %tal med en af flg. værdier:

1 - kanal A
2 - kanal B
3 - kanal C

Se også: SOUND, SQ

OPENIN

OPENIN filnavn

OPENIN "!MOLINT"

Kommando: Åbner en indlæsningsfil fra båndoptageren med informationer, der er brug for i det øjeblikkelige program.
Hvis den første karakter i filnavn er ! , undertrykkes den vanlige skærmtekst.

Se også CLOSEIN, OPENOUT

OPENOUT

OPENOUT filnavn

OPENOUT "!MOLINT"

Kommando: Abner en udlæsningsfil til kassetten med det øjeblikkelige program i hukommelsen. Hvis den første karakter i filnavn er ! , undertrykkes de vanlige skærmmeddelelser og programmet danner den første blok af data og opretter en fil med det ønskede navn. Hver blok består af 2K byte data og udskrives ikke før 2K bufferen er fyldt, eller filen lukkes med CLOSEOUT el. lign. Bemærk at NEW ikke tager hensyn til en fil i bufferen, og at data derved går tabt.

Se også: CLOSEOUT, OPENIN, PRINT

ORIGIN

ORIGIN x-koord.,y-koord.(,venstre,højre,top,bund)

```
10 CLS:BORDER 3
20 ORIGIN 0,0,50,590,530,50
30 DRAW 540,350
40 GOTO 20
```

Kommando: Bestemmer starten for den grafiske cursor (x,y). De 4 ekstra udtryk indeholder instruktionerne til et nyt grafisk vindue, der kan anvendes i alle skærmformater.
ORIGIN er punktet i vinduets nederste venstre hjørne.
Hvis nogle af vindueshjørnerne specificeres til en position uden for skærmen bringen BASIC'en ind til til den nærmeste synlige kant.

Se også WINDOW

OUT

OUT port nummer,%talsudtryk

OUT &F8F4,10

Kommando: Sender værdien af %talsudtrykket (i området 0 til 255) til den port, der er specificeret i port nummer (portens adresse).

Se også: INP, WAIT

PAPER

PAPER (#stream,)maskeret INK

Kommando: Sætter baggrundsfaven (papiret) for karaktererne. Når en karakter udskrives på tekstskærmen, fyldes karaktercellerne

først med PAPER INK før karakteren skrives, hvis der ikke er valgt at skrive transparant.
Hvis den maskerede INK ikke er fri i det valgte skærmformat, vælger BASIC'en en mulig farve.

Se også: INK, PEN, WINDOW

PEEK

```
PEEK(adresseudtryk)

PRINT PEEK(&C0FF)
 128
```

Funktion: Udskriver værdien af en hukommelsesadresse i computerens RAM - funktionen kan ikke udlæse fra ROM.
Er det en 2 bytes variabel udlæses den således:

```
PRINT PEEK(a)+256*PEEK(a+1)
```

hvor a er adressen.

Se også POKE

PEN

```
PEN (#stream,)maskeret INK

PEN 1,2
```

Kommando: PEN sætter den farve, der skal bruges, når der tegnes ved den givne stream.

Se også: INK, PAPER

PI

```
PI

PRINT PI
 3.14159265

 10 REM perspektivtegning
 20 MODE 2
 30 RAD
 40 INK 1,0
 50 INK 0,12
 60 BORDER 9
 70 FOR n=1 TO 200
 80 ORIGIN 420,0
 90 DRAW 0,200
100 REM træk vinkler fra udgangspunkt
110 DRAW 30*n*SIN(n*PI/4),(SIN(PI/2))*n*SIN(n)
120 NEXT
```

Funktion: Værdien af forholdet mellem cirklens omkreds og diameter. Nærmeste maskinrepræsentation 3.141592653468251.

Se også: DEG, RAD

PLOT

```
PLOT  x-koordinat,y-koordinat(,maskeret INK)

10 MODE 2
20 DEG
30 ORIGIN 320,200
40 FOR vinkel=1 TO 360
50 xpunkt=175*COS(vinkel)
60 ypunkt=175*SIN(vinkel)
70 PLOT xpunkt,ypunkt
80 NEXT
```

Kommando: Plotter et punkt på de givne koordinater. Se afsnit 3.5.

Se også: DRAW, DRAWR, MOVE, MOVER, ORIGIN, PLOTR, TEST, TESTR, XPOS, YPOS

PLOTR

```
PLOTR x-koordinat,y-koordinat(,maskeret INK)

90 ORIGIN XPOS,YPOS
100 FOR vinkel=1 TO 180
110 xpunkt=175*COS(vinkel)
120 ypunkt=175*SIN(vinkel)
130 PLOTR xpunkt,ypunkt
140 MOVE 0,0
150 NEXT
```

Kommando: Plotter et punkt i en relativ afstand fra den grafiske cursor nuværende placering. Læg programmet ind i forlængelse af det under PLOT.

Se også: DRAW, DRAWR, MOVE, MOVER, ORIGIN, PLOT, TEST, TESTR, XPOS, YPOS

POKE

```
POKE adresse,%tal

POKE &00FF,10
```

Kommando: Giver direkte adgang til computerens hukommelse og lægger %tal (mellem 0 og 255) direkte i den specificerede adresse.
Er det et %tal større end 255, skal det deles over to adresser. En 2 byte varibel POKEs til adressen a med værdien v således

```
POKE a,v-256*INT(v/256):POKE a+1, INT(v/256)
```

POKE 46312,255 giver automatisk store bogstaver (caps lock), når du taster ind. POKE 46312,0 stiller bogstaverne normalt igen.

Det er let at "vælte" systemet, hvis du POKEr forkerte steder, så udlæs evt. programmer til båndoptageren før du begynder.

Se også: PEEK

POS

```
POS(#stream)

PRINT POS(#8)
```

Funktion: Giver den øjeblikkelige status for en given stream.
Skærm: Giver den nuværende x-koordinat for textcursoren i det specificerede tekstvindue.
Printer: Angiver printpositionen på nuværende linie, hvor 1 er venstre margin. Alle ASCII karakterer større end 31 er includeret.
Kassetteudlæsningsstream: Som for printer.

Se også: VPOS

PRINT

```
PRINT (#stream,)(printliste)(USING klausul)(seperator)

PRINT #0, "AMSTRAD"
```

Udlæsning af data til en kassettefil:

```
10 OPENOUT "DATA"
20 PRINT #9,"MOLINT","AMSTRAD","andre data"
30 CLOSEOUT
```

Udskriv data på printer:

```
10 areal=PI*4.75*4.75
20 PRINT #8,USING "###.##;areal
30 PRINT #0
run
 70.8821842
```

og på printeren udskrives

```
 70.88
```

Se en kombination af disse i flg. sorteringsprogram:

```
10 INPUT "Hvor mange emner skal sorteres
";a
20 DIM a$(a?1)
30 For t=1 TO a
40 Input "Indlæs et emne";a$(t)
50 NEXT
60 FOR z=1 TO a
70 FOR t=1 TO a
80 b$=a$(t)
90 IF a$(t+1)>=a$(t) GOTO 100 ELSE GOTO
120
100 a$(t)=a$(t+1)
110 a$(t+1)=b$
120 NEXT t:NEXT z
130 INPUT "Udskrift til skærm (0) eller prin
ter (8)?";valg
140 FOR t=a TO 1 STEP -1
150 PRINT  valg,a$(t)
160 NEXT t
170 OPENOUT "SORTERING"
180 FOR t=a TO 1 STEP -1
190 PRINT  9,a$(t);
200 NEXT
```

Følgende formatteringer kan anvendes i forbindelse med PRINT sætninger (se også SPC, TAB):

Nummerisk:

Format symbol	Mulige cifre	Felt karakterer	Definition	Eksempel
#	1	1	Nummerisk felt	###
.	0	1	Decimalpunkt	#.#
+	0	1	Print tegn foran eller bagved.	+##
			Positive tal vil have +.	
			Negative tal kan ikke have - foran.	###+
-	0	1	Printes bagved hvis negativt ellers ikke.	##.##-
**	2	2	Udfyldning med stjerner.	**###.##
$$	1	2	Flydende $-fortegn placeres foran første ciffer.	$$##.##
**$	2	3	Udfyldning med stjerne og flydende $-tegn.	**$#.##
,	1	1	Sæt komma for hver 3. ciffer til venstre for decimalpunkt.	##,###.#
↑↑↑↑	0	4	Exponential notation opstillet så ledende ciffer ikke er nul (0).	#.##↑↑↑↑

Strenge:

!	Kun første karakter.
<mellemrum>	Antal specificerede mellemrum plus et mellemrum før og efter karakterfelt.
&	Varaiabel feltlængde.

RAD

RAD

RAD

Kommando: Omstiller computeren til RADianer, som den også befinder sig i ved opstart.

Se også: ATN, COS, DEG, SIN, TAN

RANDOMIZE

RANDOMIZE (nummerisk udtryk)

```
10 RANDOMIZE 39
20 PRINT RND(6)
```

Kommando: BASIC'en genererer pseudo tilfældige (random) tal, der er sekvensafhængige af de foregående tal - når der startes fra et givent tal, vil rækkefølgen altid være den samme. RANDOMIZE sætter en ny grundværdi for generatoren, enten til en given værdi eller til en værdi, der indlæses af operatøren.
RANDOMIZE TIME vil producere en sekvens, der vil være næsten umulig at gentage.

Se også: RND, TIME

READ

READ variabel,...,variabel

READ KC,ML,B$,C$,pe

Kommando: READ læser data fra en liste af konstanter der er specificeret i den tilhørende DATA-sætning og tildeler dem til variabler, og går automatisk fremad i DATA-sætningen. Se programmet i 4.1.

Se også: DATA, RESTORE

RELEASE

RELEASE lydkanaler

RELEASE 4

Kommando: Når en lyd er placeret i en kanal, kan den indeholde en vent (eng: hold) ordre, hvad der kan bringes til ophør med denne kommando. Udtrykket til at identificere kanalen er bit-afhængig:

Kanal A=bit 0 - kanal B=bit 2 - kanal C=bit 2

4 (0100, binær) løsner (RELEASE) kanal C.

REM

```
REM

REM resten af linien ignores:PRINT "alt inclusive"
```

Kommando: Bruges til at placere REMarks (bemærkning) eller REMinders (påmindelser) i et program uden at disse påvirker programudførelsen. Der kan stå hvilken som helst karakter efter en REM. En apostrof i en linie (ikke som en del af en streng) er det samme som :REM.

REMAIN

```
REMAIN (%tal)

REMAIN (3)
PRINT #6,REMAIN(0);
```

Funktion: Afbryder en specficeret opholds (delay) timer (%tal i området 0 til 3).
Læser manglende tællertrin fra den delaytimeren. 0 returneres, hvis timeren ikke var i venteposition.

Se også: AFTER, EVERY

RENUM

```
RENUM nyt linienummer,gammelt linienummer(,stepværdi)

RENUM
RENUM 100,100
```

Kommando renummerer programlinier i det specificerede område med den stepværdi, der er angivet. Er der ikke angivet noget startes ved 10 med step på 10, altså som RENUM 10,,10.
RENUM tager sig også af GOSUB, GOTO og andre liniekald.
Linienumre skal ligge i området 1...65535.

RESTORE

```
RESTORE (linienummer)

RESTORE 1400
```

Repositionerer READ-"pegepinden" ved begyndelsen af den specificerede linie eller hvis den ikke er opgivet, ved den første datasætning i programmet. Se programmet i 4.1.

Se også DATA, READ

RESUME

```
RESUME (linienummer)
RESUME NEXT

RESUME 300
```

Kommando: Når en fejl er blevet opdaget af en ON ERROR GOTO og er blevet udført, tillader RESUME at normal programudførelse fortsætter (linien, der fortsættes i kan specificeres).

Se også: ON ERROR GOTO

RETURN

```
RETURN

RETURN
```

Kommando: Viser afslutningen på en subrutine og får BASIC'en til at vende tilbage til linien/ordren efter den GOSUB der hører til rutinen.

Se også: AFTER n GOSUB, EVERY n GOSUB, GOSUB, ON BREAK GOSUB, ON n GOSUB, ON SQ GOSUB

RIGHT$

```
RIGHT$(strengudtryk,%tal)

a$="Amstrad":PRINT RIGHT$(a$,3)
rad
```

Funktion: Tæller %tal karakterer ind i den specificerede streng fra højre side. Er strengen kortere end %tal, returneres hele strengen.

Se også LEFT$, MID$

RND

```
RND(nummerisk udtryk)

RANDOMIZE 39:PRINT RND(6)
 0.334440658
```

Funktion: Danner et tilfældigt (random) tal, der kan være det næste i en serie, en gentagelse af det sidste eller det første i en ny sekvens. RANDOMIZE kommandoen i eksemplet garanterer, at RND(6) returnerer det samme nummer hver gang.
RND(0) returnerer en kopi af det forrige tal. Undgå negative værdier, da de har tendens til ikke at være rigtig tilfældige.
Tallet der genereres ligger i området 0n1. Skal der dannes heltal, kan det gøres som

```
n=INT(RND(1)*100)+1        Et tal i området   1 - 100
n=INT(RND(1)*150)+200      Et tal i området 200 - 349
```

Se også: RANDOMIZE

ROUND

```
ROUND(nummerisk udtryk,%tal)

10 a=PI
20 FOR afrund=9 TO 0 STEP -1
30 PRINT afrund,ROUND(a,afrund)
40 NEXT
```

Funktion: Afrunder nummerisk udtryk til en antal decimalpladser, specificeret i %tal (afrunding til nærmeste multiplum af 10). Hvis %tal er negativt afrundes tallet til et absolut tal fulgt af et antal nuller, bestemt af %tal før decimalpunktet.

Se også: ABS, CINT, FIX, INT

RUN"

```
RUN strengudtryk

RUN "MOLINT"
```

Kommando: Indlæser et program fra kasette og starter programkørslen. Hvis strengudtrykket ikke er defineret, indlæses det første program på båndet, der har et strengudtryk tilknyttet.
Se afsnit. 1.2.

Se også: LOAD, RUN

RUN

```
RUN (linienummer)
```

Kommando: Starter udførelsen af program i hukommelsen (evt. fra en specificeret linie). Hele programmet nulstilles (brugerfunktioner og variabler slettes fra hukommelsen og DEFINT, DEFREAL og DEFSTR genetableres - kassettefiler lukkes og evt. data i buffere forsvinder.

Se også: LOAD, RUN"

SAVE

```
SAVE filnavn(,filtype)(,binære parametre)

SAVE "MOLINT",P
```

Kommando: Udlæs programmet i hukommelsen til kassettebåndoptageren under navnet filnavn.
,A udlæser et program i ASCII-kode
,B udlæser et område som binær fil (f.eks. skærmen)
,P beskytter en fil mod listning
Se afsn. 1.3.

Se også: CHAIN, CHAIN MERGE, LOAD, MERGE, RUN"

SGN

```
SGN(nummerisk udtryk)

Print SGN(0),SGN(5433),SGN(-149)
 0            1           -1
```

Funktion: SiGNum udskriver efter fortegnet på nummerisk udtryk. Dvs. 0, hvis 0 - 1 hvis tallet er positivt og -1 hvis det er negativt.

Se også ABS

SIN

```
SIN(nummerisk udtryk)

PRINT SIN(PI/2)
 1
```

Funktion: Kalkulerer den reelle værdi for sinus af nummerisk udtryk i RADianer med mindre der er valgt grader (DEG).

Se også: ATN, COS, DEG, RAD, TAN

SOUND

SOUND kanal status,toneperiode(,svingning(,lydstyrke(,ENV(,ENT(,støjperiode)))))

Kommando: CPC 464's lydkommandoer er nærmere beskrevet i afsn. 3.4.

Se også: ENV, ENT

SPACE$

```
SPACE$(%tal)

SPACE$(120)
```

Funktion: Udskriver det specificerede antal mellemrum.

Se også: PRINT, SPC, TAB

SPC

```
SPC(%tal)

SPC(14)
```

Funktion: Udskriver det givne antal mellemrum. Hvis %tal er negativt sættes %tal=0. Hvis %tal er "bredere" end skærm, printer etc. bliver udtrykket reduceret til at holde sig inden for området. Det er ikke nødvendigt at adskille med komma eller semicolon - et semicolon er underforstået.

Se også: PRINT, SPACE$, TAB

SPEED INK

```
SPEED INK %tal,%tal

10 INK 0,9,12:INK 1,0,26
20 BORDER 12,9
30 SPEED INK 50,20
```

Kommando: SPEED INK får skærmkanten (BORDER) til at veksle mellem to farver. Det første %tal angiver tiden for den første farve, det andet %tal for den anden. Tiden mellem farveskiftene måles i 0,02 sekunders enheder (ved lysnetfrekvens på 50 Hz).
Pas på øjnene ved for kraftige lyseffekter (hvis svingningerne kommer til at ligger omkring 50 i sekundet giver det en stroboskob- effekt).

Se også: BORDER, INK

SPEED KEY

```
SPEED KEY startperiode,repeterperiode

SPEED KEY 20,3
```

Kommando: Hvis en tast holdes nede konstant, repeterer den automatisk efter en vis startperiode. Det kan ændres i 0,02 sekunders enheder med værdier i området 1 til 255. Ved opstart er hastigheden sat til 10,10
Meget små startperioder kan indvirke på tangentbordets evne til at omsætte modtagne informationer.

Se også: KEY DEF

SPEED WRITE

```
SPEED WRITE %tal

SPEED WRITE 1
```

Kommando: Der kan udlæses til kassetten med enten 2000 baud

(%tal=1) eller hastigheden ved opstart på 1000 (%tal=0). Når der indlæses fra kassetten, sættes baud-raten automatisk under software kontrol.

Se også: SAVE

SQ

SQ(kanal)

```
10 MODE 1
20 FOR n=20 TO 0 STEP -1
30 PRINT n
40 SOUND 1,10+n,100,7
50 WHILE SQ(1)>127:WEND
60 NEXT
```

Funktion: SQ (Sound Queue - en form for "lydkø") kontrollerer antallet af frie indgange til køen for en given kanal, hvor

Kanal A = 1 - kanal B = 2 - kanal C = 3

Funktionen undersøger hvor vidt kanalen er aktiv (og hvis ikke) hvorfor den forreste funktion venter (hvis der overhovedet er nogen). Resultatet er binært orienteret. Bit nr. kontrollerer:

0,1,2	- antallet af frie pladser i køen
3,4,5	- om kanalerne i forreste funktion overlapper hinanden
6	- sættes hvis den forreste funktion bliver "holdt" på pladsen
7	- sættes, hvis kanalen er aktiv i øjeblikket

Se også: ON SQ GOSUB, SOUND

SQR

SQR(nummerisk udtryk)

```
PRINT SQR(25)
 5
```

Funktion: Returnerer kvadratroden (SQuare Root) af det nummeriske udtryk (der skal være positivt).

STEP

FOR variabel = værdi TO variabel STEP værdi

FOR MANED = 1 TO 12 STEP 2

Kommando: Trintæller i en FOR/NEXT løkke. Værdien kan være alle reelle tal. Hvis ikke angivet, sættes den automatisk til +1.

Se også: FOR, NEXT

STOP

```
STOP

300 IF a<n THEN STOP
```

Kommando: Standser programudførelsen, men efterlader programmet på en sådan måde at det kan genstartes med CONT uden noget går tabt, og der ikke har været redigeret i programmet.

Se også: CONT, END

STR$

```
STR$(nummerisk udtryk)

PRINT STR$(B16)
 2838

PRINT STR$(X11010011)
 211
```

Funktion: Konverterer nummerisk udtryk til en decimalstreng på samme form som der anvendes i en PRINT kommando.

Se også: BIN$, HEX$, PRINT, VAL

STRING$

```
STRING$(%tal,karakterspecifikation)

PRINT STRING$(20,"£")

£££££££££££££££££££
```

Funktion: Udskriver et strengudtryk med den specificerede karakter %tal gang.

Se også: SPACE$

SWAP

Se under WINDOW SWAP.

SYMBOL

```
SYMBOL karakter nummer,1. række,...,8. række

10 SYMBOL AFTER 90
20 SYMBOL 91,0,0,236,18,124,144,110,0
30 PRINT CHR$(91)
```

Kommando: SYMBOL redefinerer det tegn i ASCII karaktersættet som

er defineret ved karakter nummer og rækker af bit informationer. Se afsnit 2.5.

Se også SYMBOL AFTER

SYMBOL AFTER

SYMBOL AFTER %tal

SYMBOL AFTER 90

Kommando: Sætter antallet af mulige brugerdefinerede karakterer. Ved opstart er værdien 240 (giver 16 definerbare karakterer). Hvis %tal sættes til 32 kan alle karakterer fra 32 til 255 redefineres.

Se også: SYMBOL

TAB

TAB(%tal)

TAB(35)

Funktion: Indsætter mellemrum fra sidste printposition og til den specificerede nye printposition. Hvis %tal negativ sættes værdien=1. Hvis den går ud over skærmens "bredde" sendes den til næste linie.

Se også: PRINT, SPACE$, SPC

TAG

TAG (#stream udtryk)

```
10 MODE 2
20 BORDER 9:INK 0,12:INK 1,0
30 FOR n=1 TO 100:MOVE 200+n,320+n
40 TAG
50 IF n<70 GOTO 60 ELSE 70
60 PRINT "AMSTRAD";:GOTO 80
70 PRINT "COMPUTER";
80 NEXT
90 GOTO 20
```

Kommando: Tekst sendt til en given stream kan omdirigeres til at skrive på den grafiske cursors plads, så tekst og symboler kan mixes. stream udtrykket sættes til 0, hvis det ikke specificeres. Øverste venstre hjørne i karakteren "hægtes" på grafisk cursor, og ikke printbare controlkarakterer (f.eks ENTER) vil blive udskrevet på skærmen, hvis PRINT ordrerne ikke efterfølges af semicolon.

Se også: TAGOFF

TAGOFF

```
TAGOFF (#stream)

TAGOFF #0
```

Kommando: Afbryder TAG for en given stream og sender teksten til den tidligere tekst cursor position på det tidpunkt hvor TAG blev sat til.

Se også: TAG

TAN

```
TAN(nummerisk udtryk)

PRINT TAN(45)
 1,61977519
```

Funktion: Kalkulerer TANgenten for den givne vinkel i nummerisk udtryk, der skal ligge i området -200000 til +200000, beregnet i RADianer.

Se også: ATN, COS, DEG, RAD, SIN

TEST

```
TEST(x-koordinat,y-koordinat)

TEST 300,200
```

Funktion: Rapporterer værdien af den INK der er på den specificerede position på grafisk skærm.

Se også: DRAW, DRAWR, MOVE, MOVER, PLOT, PLOTR, TESTR

TESTR

```
TESTR(x-tillægsværdig,y-tillægsværdi)

TEST 5,5
```

Funktion: Rapporterer værdien af den INK, der er på den nuværende placering af grafisk cursor efter tillæg af de specificerede værdier.

Se også: DRAW, DRAWR, MOVE, MOVER, PLOT, PLOTR, TEST

THEN

```
IF udtryk THEN udtryk

IF a=1 THEN 300
```

Kommando: Efterfølger IF kommandoen med besked om, hvad der skal foretages. Kan udelades i forbindelse med GOTO.

Se også: IF

TIME

```
TIME

10 d=INT(TIME/300)
20 t=((TIME/300)-d)
30 PRINT t
40 GOTO 20
```

Funktion: Rapporterer den forløbne tid siden der blev tændt for computeren, eksklusiv perioder, hvor der har været læst til eller fra båndoptageren. Enheden er 1/300 sekund.

TO

```
FOR variabel = værdi TO værdi

FOR a=1 TO 12
```

Kommando: Angiver den slutværdi FOR/NEXT løkken har.

Se også FOR, NEXT

TRON
TROFF

```
TRON

TROFF

TRON
```

Kommando: BASIC indeholder muligheden for at spore udførelsen af programmet med TRace ON, der melder om hver linie, der bliver eksekveret, i firkantede paranteser (nogle af de danske karakterer, hvis du har lagt dem ind). TRace OFF afbryder igen.

UNT

```
UNT(adresse udtryk)

PRINT UNT(&FF66)
```

Funktion: Konverterer positiv 16 bit (%tal) i området 0 - 65535 og returnerer en værdi i området -32768 til +32767.

Se også: CINT, FIX, INT, ROUND

UPPER$

```
UPPER$(streng udtryk)

PRINT UPPER$("Amstrad")
AMSTRAD
```

Funktion: Konverter alle små bogstaver til store.

Se også: LOWER$

USING

```
USING

PRINT USING
```

Funktion: Se anvendelse under PRINT.

VAL

```
VAL(strengudtryk)

10 a$="Amstrad CPC 464 Computer"
20 PRINT VAL(a$)
```

Funktion: Uddrager et nummerisk udtryk fra begyndelsen af et strengudtryk. Det modsatte af STR$.

Se også: STR$

VPOS

```
VPOS(#stream)

PRINT VPOS(#0)
```

Funktion: Returnerer den vertikale position af tekst cursoren for den pågældende stream (skal opgives).

Se også: POS

WAIT

```
WAIT port nummer,maske(,invertering)

WAIT &FF34,20,25
```

Kommando: Afbryder operationen til til en given I/O port returnerer en bestemt værdi i området 0 til 255. BASIC kører i en intern løkke, medens porten aflæses. Den læste værdi er EXklusivt ORet med inverteringen og derpå ANDed med masken indtil der op-

står en værdi, der er forskellig fra 0. BASIC bliver hængende i en WAIT løkke, hvis den ønskede kondition ikke opstår. Hvis du indtaster ovenstående eksempel, bliver du nød til at trykke CTRL, SHIFT og ESC ned samtidg for at komme ud af den.

Se også INP, OUT

WEND

```
WEND
```

```
10 MODE 1
20 INPUT "Indlæs timer, minutter og sekun
der       (t,m,s)";t,m,s
30 CLS:d=INT(TIME/300)
40 WHILE t<13
50 WHILE m<60
60 WHILE ti<60
70 ti=(INT(TIME/300)-d)+s
80 LOCATE 70,4
90 PRINT #0,USING "## ";t,m,ti
100 WEND
110 ti=0
120 s=0
130 m=m+1
140 GOTO 30
150 WEND
160 m=0
170 t=t+1
180 WEND
190 t=1
200 GOTO 40
```

Kommando: WHILE/WEND (WEND - While END) udfører et delprogram gentagne gange indtil en given kondition er sand. WEND kommanden afslutter WHILE løkken.

Se også: WHILE

WHILE

```
WHILE logisk udtryk
```

```
WHILE time<0
```

Kommando: WHILE kommandoen definerer "hovedet" på løkken og angiver hvilken kondition, der skal arbejdes efter.

Se også WEND

WIDTH

```
WIDTH %tal
```

```
WIDTH 40
```

Kommando: Fortæller BASIC'en hvilken bredde, der skal udskrives på printeren med, hvorved BASIC'en kan indsætte en nødvendig "vognretur" når der udskrives.

Se også PRINT, POS

WINDOW

```
WINDOW (#stream,)venstre, højre,top, bund

WINDOW #2,10,30,7,18
```

Kommando: Sætter et tekst vindue for en given skærm stream. Se afsn. 3.2.

Se også: ORIGIN

WINDOW SWAP

```
WINDOW SWAP streamudtryk,streamudtryk

WINDOW SWAP 3,5
```

Kommando: Ombytter de to specificerede tekst vinduer. Se afsn. 3.2.

Se også: PAPER, PEN, TAG, WINDOW

WRITE

```
WRITE (#stream,)(liste,...,liste)

WRITE #2,"MOLINT",26,4
```

Kommando: PRINTer værdierne fra listen til en given stream, idet de sepereres med kommaer og strengudtryk sættes i dobbelte gåseøjne.

Se også: PRINT

XOR

```
variabel XOR variabel

10 a$="Januar":b$="Januar"
20 IF a$ XOR b$
```

Logisk operator: Se afsnit 7.3

XPOS

XPOS

PRINT XPOS

Funktion: Sætter den horisontale position af den grafiske cursor.

Se også: MOVE, MOVER, ORIGIN, YPOS

YPOS

YPOS

PRINT YPOS

Funktion: Sætter den vertikale position af den grafiske cursor.

SE også: MOVE, MOVER, ORIGIN, XPOS

ZONE

ZONE %tal

ZONE 19

Kommando: Ændrer bredden af zoner, der anvendes i forbindelse med PRINT fra startværdien på 13 til en ny værdi i området 1 til 255. Stilles igen på 13 ved NEW, LOAD, CHAIN og RUN" kommandoer- ne.

Se også: PRINT, WIDTH

8.2 KONTROLKODER.

I mange operationer placeres tekstcursoren uden for det tekstvindue, man arbejder med. Forskellige arbejdsfunktioner bringer den tilbage (om ikke altid synlig) før de kan udføres:

- printning af en karakter
- indsættelse af cusor' når den skal bruges
- adlyde de kontrolkoder, der i den følgende liste er mærket med en stjerne (*).

Proceduren, der følges for at placere cursor' er:

1) Hvis cursor er til højre for den højre kant, flyttes den til yderste position i næste linie.
2) Hvis cursor er til venstre for venstre kant, flyttes den til yderste position i den foregående kant.
3) Hvis cursor er over den øverste kant, rulles (scrolles) vinduet en linie ned.
4) Hvis cursor er under den nederste kant rulles (scrolles) vinduet en linie op.

Test og operationer udføres i nævnte rækkefølge. Illegale cursor positioner kan være nul eller negative, hvilket er ved siden af venstre kant eller over vinduet.

Kontrolkode karakterer i området 0 til 31, der sendes til skærmen, skriver ikke en karakter på skærmen (og kan låse computeoperationssystem, hvis det sker) men udføres udelukkende som kontrolkarakterer.
Kontrolkaraktererne udskriver deres symbol, hvis de tastes fra tangentbordet (ØG - samtidig tastning af CTRL og G), de vil kun udføre deres funktion, hvis de aktiveres ved PRINT CHR(7) (se også afsnit 2.5 og nøgleordet TAG).

Koderne, der er mærket med en stjerne vil placere cursoren på en legal position i vinduet før de adlydes.

Koderne er beskrevet med deres hexadecimale værdi, deres navn, parametre og deres funktion.

Hex Værdi	Navn	Parametre	Funktion
00	NUL		Ingen effekt - udeladt.
01	SOH	0-255	PRINT det symbol, der er angivet ved parametrene. Dette symbol tillader karaktererne i området 0-31 at blive udskrevet.
02	STX		Afbryder tekstcursor.
03	ETX		Tænder for tekstcursor. Bemærk at BASIC bruger en speciel cursor, der kun kommer i brug, når der afventes en indtastning fra tangentbordet.
04	EOT	0-2	Sætter skærmformatet. Arbejder efter MOD 4. Svarer til MODE n.
05	ENQ	0-255	Sender parameter til grafisk cursor.
06	ACK		Tillader tekstskærm (se 15,NAK, næste side).
07	BEL		Klokke (bemærk at den forrykker SOUND køen (Sound Queues).
08*	BS		Flytter cursor en plads til venstre.
09*	TAB		Flytter cursor en plads til højre.
0A*	LF		Flytter cursor en linie ned.
0B*	VT		Flytter cursor en linie op.
0C	FF		Sletter tekstvindue og placerer cursor i øverste venstre hjørne. Svarer til CLS.
0D*	CR		Flytter cursor til venstre kant på ny linie.
0E	SO	0-15	Sætter INK til papirfarven. Arbejder efter MOD 16. Svarer til PAPER.
0F	SI	0-15	Sætter pen INK. Arbejder efter MOD 16. Svarer til PEN.
10*	DLE		Sletter den øjeblikkelige karakter. Fylder karaktercellen med den nuværende paper INK.
11*	DC		Sletter fra venstre kant af vinduet til den øjeblikkelige karakterposition (incl.). Fylder de berørte karakterceller med den nuværende paper INK.
12*	DC2		Sletter fra den nuværende karakter (incl.) til højre kant af skærmen. Fylder karaktercellerne i berørte område med nuværende paper INK.
13*	DC3		Sletter fra vinduets øverste venstre hjørne og til nuværende karakter (incl.) Fylder de berørte celler med nuværende paper INK.
14*	DC4		Sletter fra nuværende position (incl.) til vinduets nederste højre hjørne. Fylder cellerne med nuværende paper INK.
15	NAK		Slukker tekst skærmen (den modtager intet efter en ACK(06)).
16	SYN	0-1	Parametre MOD 2. 0 slukker for den transparante funktion, der tændes med 1.
17	ETB	0-3	Parameter MOD 4. 0 sætter normal grafisk INK mode. 1) 'XOR''' 2) 'AND''' 3) 'OR'''
18	CAN		Skifter pen og paper INK.
19	EM	0-255 0-255 0-255 0-255	Sætter matrix for brugerdefinerbare karakterer. Svarer til SYMBOL. Tager 9 parametre. Det første specificerer hvilken karakter, der skal sættes. De næs-

		0-255	te 8 specificerer matrixen. Den mest bety-
		0-255	dende bit i den første byte er den øverste
		0-255	venstre celle i karakteren. Den sidste
		0-255	mindst betydende bit i den sidste byte er
		0-255	den nederste højre cele i karakteren.
1A	SUB	1-80	Sætter et vindue. Svarer til WINDOW.
		1-80	De første 2 parametre specificerer venstre
		1-25	og højre kant af vinduet - den mindste
		1-25	bruges til venstre kant, den største til højre kant. De sideste 2 parametre specificerer vinduets top og bund - den mindste tages som top, den største som bund.
1B	ESC		Ingen effekt. Udeladt.
1C	FS	0-15	Sætter INK til et sæt af 2 farver. Det
		0-31	første parameter (MOD 16) specificerer INK
		0-31	- de næste to (MOD 32) er de ønskede farver. Svarer til INK kommandoen.
1D	GS	0-31	Sætter BORDER til et sæt af 2 farver. Sva-
		0-31	rer til BORDER. De to parametrer (MOD 32) specificerer de to farver.
1E	RS		Flytter cursor til det øverste venstre hjørne i vinduet.
1F	US	1-80	Flytter cursor til den givne position i det
		1-25	øjeblikkelige vindue. Svarer til LOCATE. Det første paprameter angiver kolonnen, det andet linien.

8.3 FEJLKODER.

Når BASIC møder en programordre, et ord eller en variabel, den ikke forstår, vil den stoppe og udskrive en fejlkode. Fejlkoden indikerer, hvad der er galt - og hvis fejlen er en typisk skrivefejl under programskrivningen, bliver fejllinien udskrevet i editerings-mode.

Se fejlbehandlingsrutinerne ERL, ERR, ON ERROR GOTO, RESUME.

Fejlbehandlingsrutinerne vil ikke fungere for fejlbehæftede indtastninger, der er overholder BASIC syntaksen.

Følgende liste indeholder en oversigt over fejlkoderne i nummerisk rækkefølge.

1 Unexpected NEXT

En NEXT kommando er indlagt i programmet uden et forudgående FOR, eller kontrolvariablen i NEXT kommandoen svarer ikke til den i FOR sætningen.

2 Syntax Error

BASIC forstår ikke den givne linie på grund af en konstruktionsfejl i linien (f.eks. manglende mellemrum).

3 Unexpected RETURN

Der er en RETURN kommando, der ikke hører til en GOSUB ordre, eller programmet er fejlkonstrueret, så det passerer RETURN før det er nået til GOSUB.

4 DATA exhausted

En READ kommando har forsøgt at læse flere data, end der er specificeret i den tilhørende DATA sætning.

5 Improper argument

En generel fejludskrift. Værdien af en funktions argument eller parametrene til en kommando er fejlbehæftede på en eller anden måde.

6 Overflow

Resultatet af en aritmetisk operation ligger over det område computeren kan behandle. Er det decimaltal er grænsen på 1.7E-38 overskredet. Kan også værer et forsøg på at omdanne et et decimaltal til et heltal (%tal), der skal ligge i området -32768 til +32767.

7 Memory full

Programmet og dets variabler er for stort, eller den kan være for mange GOSUB, WHILE eller FOR løkker lagt ind i hinanden.
En MEMORY kommando vil give denne fejl, hvis hukommelsen lægges for lavt eller for højt. Bemærk, at en åben kassette fil (og et pladelager) bruger en del af hukommelsen til styreordrer og derfor kan influrere på mængden af fri hukommelse.

8 Line does not exist

Den linie der er refereret til eksisterer ikke.

9 Subscript out of range

En af værdierne i et område er enten for stort eller for lille.

10 Array already dimensioned

Et af områderne i en DIM kommando er allerede dimensioneret.

11 Division by zero

Der kan matematisk set ikke divideres med nul. Kontroller om den variabel, der divideres med, får værdien nul på noget tidspunkt i programkørslen.

12 Invalid direct command

Den sidst indtastede direkte kommando var ikke korrekt.

13 Type mismatch

En nummerisk værdi er fundet, hvor det skulle have været en streng, eller et forkert nummerisk udtryk er blevet fundet i READ eller INPUT (komma i stedet for decimalpunktum eller et bogstav i talrækken.

14 String space full

Der er dannet så mange strenge, at der ikke er mulighed for flere, en ikke efter "oprydning" (garbage collection).

15 String too long

Strengens længde overskriver 255 karakterer. Kan opstå ved addering af to eller flere strenge.

16 String expression too complex

Streng-udtryk kan generere et antal interne mellemudregninger, der kan kan få BASIC'en til at opgive.

17 Cannot CONTinue

Af en eller anden grund kan programmet ikke genstartes ved brug af CONT. Bemærk at CONT ikke kan bruges, hvis programmet ændres efter det er standset af STOP, break, eller en fejlmelding.

18 Unknown user function

Der er forsøgt at udføre en FN funktion, der ikke først er defineret med kommandoen DEF FN.

19 RESUME missing

Afslutningen på programmet er nået, medens computeren var i gang med en fejlprocedure-rutine (f.eks. ON ERROR GOTO).

20 Unexpected RESUME

RESUME kan kun anvendes i forbindelse med fejlprocedure-rutiner (f.eks. ON ERROR GOTO), og er fundet uden for en sådan.

21 Direct command found

Hvis der ved indlæsning fra kassette findes en linie uden linienummer.

22 Operand missing

BASIC'en har fundet et udtryk, der ikke er komplet.

23 Line too long

En linie der er fundet for lang, når den er konverteret til BASIC'ens interne form.

24 EOF met

Der er gjort forsøg på at læse forbi afslutningen på en kassette indlæsningsstream.

25 File type error

Kassettefilen, der forsøges indlæst, er ikke af rette type. OPENIN kan kun indlæse ASCII tekstfiler. LOAD, RUN", etc.

kun beregnet til at indlæse de filtyper, der er dannet ved SAVE.

26 NEXT missing

BASIC'en kan ikke finde et NEXT til en FOR kommando.

27 File already open

En OPENIN eller OPENOUT kommando er forsøgt udført før en tidligere åbnet fil er blevet lukket.

28 Unknown command

BASIC'en kan ikke den indtastede ordre indbygget.

29 WEND missing

BASIC'en kan ikke finde et WEND til en WHILE kommando.

30 Unexpected WEND

Der er fundet et WEND uden en foregående WHILE kommando, eller et WEND der ikke svarer til den aktuelle WHILE-løkke.

Vedr. læsefejl ved brug af båndoptageren, se side 11.

INDEX

BRYNDUM OFFSET