AMSTRAD

CPC 464

BRUGER-VEJLEDNING

CPC464 FARVE PERSON-DATAMATEN

64K

Brugen af denne brugervejledning

Datamaterne har udviklet sig umådeligt meget på ganske kort tid. Blandt alle de teknologiske fremskridt i Det tyvende Århundrede er datamatikkens klart det mest overraskende.

Selve datamaterne såvel som deres programmer har været igennem en udvikling, der er gået så hurtigt, at selv de bedste brugere har haft svært ved at følge med. Hvis vi skulle forsøge at vise ejerne af CPC464 al den kraft og alle de finere detaljer i denne datamats BASIC, dens styresystem og detaljerne i dens opbygning, skulle vi bruge adskillige tusinde tekstsider.

Denne brugervejledning er derfor en kortfattet indføring i CPC464 og dens programmer. Den vil blive fulgt op af mange flere specifikke og detaljerede instruktionskurser og publikationer.

Brugere, der allerede er inde i andre BASIC-dialekter, vil hurtigt kunne sætte sig ind i AMSTRAD BASICs struktur - og nybegyndere vil hurtigt lære at sætte pris på det direkte og enkeltydige ordvalg, der anvendes. Denne BASIC er specielt udformet for at undgå de mange særheder, man finder i andre BASIC-dialekter, samtidig med at den medtager grundlæggende muligheder for at programmere "sandtid", der ikke tidligere har været til rådighed i en så billig datamat.

Brugervejledningen er delt i tre afsnit

Det første er det grundlæggende begynderkursus, der er specielt skrevet for at indføre en nybegynder i datamater og dataspoget. Hvis du ikke tidligere har haft en datamat eller har skrevet et lille program selv, så råder vi dig til at arbejde dig igennem begynderkurset

De, der har forudgående erfaring med datamater, kan starte med kapitel 1. Vi har i dette gentaget en del vigtige oplysninger om, hvordan man stiller datamaten op og gør sig bekendt med den, men vi har især koncentreret os om at gennemgå de særlige træk ved CPC464-systemet, idet vi går ud fra, at du har en vis viden om terminologien.

Hvert af de indledende kapitler sigter mod i brede træk at beskrive nogle af de spændende punkter ved CPC464. Nogle grundlæggende punkter gentages for at understrege dem - og fordi mange brugere har lyst til at kaste sig direkte ud i lyd og grafik med den kortest mulige indføring i tastaturet og BASICen.

AMSTRADs *Vejledning i BASIC* træningskursus er beregnet på en mere gennemgribende indføring i forståelsen af de mange sider af din CPC464 og dens næsten ubegrænsede anvendelse som undervisningsmiddel, spillemaskine og *ren* datamat, og vi tilråder kraftigt, at du anskaffer dig denne bog, hvis du ønsker at lære dette fuldstændigt.

Til sidst i instruktionsbogen finder du en række appendikser, der skal give dig en bred oversigt over datamater og mere specifikke træk ved denne datamat.

Vi ønsker dig held og lykke. Du kunne ikke have valgt dig noget bedre for pengene, ejheller en datamat med større muligheder for udvikling af din forståelse for alle sider af sagen. Der er ingen bedre måde, hvorpå man kan lære om datamater end ved at bruge en datamat - og CPC464 er særligt *brugervenlig* til dette.

AMSOFT

A division of

AMSTRAD

CONSUMER ELECTRONICS PLC

Produktet, der beskrives i denne bog, og produkter til brug med det, er under stadig udvikling og forbedring. Al slags tekniske og detaljeret information om produktet og dets brug (herunder oplysninger og detaljer i denne bog) er givet i god tro af AMSTRAD. Det anerkendes imidlertid, at der kan være fejl og udeladelser i denne bog. En liste over eventuelle ændringer og tilføjelser kan fås ved skriftlig henvendelse til importøren, Dinamicro.

AMSOFT tager med tak imod kommentarer og forslag til produktet eller brugervejledningen. Al korrespondance skal ske til:

AMSOFT
169 Kings Road
Brentwood
Essex CM14 4EF
U.K.

Al service og reparation af produktet skal udføres af AMSOFTs autoriserede forhandlere. Hverken AMSOFT eller AMSTRAD kan gøres ansvarlig for noget tab eller nogen skade, forårsaget af service eller reparation, foretaget af uautoriserede folk. Denne vejledning er beregnet på at hjælpe læseren i brugen af produktet, og hverken AMSOFT eller AMSTRAD kan derfor gøres ansvarlig for noget tab eller nogen skade, der opstår ved brugen af disse oplysninger eller detaljer, ejheller for nogen fejl eller udeladelse i denne vejledning eller ukorrekt brug af produktet.

Brugen af termen Z80 i denne bog sker efter tilladelse fra Zilog Inc.

Første udgave 1984
Udgivet af AMSTRAD
Oversat af Svend Garbarsch/Rolf Clausen

VIGTIGT

Under læsningen af denne brugervejledning bedes du bemærke, at der anvendes forskellige satstyper for at henvise til programmer fx |**TASTER**|, der findes i datamaten, men ikke giver tegn (karakterer) på skærmen og *almen beskrivelse,* som er knyttet til programmeringsord, men ikke skal indtastes som del af et program.

1. Tilslut aldrig datamat, skærm (monitor) eller netdel til noget andet udstyr eller til anden kraftkilde end den i bogen beskrevne. Du risikerer alvorlige skader og mister din garanti.

2. Hold blomstervaser, drikkevarer og lignende langt væk fra datamaten, skærme og netdelen. Hvis du spilder væsker på disse dele, kan det resultere i alvorlige skader. Er uheldet alligevel sket, bør du kontakte servicefolk.

3. Du må ikke blokere eller dække ventilationshullerne ovenpå eller bagpå datamaten, skærmen eller netdelen.

4. Når du slukker strømmen, mister du alle data i datamaten. Læs kapitel 2 efter *Indføringskurset*, hvis du vil gemme et program.

5. Det anbefales at bruge specielle datakassettebånd til programmer, men der er ikke noget i vejen for at bruge lydbånd af et godt anerkendt mærke, blot der ikke er tale om Krom- eller Metalbånd, og båndene ikke er længere end 90 minutter (C-90).

For at du lettere og hurtigere kan finde tilbage til dine programmer, anbefaler vi imidlertid, at du anvender C-12 kassetter (med seks minutter på hver side).

6. Bemærk, at programmer optaget på kassettebånd fra andre datamater ikke kan anvende på en CPC464.

7. Hvis den kassette, du bruger, har fået sikkerhedstapperne fjernet for at sikre mod sletning ved en fejltagelse, kan optageknappen ikke nedtrykkes på båndoptageren. Forsøg ikke at bruge magt, du risikerer at beskadige mekanismen. Hvis du ønsker at genbruge sådanne bånd, må du dække hullerne, hvor tapperne sad, med en klæbestrimmel.

8. Hvis der er en indføringstape i en anden farve på din programkassette, må du spole båndet frem forbi denne, inden du begynder optagelsen af et program.

9. Pas godt på ikke at bruge eller opbevare nogle af enhederne i direkte solskin eller på særligt varme, kolde, våde eller støvede steder. Undgå også steder med kraftige rystelser. Gem aldrig programmer nær stærke magnetiske felter, såsom højttalere eller elmotorer.

10. Almindelig omhyggelig behandling af dine bånd og jævnlig rengøring af dine bånd og båndoptageren skulle sikre dig fejlfri optagelse og gengivelse af dine programmer.

11. Inden i enhederne findes intet, som brugeren bør betjene. Forsøg ikke at lukke dem op, men overlad al service til kvalificeret servicepersonale.

INDHOLD

Om denne brugervejledning

Indføringskursus for begyndere

1 Opstarten

2 Kassetteoptageren

3 BASICskolen

4 Variabler, operatorer og data

5 Grafikkursus

AMSTRAD CPC464
INDFØRINGSKURSUS FOR BEGYNDERE

Grundlaget 1:
Opstillingen

De første instrukser i udpakning, forbindelse og tilslutning af dit CPC464-system.

AMSTRAD CPC464 farve-persondatamaten kan opstilles ved brug af enten:

1.1 AMSTRAD GT64 med grøn skærm
1.2 AMSTRAD GTM640 med farveskærm
1.3 AMSTRAD MP1 netdel og et UHF-fjernsyn

Se venligst under det rette afsnit, så du kan forbinde dit datamat-system korrekt og fortsætte med instrukserne i brugen.

ADVARSEL

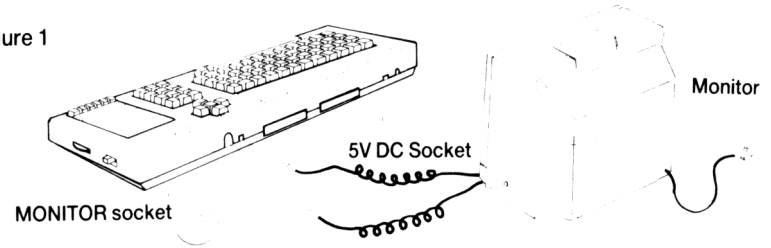
Træk stikket ud af kontakten, når datamaten ikke bruges.

Du skal ikke foretage nogen indvendige tilslutninger.
Du må derfor ikke forsøge at åbne udstyret.

1.1 AMSTRAD GT64 med grøn skærm

Udpak delene og anbring dem som vist med datamaten foran skærmen på et passende bord nær en stikkontakt. Som vist i figur 1 forbindes ledningen med det store 6-polede DIN-stik fra skærmen til en kontakt, der er mærket **MONITOR** bag på datamaten. Forbind derefter ledningen med det mindre DC-stik fra skærmen til kontakten, der er mærket **5VDC** bagpå datamaten.

Figure 1



Du må sikre dig, at der er slukket for skærmen. Knappen mærket **POWER** skal stå på **OFF**, før du går videre.

Sæt nu elstikket fra skærmen ind i stikkontakten (220v vekselstrøm) og tænd for skærmen. Tænd herefter for datamaten ved en skydeknap i højre side mærket **POWER.**

Den røde **ON**lampe ovenpå tastaturet skulle nu lyse, og skærmen viser følgende billede:

```
Amstrad 64K Microcomputer  (v1)
©1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.0

Ready
```

markør

For at undgå at belaste dine øjne unødigt bør du nu regulere lysstyrken på skærmen ned til et passende niveau ved hjælp af knappen **BRIGHTNESS**. Det skal være behageligt at se på skærmen, der hverken må blænde eller gøre bogstaverne uskarpe.

Nu bør du også regulere kontrasten med knappen **CONTRAST** til den laveste styrke, du finder passende.

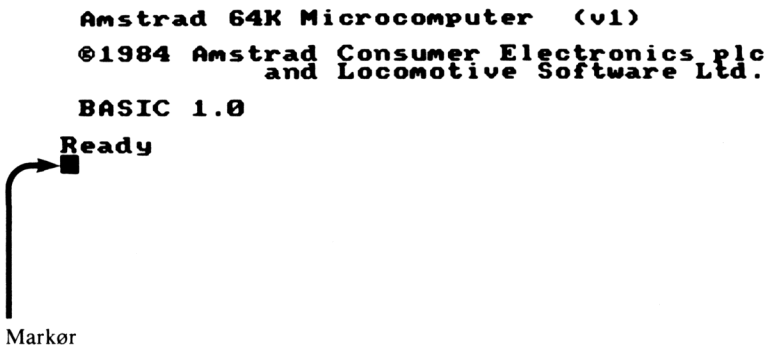
På GT64 er lodret hold mærket **V-HOLD**, og denne kontrolknap skal justeres, så billedet står korrekt midt på skærmen uden at hverken ryste eller *rulle.*

1.2 AMSTRAD CTM640 Farveskærmen

Udpak delene og anbring dem med datamaten foran skærmen på et passende bord nær en stikkontakt. Som vist i figur 1 forbindes ledningen med det store 6-polede DIN-stik fra skærmen til en kontakt, der er mærket **MONITOR** bag på datamaten. Forbind derefter ledningen med det mindre DC-stik fra skærmen til kontakten, der er mærket **5VDC** bagpå datamaten.
Du må sikre dig, at der er slukket for skærmen. Knappen mærket **POWER** skal stå på **OFF**, før du går videre.

Sæt nu elstikket fra skærmen ind i stikkontakten (240v vekselstrøm) og tænd for skærmen. Tænd herefter for datamaten ved en skydeknap i højre side mærket **POWER.**

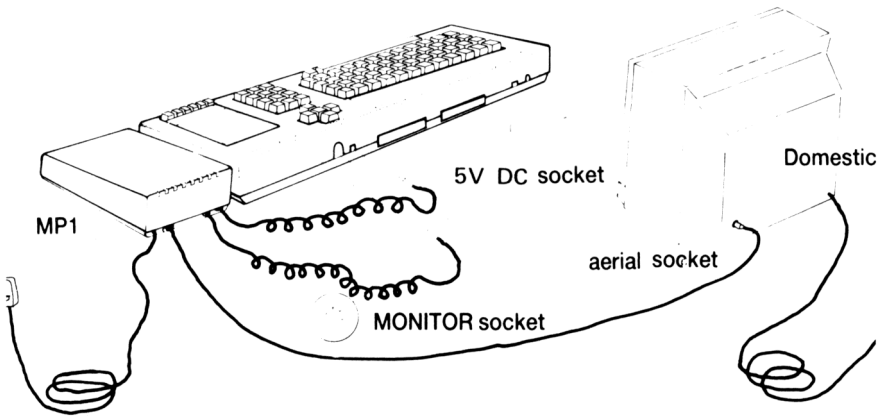
Den røde **ON**lampe ovenpå tastaturet skulle nu lyse, og skærmen viser følgende billede:



For at undgå at belaste dine øjne unødigt bør du nu regulere lysstyrken på skærmen ned til et passende niveau ved hjælp af knappen **BRIGHTNESS**. Det skal være behageligt at se på skærmen, der hverken må blænde eller gøre bogstaverne uskarpe.

1.3 AMSTRAD MP1 netdel og et almindeligt UHF-farvefjernsyn

MP1-netdelen er et ekstraudstyr, som du måske ønsker at anskaffe dig, såfremt du p.t. anvender din CPC464-datamat med den grønne GT64-skærm. Med MP1 kan du slutte datamaten til dit farvefjernsyn og dermed få fuld glæde af din CPC464-datamats farvemuligheder.



Figur 2: MP1-netdelen og strømforsyningen

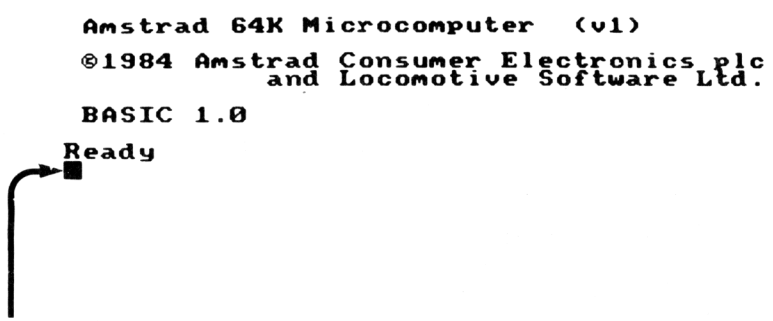
Anbring MP1-netdelen til højre for datamaten, der skal stå på et passende bord tæt ved fjernsynet og en stikkontakt. Som vist i figur 2 forbindes ledningen med det store 6-polede DIN-stik fra MP1 med den kontakt, der er mærket **MONITOR** bagpå datamaten. Forbind også ledningen med det mindre DC-stik fra MP1 til kontakten mærket **5V DC** bagpå datamaten.

Forbind ledningen med antennestikket fra MP1 til antennekontakten på dit fjernsyn.

Se til, at datamaten er slukket. Kontakten **POWER** i højre side skal stå på **OFF**, inden du går videre. Sæt nu stikket fra MP1 i din stikkontakt.

Skru nu ned for lydstyrken på dit fjernsyn, tænd for det og tænd for datamaten med glideknappen mærket **POWER** i højre side.

Nu skulle **ON**-lampen ovenpå datamaten tændes, og nu må du indstille dit fjernsyn, så det modtager signalet fra datamaten. Det ligger på kanal 36. Billedet skulle se således ud:



Markør

Hvis dit fjernsyn har en trykknap-indstilling, skulle du være i stand til at reservere en af knapperne permanent til kanal 36, så du ikke skal lede efter den, hver gang du vil bruge din datamat.

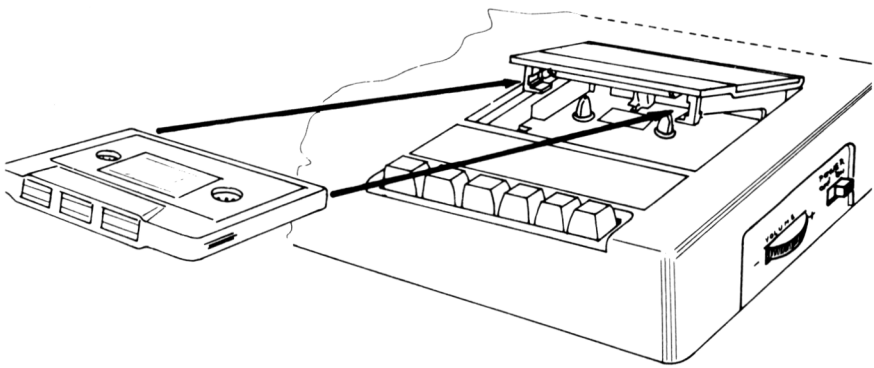
Finindstil fjernsynet, så billedet bliver så klart som muligt. Skriften skal stå gyldent/gult på en dybblå baggrund.

1.4 Styrepind

AMSOFT-styrepinden model JY1 er et ekstraudstyr, som du måske får lyst til at anskaffe dig, dersom du bruger din CPC464-datamat til spil, der rummer mulighed for den slags styring. JY1 kan tilsluttes bag i datamaten ved den ni-polede kontakt mærket **USER PORTS (I/O).** AMSTRAD CPC464-datamaten kan bruge to styrepinde. Den anden JY1-styrepind tilsluttes en kontakt på den første.

1.5 Velkomstkassetten

I en af polystyrol-endestykkerne i din datamatpakke fandt du en velkomstkassette. Luk din dataoptager op ved at trykke på tasten |**STOP/EJECT**| og indsæt kassetten som vist på figur 3, idet du skal sikre dig, at **SIDE 1** vender opad.



Figur 3 Sådan indsættes kassetten rigtigt i dataoptageren

Tryk ned på lågen til den klikker, tryk så på knappen |**REW**| for at sikre dig, at tapen er rullet tilbage til starten. Når tapen stopper, trykker du |**STOP/EJECT**|. Sæt nu optagerens tæller på 000 ved at trykke på knappen **COUNTER RESET.**

Tryk på kontrolknappen mærket |**CTRL**| og tryk *samtidigt* på den lille knap |**ENTER**| helt nede til højre på datamatens tastatur, lige ved siden af dataoptageren. Så vil skærmen vise disse instrukser:

RUN"
Press PLAY then any key:

Tryk nu på knappen |**PLAY**| i dataoptagerens forreste række, indtil den låser, og tryk derefter på en eller anden tast på datamatens tastatur. Nu begynder tapen at køre, og efter få øjeblikke ser du denne melding på skærmen:

Loading WELCOME 1 block 1

Indkøringen af båndet tager ca. 5 minutter. Du kan se, at noget sker, hvis du holder øje med skærmen. Blok-nummeret skifter fra 1 til 2, til 3 o.s.v., indtil båndet standser af sig selv. Så begynder velkomstprogrammet at køre. Bare læn dig tilbage og se på. Programmet kører uendeligt, så når du er blevet træt af det, må du trykke to gange på *undslipningstasten* |**ESC**|. Så standser programmet, og du kan nu trykke |**STOP**| og få kassetten ud. Vend den til side to.

Husk efter indsættelsen af kassetten igen at trykke på knappen |**REW**| for at sikre dig, at båndet er spolet tilbage til starten.

Pres |**CTRL**| og *samtidigt* den lille knap |**ENTER**| nederst til højre på tastaturet. Nu svarer skærmen med instruksen.

RUN"
Press PLAY then any key:

Tryk på tasten |**PLAY**| forrest på dataoptageren, indtil den låser, og tryk derefter på en eller anden tast på datamaten. Nu kører båndet, og skærmen viser denne melding:

Loading WELCOME 2 block 1

Følg de instrukser, der står på skærmen. Programmet beder dig om at indtaste instrukserne en efter en.

1.6 Indkøring af andre programkassetter

VELKOMSTKASSETTEN KAN KUN INDKØRES OG BENYTTES som beskrevet i den foregående sektion (1.5). Ubeskyttede BASIC-programmer kan indkøres ved følgende alternative metoder. Indsæt båndet og spol det tilbage med knappen |**REW**|, indtil dataoptageren standser. Tryk så straks |**STOP/EJECT**|.

Slet datamatens lager ved at trykke på tasterne |**CTRL**|, |**SHIFT**| og |**ESC**| i den nævnte rækkefølge, men hold på hver tast, indtil du har trykket |**ESC**| til sidst. Så tømmes skærmen, og du får datamatens startmelding, som om du lige havde tændt for den.

Udtrykket |**ENTER**| i de følgende instrukser viser, at du må trykke på den ene af de to knapper, der er påtrykt dette. Du skal ikke skrive ordet bogstav for bogstav. Tegnet " får du ved at trykke den ene af de to knapper |**SHIFT**| **samtidig** med, at du trykker på tasten med tallet 2 i tastaturets øverste række. Skriv nu:

Load "" |**ENTER**|

så beder datamaten dig om at trykke på |**PLAY**| og en eller anden tast med ordene:

Press PLAY then any key:

Tryk på knappen |**PLAY**| på dataoptageren, til den låser, og tryk på en af datamatens taster. Så kører båndet, og kort efter melder skærmen:

Loading *programnavnet* block 1

Efterhånden som programmet indkøres, skifter bloknumrene, indtil båndet er slut, og datamaten melder klar med ordet:

Ready

Alternativt kan du også indkøre programmet med det navn, det har. Denne metode bruger du, hvis der er flere forskellige programmer på båndet. Skriv da:

Load "*dit programnavn*" |**ENTER**|

Datamaten beder igen om, at du starter båndet og trykker på en eller anden tast, med ordene:

Press PLAY then any key:

Tryk på |**PLAY**|, indtil knappen låser, og derefter en tast på datamaten.

Nu kører båndet, og hvis det program, du bad om, ikke er i starten af båndet, vil datamaten søge båndet igennem, indtil den finder nøjagtigt den titel, du bad om. Hvis du har stavet forkert, finder den intet program til dig. Vær derfor altid meget forsigtig med at indskrive programmets navn helt korrekt.

Datamaten sladrer om de programmer, den passerer undervejs til det program, du bad om. Hvis den fx passerer et program, der hedder Peter, så skriver den:

Found *Peter* block 1

Datamaten indlæser ikke dette program, men fortsætter gennemsøgningen af båndet, indtil den træffer nøjagtigt den titel, du har bedt om, eller indtil du trykker på knappen |**ESC**| for at standse eftersøgningen.

Når datamaten har fundet det ønskede program - lad os kalde det Ole - så ser du på skærmen meldingen:

Loading *Ole* block 1

Block-tallet skifter, indtil programmet er overspillet, og datamaten melder klar med ordet:

Ready

Indtast så:

run |**ENTER**|

som giver datamaten ordre til at køre det indlæste program. Hvis der allerede var et program i datamatens arbejdslager (hukommelse), vil det blive smidt ud, og det nyindlæste program vil erstatte det.

Man kan også få et program til at kore straks efter indlæsningen ved i stedet for at bede datamaten *loade* det straks bede om *run* ved indkorslen. Du skal simpelthen blot skrive:

run "" |**ENTER**|

...datamaten svarer med ordene:

Press PLAY then any key:

og når du har gjort som beordret, indlæser datamaten programmet og kører det straks efter - uden yderligere ordrer fra tastaturet. Du kan standse det ved at trykke |**ESC**|.

1.7 Indlæsning af programmer, optaget på kassetter

Med de instrukser, du hidtil har fået, vil du kunne indkøre ethvert af de mange programmer, man kan købe på kassetter til CPC464-datamaten. Men se for en sikkerheds skyld efter på de indlæsningsinstrukser, der er trykt på programpakken.

1.8 SAVE

Et program kan gemmes til senere brug. Indsæt en kassette (hvor beskyttelsestapperne ikke er fjernet) i dataoptageren på korrekt vis. Tryk |**REW**| for at køre båndet tilbage til start og husk at trykke |**STOP/EJECT**|, når båndet stopper. Indtast så:

save "programmets navn" |**ENTER**|

og datamaten svarer med:

Press REC and PLAY then any key:

altså *tryk på knappen* |**REC**| og knappen |**PLAY**|, derefter en eller anden tast.

Når du har gjort det (husk enter), vil datamaten svare med:

Saving *programnavnet* block 1

Når programmet er blevet gemt, vil dataoptageren standse, og du vil se klarmeldingen *ready* på skærmen. Tryk så på dataoptagerens |**STOP/EJECT**|, og programmet er gemt.

Bemærk, at du ikke kan gemme købte programmer på dine egne tomme bånd. Det skyldes, at disse programmer er kopieringsbeskyttede for at sikre mod, at uautoriserede folk kopierer dem.

Grundlaget 2:
DUS MED TASTATURET

Vi vil nu forklare, hvorledes visse af datamatens taster fungerer. De, der allerede har erfaring med datamater, kan springe dette afsnit over.

|ENTER|

Der er to taster med |**ENTER**|. Begge har samme funktion. De sender de oplysninger, du har indtastet, ind til forarbejdning i datamaten. Når du har trykket på en af dem, begynder en ny linje på skærmen. Hver gang, du har skrevet en instruks til datamaten, skal denne efterfølges af |**ENTER**|.

Når vi fra nu af skriver |**ENTER**|, betyder det: Tryk |**ENTER**| efter hver instruks eller programlinje!

|DEL|

Denne tast bruges til at slette en karakter (et tegn eller et bogstav), der står til venstre for markøren på skærmen.

Prøv fx at indtaste *abcd* og prøv at slette *d* med |**DEL**|, som forklaret ovenfor. Hvis du holder |**DEL**| nedtrykket, vil *abc* også blive slettet.

|SHIFT|

Der er også to |**SHIFT**|-taster. Hvis du holder en af dem nede, mens de skriver en karakter, så vil du få et stort bogstav eller det symbol, der står øverst på tasten, skrevet på skærmen.

Indtast fx bogstavet **a** og hold derefter en af tasterne med |**SHIFT**| nedtrykket, mens du igen taster a. På skærmen vil der nu stå:

aA

Tryk nogle gange på mellemrumstangenten og prøv det samme med 2-tasten i øverste linje. Så skulle du på skærmen få:

2"

Nu kan du selv eksperimentere lidt videre med tasterne og |**SHIFT**|.

|CAPS LOCK|

Denne tast har en funktion, der ligner |**SHIFT**|, men her behøver du ikke at holde tasten nede. Blot du trykker een gang, får du alt skrevet med store bogstaver, men taltasterne bliver ikke ændret. Prøv en gang at trykke |**CAPS LOCK**| og indtaste:

abcdef123456

På skærmen ser du selvfølgelig, at bogstaverne er blevet de store, mens tallene er uændrede. Hvis du vil have symbolerne over tallene frem, mens |**CAPS LOCK**| er i funktion, må du holde på |**SHIFT**|, før du trykker på et tal. Hvis du holder på |**SHIFT**|, mens du igen skriver rækken:

abcdef123456

så står der nu på skærmen:

ABCDEF!" # $%&

Hvis du vil til små bogstaver igen, må du pånyt trykke på |**CAPS LOCK**|.

Hvis du vil skrive både store bogstaver *og* de skiftede tegn uden hele tiden at skulle holde på |**SHIFT**|, så kan det nemt lade sig gøre. Hold på tasten |**CTRL**|, og tryk så *samtidig* på tasten |**CAPS LOCK**|. Prøv så igen at skrive den bogstav-og-tal-række, og det fungerer akkurat som ventet.

Men hvad så, hvis du er helt skør og også gerne ville skrive et tal, mens du har |**CTRL**| og |**CAPS LOCK**| sat på. Jamen, så bruger du bare taltasterne helt til højre på tastaturet.

Hvis du vil ud af ovennævnte funktion, så hold igen på |**CTRL**| og tryk |**CAPS LOCK**|, så er du tilbage ved udgangspunktet: De små bogstaver.

|CLR|

Denne tast bruges til at slette en karakter, der står i markøren.

Prøv at skrive ABCDEFGH. Markøren står til højre for det sidste bogstav, H. Tryk nu fire gange på piletasten, der peger mod venstre, så flytter markøren sig fire positioner til venstre og står nu over bogstavet E.

Du kan stadigvæk se E indeni markøren. Hvis du nu trykker een gang på tasten |**CLR**|, så sletter du E, og bogstaverne FGH rykker en plads til venstre. Nu står F indeni markøren. Hvis du nu prøver at holde |**CLR**| nedtrykket, vil du se markøren æde både F, G og H.

|**ESC**|
Denne tast, hvor ordet er en forkortelse af det engelske **escape** (undslip), bruges til at slippe fri fra en funktion, datamaten er i gang med. Hvis du trykker een gang på |**ESC**|, så holder datamaten en pause i en funktion, men fortsætter udførelsen, hvis du trykker på en eller anden tast.

Hvis du trykker to gange på |**ESC**|, så opgiver datamaten helt den funktion, den var i gang med og er parat til at få nye instrukser fra dig. Prøv nu at trykke to gange på |**ESC**|.

VIGTIGT
En skærmlinje kan rumme 40 karakterer. Hvis du skriver 41 eller flere på en linje, så fortsætter datamaten automatisk på næste linje i venstre side af skærmen. Det betyder, at hvis du er vant til at skrive på en elektrisk skrivemaskine, må du endelig passe på ikke at trykke |**ENTER**| efter en linje, sådan som du er plejer på skrivemaskinen.

Datamaten gør det automatisk for dig, og hvis du kommer til at trykke |**ENTER**|, vil den svare med en såkaldt fejlkode, sædvanligvis med *syntax error,* med det samme, eller når programmet køres.

Syntaksfejl

AMSTRAD CPC464-datamaten er forsynet med et regelsæt, kaldet syntaksen. Det er dens BASIC-grammatik, der skal fortælle den, om de instrukser, den modtager, er korrekte. Hvis meldingen *syntax error* viser sig på skærmen, vil datamaten derfor fortælle dig, at du har begået en fejl. Du har skrevet en instruks, den ikke forstår. Prøv fx at indtaste:

printt|**ENTER**|

straks ser du fejlmeldingen på skærmen, for datamaten forstod ikke ordet *printt.*

Hvis du laver en sådan fejl midt i en programlinje som fx:

1Ø printt ''abc''|**ENTER**|

så kommer meldingen *syntax error* først, når datamaten under programkørslen kommer til den linje.

Prøv at indtaste:

run|**ENTER**|

så vil du på skærmen se:

```
Syntax error in 10
10 printt "abc"
```

Denne melding fortæller dig, hvilken linje fejlen var i, og den fremviser linjen med en redigeringsmarkør, så du kan korrigere din fejl.

Tryk nu på tasten med pil til højre, indtil markøren er over et t i ordet *printt*, og tryk på tasten |**CLR**| for at fjerne det uønskede t. Når du derefter trykker |**ENTER**|, indsætter du den rettede linje i datamaten.

Hvis du nu taster: *run* |**ENTER**|, vil du se, at datamaten har accepteret instruksen og har skrevet *abc*.

En præsentation af AMSTRAD BASIC-nøgleordene

I kapitel 8 finder du en illustreret beskrivelse af alle de BASIC-nøgleord, der er i AMSTRAD BASIC. Vi vil her præsentere nogle af de mest almindeligt anvendte BASIC-nøgleord.

CLS

Indtast cls (clear screen, d.v.s. rens skærmen). Du kan bruge enten store eller små bogstaver, ganske som du lyster. Tryk så |**ENTER**|. Nu er skærmen ren, bortset fra ordet *Ready* (klar) og den lille sorte markør i skærmens øverste venstre hjørne.

PRINT

Denne ordre bruges, når du ønsker at få karakterer, ord eller figurer i et program *trykt* på skærmen. Prøv at indtaste denne instruktionslinje:

print ''hallo''|**ENTER**|

På skærmen vil du nu se:

hallo

Citationstegnene '''' bruges til at fortælle datamaten, hvad den skal skrive på skærmen. Indtast nu cls |**ENTER**| for at slette skærmbilledet.

RUN

Det foregående eksempel viste et program på en enkelt linje. De fleste programmer består af flere linjer. Foran hver linje sætter man et nummer for at fortælle datamaten, i hvilken rækkefølge den skal køre programlinjerne. Når man trykker |**ENTER**|, gemmes linjen i datamatens arbejdslager, indtil programmet køres. Indtast:

1Ø print ''hallo'' |**ENTER**|

Bemærk, at da du trykkede på |**ENTER**|, blev *hallo* ikke trykt på skærmen. For at få dette til at ske må du skrive *run*. Indtast:

run |**ENTER**|

Nu dukkede *hallo* op på skærmen. I stedet for hele tiden at skrive *print* kan du nøjes med tegnet ?, fx:

1Ø ? ''hallo'' |**ENTER**|

LIST

Når et program er gemt i arbejdslageret, er det muligt at kontrollere, hvad der er indskrevet, gennem en såkaldt LISTning af programmet. Indtast:

list |**ENTER**|

og på skærmen vil du se:

1Ø PRINT ''hallo''

som er det program, du gemte i arbejdslageret.

Bemærk, at ordet PRINT nu står med store bogstaver. Det er datamatens måde at fortælle dig, at den har genkendt det som et BASIC-nøgleord.

Indtast cls |**ENTER**| for at slette skærmbilledet. Skønt billedet nu er slettet med cls (der betyder clear screen eller slet skærmen), så er programmet ikke slettet i arbejdslageret.

GOTO

Nøgleordet GOTO (d.v.s. gå til) er en ordre til datamaten om at gå frem til en anden linje - enten for at springe nogle linjer over eller for at danne en løkke. Indtast:

1Ø print ''hallo'' |**ENTER**|
2Ø goto 1Ø |**ENTER**|

Indtast så:

run |**ENTER**|

og ordet *hallo* bliver nu uophørligt skrevet på skærmen i en lang spalte ude i venstre side. For at standse programmet må du trykke een gang på |**ESC**|. Med et enkelt tryk på en tast kan du starte det igen. Hvis du skal standse det helt, må du trykke to gange på |**ESC**|.

Skriv nu:

cls |**ENTER**|

for at slette skærmen.

Hvis du vil se *hallo* skrevet over hele skærmen, må du indtaste programmet igen, men denne gang med et semikolon i linje 1Ø, som vist her:

1Ø print ''hallo''; |**ENTER**|

Et semikolon fortæller datamaten, at den skal skrive den næste karakter umiddelbart efter den foregående. Igen undslipper du fra programmet ved at trykke to gange på |**ESC**|. Prøv så at skrive linje 1Ø igen, men denne gang med et komma i stedet for semikolon.

Nu kan du se, at et komma virker som tabulatoren på en skrivemaskine. Kommaet fortæller datamaten, at den næste karakter skal stå 13 pladser til højre for den foregående - en nyttig funktion, hvis du vil opstille en tekst i kolonner. Bemærk dog, at hvis der er mere end 12 bogstaver i et ord, vil det næste bogstav, efter kommaet, blive flyttet yderligere 13 pladser frem.

Du trykker igen to gange |**ESC**| for at slippe ud af programmet. Man sletter det helt fra arbejdslageret ved at trykke på |**CTRL**| |**SHIFT**| og |**ESC**| i den nævnte rækkefølge, så sker der en total sletning såkaldt *reset*.

INPUT

Denne ordre bruges til at fortælle datamaten, at den skal forvente en indtastning - fx et svar på et spørgsmål. Indtast dette:

1Ø input ''hvor gammel er du?''; alder |**ENTER**|
2Ø print ''du ser da ikke ud som en paa''; alder ''aar'' |**ENTER**|
run |**ENTER**|

På skærmen står nu:

hvor gammel er du?

Du svarer med et tal, nemlig din alder, og |**ENTER**| og hvis du fx svarede 18, står der nu på skærmen:

du ser du ikke ud som en paa 18 aar

Dette eksempel viser dig brugen af *input*-kommandoen og en talvariabel. Ordet *alder* blev lagt ind i arbejdslageret i linje 10, for at datamaten skulle forbinde dette ord med ethvert tal, der blev indtastet og *printe* dette tal, hvor ordet *alder* står i linje 20. Skønt vi bruger udtrykket *alder* i ovenstående som en variabel med navnet *alder,* kunne vi lige så let have brugt et bogstav, fx b.

Reset nu datamaten med |**CTRL**| |**SHIFT**| og |**ESC**|-tasterne. Hvis du havde ønsket at få et *input* i form af bogstaver eller bogstaver og tal, skulle du have sat et $-tegn i enden af variablen.

Prøv nu at indtaste følgende program (bemærk, at du i linje 20 skal sætte et mellemrum efter s i davs og før j i jeg):

```
1Ø input ''hvad hedder du?''; navn$ |ENTER|
2Ø print ''davs ''; navn$'' jeg hedder Arnold'' |ENTER|
run |ENTER|
```

På skærmen ser du nu:

hvad hedder du?

Nu skriver du så dit navn og |**ENTER**|. Hvis du nu hedder Peter, så står der nu:

davs Peter jeg hedder Arnold

(For kuriøsitetens skyld skal oplyses, at Arnold var kodenavnet for AMSTRAD CPC464 under udviklingen af datamaten).

Vi brugte navn$ som en strengvariabel i ovennævnte, men kunne lige så godt have brugt et bogstav, fx a$. Nu vil vi sammenstille de ovennævnte to eksempler til eet program.

Reset datamaten med |**CTRL**| |**SHIFT**| og |**ESC**|. Indtast så følgende og husk at trykke |**ENTER**| efter hver linje:

```
5 cls
1Ø input ''hvad hedder du?; a$
Reset nu datamaten med |CTRL| |SHIFT| og |ESC|.
Reset datamaten med |CTRL| |SHIFT| og |ESC|.
   b''aar''
run
```

I dette program har vi brugt to variabler,a$ for navnet og b for alderen. På skærmen vil du se:

hvad hedder du?

Nu indskriver du dit navn og trykker |**ENTER**|. Så spørger datamaten om din alder, og her svarer du med et tal og |**ENTER**|. Hvis du nu hedder Peter og er 18 år, så siger datamaten nu:

det maa jeg virkelig sige, Peter , du ser ikke ud som en paa 18 aar

REDIGERING AF ET PROGRAM

Hvis man har indtastet forkerte linjer i et program og har fået en *syntax error* eller anden fejlmelding, så kan man redigere den eller de forkerte linjer i stedet for at skrive det hele om. For at prøve dette kan du nu indtaste det tidligere program med nogle fejl. Husk |**ENTER**| efter hver linje:

```
5 clss
1Ø input ''hvad heder du?''; a$
2Ø input ''hvor gammel er du?''; b
3Ø print ''det maa jeg virkelig sige''; a$'' du ser ikke ud som en paa''; b''aar''
```

Der er tre fejl i dette program:

I linje 5 skrev vi et s for meget i *cls*
I linje 10 skrev vi *heder* i stedet for *hedder*
I linje 30 glemte vi et mellemrum mellem *sige* og citationstegnet

Der er tre måder, hvorpå man kan redigere programmet. Den første er simpelthen at indtaste den nye linje igen. Hvis man skriver en linje med samme nummer som en, der allerede står i et program, så erstattes den gamle linje automatisk med den nye. Den anden måde er ved *edit*-metoden, og den tredje hedder *kopimarkør*-metoden.

EDIT METODEN

For at rette fejlen i linje 5 skriver du:

edit 5 |**ENTER**|

så skrives linje 5 med markøren oveni 5-tallet. For at fjerne det ekstra s i *clss*, trykker du på højrepilen, indtil markøren står over det sidste s. Tryk så på tasten |**CLR**|, så forsvinder s'et. Tryk nu |**ENTER**| og læg dermed den rettede linje 5 ind i arbejdslageret. Hvis du skriver *list* |**ENTER**|, kan du efterkontrollere dette.

KOPIMARKØR-METODEN

For at rette fejlene i linje 10 og linje 30 skal du holde på |**SHIFT**| og derefter trykke på oppilen, indtil *kopimarkøren* står ved begyndelsen af linje 10. Læg mærke til, at hovedmarkøren ikke har flyttet sig, således at du nu har to markører på skærmen. Tryk nu på tasten |**COPY**|, indtil kopimarkøren står over mellemrummet mellem *heder* og *du*. Du vil opdage, at linje 10 repeteres på sidste linje, og at hovedmarkøren standser på samme sted som kopimarkøren. Indtast nu bogstavet r, det dukker kun op på nederste linje.

Hovedmarkøren har flyttet sig, men kopimarkøren blev, hvor den var. Tryk nu på |**COPY**|-tasten, indtil hele linje 10 er trykt. Når du nu trykker |**ENTER**|, lægger du den nye linje 10 ind i arbejdslageret. Kopimarkøren forsvinder, og hovedmarkøren stiller sig under den nye linje 10.

For at rette den anden fejl holder du på |**SHIFT**| og trykker på oppilen, indtil kopimarkøren står ved begyndelsen af linje 30. Tryk nu |**COPY**|, indtil kopimarkøren står over citationstegnet ved siden af *sige*. Tryk nu een gang på mellemrumstasten. På nederste linje ser du et mellemrum. Hold nu på tasten |**COPY**| indtil hele linje 30 er trykt. Tryk |**ENTER**|, og hvis du nu *lister* programmet, kan du se, at fejlene er rettet i arbejdslageret.

Reset nu datamaten med |**CTRL**| |**SHIFT**| og |**ESC**|.

IF THEN

Vi vil nu udvide det foregående program med brug af kommandoerne *if* og *then*.

Indtast det følgende og bemærk to nye tegn: <, der betyder *mindre end* og >, der betyder *større end*. De står begge ved M-tasten. Husk at taste |**ENTER**| efter hver nummereret programlinje

```
5 cls
1Ø input ''hvad hedder du?'';a$
2Ø input ''hvor gammel er du?'';alder
3Ø if alder < 13 then 6Ø
4Ø if alder < 2Ø then 7Ø
5Ø if alder > 19 then 8Ø
6Ø print ''ja så, '';a$'' . Du er ikke helt teenager, naar du er''alder''aar'':end
7Ø print ''ja så, '';a$'' .Du er teenager, naar du er''
 alder''aar'':end
8Ø print ''ja,ja, '';a$'', saa kan man jo ikke ligefrem kalde dig en teenager, naar du er''al-
der''aar''
```

For at undersøge, om programmet er rigtigt, taster du:

list |**ENTER**|

skriv så:

run |**ENTER**|

og besvar datamatens spørgsmål

Du kan nu let se, hvordan kommandoerne *if* og *then* virker i et program. Vi har også indføjet et nyt ord, *end,* sidst i linjerne 60 og 70. Dette nøgleord hindrer datamaten i at gå til næste linje. Hvis der fx ikke var et *end* sidst i linje 60, ville programmet blot fortsætte med linjerne 70 og 80. Kolonnet før ordet *end* er et skilletegn. Med brug af kolon kan du skrive to eller flere instrukser ind i en og samme programlinje. Vi har anbragt et *cls* i starten af programmet for at viske skærmen ren. Det vil vi gøre i alle de følgende programmer, så tingene ser pænere ud.

Andre nøgleord, der er knyttet til *if* og *then* er *else, or* og *goto*. Brugen af disse ord i *if*-ordren vil stå klart for dig, efterhånden som du arbejder dig gennem denne vejledning.

Reset datamaten med |**CTRL**| |**SHIFT**| og |**ESC**|.

FOR TO NEXT

Vi vil nu vise brugen af ordrerne *for, to* og *next*. Som et eksempel på dette, vil vi vise, hvordan datamaten skriver 12-tabellen. Indtast det følgende og bemærk, at ★ er datamatens tegn for *gange* eller *multiplikation. Husk* |**ENTER**| efter hver nummereret programlinje.

```
5 cls
1Ø for a = 1 to 2Ø
2Ø print a''★ 12 =''a★12
3Ø next a
run
```

Kolonnerne ser lidt rodede ud. Indtast i stedet:

```
5 cls
1Ø for a = 1 to 9
2Ø print a'' ★ 12 =''a★12
3Ø next a
4Ø for a = 1Ø to 2Ø
5Ø print a''★ 12 =''a★12
6Ø next a
run
```

Du kan selvfølgelig anvende dette program til andre tabeller (fx 17-tabellen). Du skal blot udskifte 12-tallet i linjerne 20 og 50 med 17.

Det er muligt at fastsætte værdien af hvert skridt i en for-next-kommando ved hjælp af ordren *step*. Du kan få yderligere oplysninger om *for* i kapitel 8.

Reset nu datamaten med |**CTRL**| |**SHIFT**| og |**ESC**|.

SIMPEL ARITMETIK

Din AMSTRAD CPC464-datamat kan let anvendes som regnemaskine.

For at forstå dette, må du prøve de følgende eksempler. Vi bruger ? i stedet for ordet *print* i dette afsnit. Så snart du trykker på |**ENTER**|, kommer det svar, vi har skrevet under regnestykket.

ADDITION

Skriv:
?3 + 3 |**ENTER**|
6

(bemærk, at du ikke skriver noget = tegn)

Skriv:
?8 + 4 |**ENTER**|
12

SUBTRAKTION

Skriv:
?4-3 |**ENTER**|
1

MULTIPLIKATION

Skriv:
?3★3 |**ENTER**|
9

Skriv:
?8★4 |**ENTER**|
32

DIVISION

Skriv:
?3/3 |**ENTER**|
1

Skriv:
?8/4 |**ENTER**|
2

KVADRATROD

Hvis du vil finde kvadratroden af et tal, må du skrive *sqr* fulgt af en parentes med tallet.

Skriv:
?sqr (16) |**ENTER**|
4

Skriv:
?sqr (1ØØ) |**ENTER**|
1Ø

POTENSOPLØFTNING

Potensopløftning er et tal ganget med sig selv et vist antal gange, f'x 3 i 3. er det samme som 3★3★3 = 27.

Skriv:
?3^3 |**ENTER**|
27

Skriv:
?8^4 |**ENTER**|
4096

KUBIKROD

Du kan let udregne kubikrødder ved en metode som eksemplet under kvadratrod. Hvis du vil finde kubikroden af 27, skriver du:

?27↑(1/3) |**ENTER**|
3

Kubikroden af 125 findes ved at skrive:
?125↑(1/3) |**ENTER**|
5

BLANDEDE BEREGNINGER (+,-,★,/)

Blandede beregninger forstås af datamaten, men datamaten giver visse aritmetiske tegn forret fremfor andre. Førsteret har multiplikation, næsteret har division, så kommer addition og til slut subtraktion. Denne prioritetsrækkefølge gælder kun for disse fire regnearter. Du vil bedre kunne forstå det, når du ser regnestykkerne.

Hvis en opgave så således ud:

3 + 7-2★7/4

Ville du forvente, at datamaten ville regne det således ud:

3 + 7 er 10, 10-2 er 8, 8★7 er 56, 56/4 er 14.

Men faktisk regner datamaten således (idet den vælger efter prioritetsrækkefølgen ★, /, +, -):

2★7 er 14, 14/4 er 3.5, 3 + 7 er 10, 10-3.5 er 6.5 - eller opstillet i regnestykkets form:

3 + 7-2★7/4
= 3 + 7-14/4
= 3 + 7-3.5
= 10-3.5
= 6.5

Du kan bevise dette for dig selv ved at indtaste regnestykket, sådan som det står:

?3 + 7-2★7/4 |**ENTER**|
6.5

Du kan ændre datamatens prioriteter ved at tilføje parenteser, så vil datamaten foretage udregningerne inden i parenteserne før de andre udregninger. Det kan du se ved at indsætte en parentes i ovenstående regneeksempel:

?(3 + 7-2)★7/4 |**ENTER**|
og sandelig, datamaten regner som et menneske. Den siger, at resultatet er 14.

EKSPONENTIEL NOTATION

Hvis du vil bruge meget store eller meget små tal i en beregning, er det somme tider nyttigt at bruge eksponentiel notation. Bogstavet e bruges som eksponent for tallet 10. Du kan bruge stort eller lille e efter behag. Fx er 300 det samme som 3 x 10 i anden. I eksponentiel notation bliver dette 3E2. På samme måde er 0.03 det samme som 3 gange 10 i minus anden. I eksponentiel notation bliver det til 3E-2. Prøv de følgende eksempler.

1. 30x10

Du kan indtaste:
?3Ø★1Ø |**ENTER**|
3ØØ

men du kan også indtaste:
?3e1★1e1 |**ENTER**|
3ØØ

2. 3000X1000

Indskriv:
?3e3★1e3 |**ENTER**|
3ØØØØØ

3.3000x0.001

Indskriv:
?3e3★1e-3 |**ENTER**|
3

Grundlaget 3:
Grafik, funktions-tilstande og lyd

AMSTRAD CPC464-farvedatamaten har tre funktionstilstande for skærmen. De kaldes *modes* - altså: Mode 0, Mode 1 og Mode 2.

Når du tænder for datamaten, er den automatisk i Mode 1.

For at forstå de forskellige modes må du tænde for datamaten og trykke på 1-tasten. Hold på den, indtil du har to hele linjer 1-taller. Hvis du nu tæller antallet af 1-taller på den første linje, vil du se, at der er 40. Tryk nu |**ENTER**| - så får du en melding om *syntax error,* men bryd dig ikke om det. Det er bare en hurtig måde til at få datamaten tilbage til *ready,* så den er klar til din næste ordre.

Prøv så at indtaste: mode Ø |**ENTER**|.

Nu kan du se, at karakterne på skærmen er større. Hvis du igen holder på 1-tasten, indtil to linjer er fyldt med 1-taller, og tæller efter, vil du nå til tallet 20 på en linje. Tryk igen |**ENTER**|.

Indtast nu: mode 2 |**ENTER**|.

Her er vi ved den smalleste funktionstilstand, og du har nu hele 80 karakterer på en linje. Lad os repetere:

Mode 0 = 20 anslag pr. linje
Mode 1 = 40 anslag pr. linje
Mode 2 = 80 anslag pr. linje

Tryk til slut endnu en gang |**ENTER**|.

FARVER

AMSTRAD CPC464-datamaten har 27 farver at vælge imellem. På en grøn skærm (GT64) vises de som forskellige afskygninger af grønt. Hvis du har købt en GT64, kan du nu anskaffe dig en MP1-netdel, der gør det muligt for dig at benytte dit eget farvefjernsyn som skærm til datamaten.

I Mode 0 kan man på een gang bruge 16 af de 27 mulige farver.
I Mode 1 kan på een gang anvendes 4 af de 27 mulige farver.
I Mode 2 kan man bruge to af de 27 farver på een og samme gang.

Du kan ændre farverne på skærmens kant *(BORDER),* skærmens grundfarve *(PAPER)* og på selve karakteren *(PEN),* og alle disse tre kan ændres uafhængigt af hinanden.

De 27 mulige farver vises i tabel 1, og hver er her forsynet med det nummer, man bruger i forbindelse med kommandoen *INK*.

FARVEKORTET

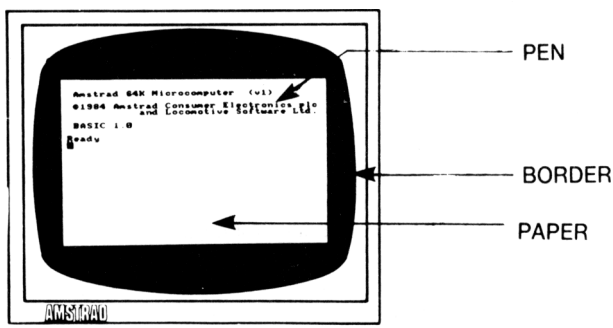
Ink-tallet	Farve/Ink	Ink-tallet	Farve/Ink
0	sort	14	pastelblå
1	blå	15	orange
2	højblå	16	lyserød
3	red	17	pastelmagentarød
4	magentarød	18	højgrøn
5	grålilla	19	søgrøn
6	højrød	20	højcyan(blå)
7	purpur	21	lysegrøn
8	højmagenta	22	pastelgrøn
9	grøn	23	pastelcyan
10	cyan	24	højgul
11	himmelblå	25	pastelgul
12	gul	26	højhvid
13	hvid		

Tabel 1: INK-tallene og farverne (de farver, der har betegnelsen høj som forstavelse, er de lysende klare versioner af dem).

Som tidligere fortalt, er datamaten i Mode 1, når den tændes. For at komme tilbage til Mode 1 skal du blot indtaste dette og trykke |**ENTER**|

Når datamaten tændes, er farverne således fordelt:

Border: Farve nummer 1 (blå)
Paper (selve skærmen): Farve nummer 1 (blå)
Pen (karakteren): Farve nummer 24 (højgul)



BORDER er området uden om *PAPER* (papiret, man skriver på), og *PEN* er farven på karakteren (et tal, et tegn eller et bogstav). Du kan fylde *INK* (blæk) i en *PEN*, og ligesom du kan ændre farven på blækket i en rigtig pen, sådan kan du også ændre farven på de karakterer, der står på skærmen. På samme vis med papiret.

Prøv at skifte rammens farve med ordren:

border Ø |**ENTER**|

Nu skifter BORDERs farve fra blå til sort. I tabel 1 kan du se, at farven 0 er sort. Du kan skifte BORDER-farven til enhver af disse farver ved at indtaste *border* og nummeret på den ønskede farve.

Indtast nu:

cls |**ENTER**|

for at viske skærmen ren, så du kan beordre en ny farve på *PAPER*. Herefter indtaster du:

paper 2 |**ENTER**|

så skifter skærmfarven bag ordet *ready* til højcyan. Indtast nu igen:

cls |**ENTER**|

så du kan få en ny farve på *PAPER*

For at se farven på *PEN* skifte, kan du indtaste:

pen 3 |**ENTER**|

så ser du *PEN* ændre farve, idet ordet *Ready* nu står med højrødt. Indtast herefter:

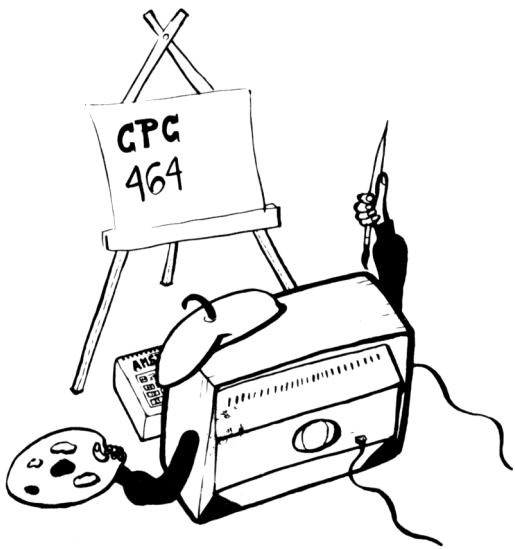
cls |**ENTER**|

Du vil lettere kunne forstå, hvad der sker, hvis du kigger lidt i tabel 2. Når datamaten tændes, er tallet 0 for *PAPER*. Hvis du ser i tabel 2, første spalte, vil du se tallet 0 udfor *PEN,* og følger du nu samme linje i Model-kolonnen, ser du farvetallet 1. Et blik tilbage i tabel 1 vil vise dig, at tallet 1 svarer til blå, d.v.s. den farve, datamaten har på *PAPER,* når man tænder for den.

Vi har lige skiftet *PAPER*-farven til 2. Kigger du nu på 2 udfor *PAPER* i venstre spalte i tabel 2, vil du se, at det i Mode 1 svarer til farvetallet 20. Et kig i tabel 1 viser, at farven 20 er højcyan.

	Inkfarve		
Paper/Pen No.	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	blinkende 1,24	20	1
15	blinkende 16,11	6	24

Tabel 2: PAPER/PEN/MODE/INK-henvisninger



Den *PEN,* der anvendes, når datamaten tændes, er nr. 1. I tabel 2 ser du i Mode 1-kolonnen, at *PEN* nr. 1 er farve nr. 24. Går du nu over til tabel 1, ser du, at farve nr. 24 er højgul - og altså farven på *PEN,* når datamaten tændes.

Vi har netop ændret *PEN* til nr. 3. I tabel 2 kan du se, at *PEN* nr. 3 er farve nr. 6 i Mode 1, og i tabel 1 ser du, at farve 6 er højrød.

I øjeblikket bruger vi *PAPER* nr. 2 og *PEN* nr. 3. Vi vil nu lave om på de farver. Det gør vi med ordren *INK.* Denne ordre har to tal. Det første er nummeret på den *PEN* eller det *PAPER,* du vil ændre farve på. Det andet tal er farvenummeret, du ønsker farven ændret til. Find farvens nummer i tabel 1. Vi vil som et eksempel vise dig, hvordan du ændrer farven på *PAPER 2* til sort og farven på *PEN 3* til højhvidt. I tabel 1 kan du se, at farve nr. 0 er sort og nr. 26 er højhvidt. Prøv nu at indtaste:

ink 2,Ø |**ENTER**|

(som forklaret er 2 den nuværende farve på *PAPER* og 0 er sort)

Indtast derefter:

ink 3,26 |**ENTER**|

(3 er den nuværende *PEN* og 26 er højhvidt)

Reset nu din datamat ved at trykke |**CTRL**| |**SHIFT**| og |**ESC**|.

Som tidligere forklaret er farven på *PAPER* nr. 0 og *PEN* nr. 1, når datamaten tændes, eller når du laver reset. D.v.s., at papiret er blåt (1) og pennen er højgul (24). Hvis du skulle indtaste disse farver, ville du skrive *ink Ø,1* for *PAPER* og *ink 1,24* for *PEN.* For at ændre disse farver, så du får hvide karakterer (PEN) på en sort baggrund (PAPER), skal du indtaste:

ink Ø,Ø |**ENTER**|

og derefter indtaste:

ink 1,26 |**ENTER**|

BLINKENDE FARVER

Du kan få farverne til at blinke, d.v.s. skifte mellem een farve og en anden. Det gøres ved at give *PEN* et ekstra farvenummer. Hvis du fx vil have en karakter på skærmen til at blinke mellem højhvidt og højrødt, skal du indtaste:

ink 1,26,6 |**ENTER**|

I dette tilfælde er 1 nummeret på *PEN,* mens 26 er højhvidt og 6 er den farve, der skiftes til, højrødt.

Man kan også få farven på *PAPER* bag karaktererne til at blinke mellem to farver. Dette gøres ved at tilføje et ekstra tal til ordren *INK* for *PAPER.* Hvis du fx vil se papiret blinke mellem grøn og højgul bagved karaktererne, så indtaster du blot:

ink Ø;9,24 |**ENTER**|

I dette tilfælde er 0 papirets farvetal, mens 9 er grøn og 24 er den farve, der skal skiftes til, højgul. Reset nu din datamat med |**CTRL**| |**SHIFT**| |**ESC**|.

Bemærk, at i Mode 0 er to af farverne til *PEN* (nr. 14 og 15) sammen med to af farverne til *PAPER* (også 14 og 15) automatisk blinkende. Det er med andre ord ikke nødvendigt at tilføje et ekstra tal til ordren *INK*. Prøv at indtaste følgende:

mode Ø |**ENTER**|
pen 15 |**ENTER**|

og på skærmen vil du se ordet *Ready* blinke mellem himmelblåt og lyserødt. Prøv nu at indtaste:

paper 14 |**ENTER**|
cls |**ENTER**|

så ser du ikke blot ordet *Ready* blinke mellem himmelblåt og lyserødt, men baggrunden *PAPER* blinker samtidig mellem gult og blåt.

Man kan godt ændre disse automatiske blinkfarver ved at indtaste en ny *INK*-ordre til *PEN* eller *PAPER*. Hvis du fx vil ændre blinkfarverne for *PEN* til sort og højhvidt, så indtaster du:

ink 15,Ø,26 |**ENTER**|

I dette tilfælde er pennens nr. 15, mens 0 er sort, og 26 er højhvid og dermed farven, der skal skiftes til.

Sluttelig vil vi vise, hvordan man får *BORDER* til på samme vis at skifte mellem to farver. Også det sker ved at tilføje et ekstra farvetal til ordren. Indtast fx:

border 6,9 |**ENTER**|

så vil du se *BORDER* blinke mellem højrød og grøn. Reset nu datamaten som tidligere vist

DEMONSTRATIONSPROGRAM

Hvis du ønsker en yderligere demonstration af, hvilke farver der er til din rådighed, kan du indtaste det følgende program og derefter give ordren *run*. Vi har også lagt lidt lyd ind i programmet. Dette forklares i et senere afsnit.

```
10 MODE 0: INK 0,2: INK 1,24: PAPER 0
20 PEN 1: FOR B=0 TO 26: BORDER B
30 LOCATE 3,12:PRINT "BORDER FARVE ";B
40 SOUND 4,(40-B)
50 FOR T=1 TO 600:NEXT T:NEXT B:CLS
60 FOR P=0 TO 15:PAPER P:PEN 5:PRINT "PAPER";
   P:PRINT
70 FOR N=0 TO 15:PEN N:PRINT "PEN";N
80 SOUND 1,(N*20+P)
90 FOR T=1 TO 100:NEXT T:NEXT N
100 FOR T=1 TO 1000:NEXT T:CLS:NEXT P
110 CLS:PAPER 0:PEN 1:LOCATE 7,12:PRINT
    "THE END":FOR T=1 TO 2000:NEXT T
```

```
120 MODE 1:BORDER 1:INK 0,1:INK 1,24:PAPER 0:
    PEN 1

RUN
```

GRAFIK

Nu skulle du efterhånden have lært at trykke |**ENTER**| efter hver programlinje, så nu vil vi holde op med at fortælle dig det. Vi går ud fra, at du selv husker det - ellers skal datamaten nok minde dig om, at du har gjort et eller andet galt. På samme vis vil vi heller ikke mer fortælle, hvorledes du laver reset på din datamat. Når vi siger: reset datamaten, går vi ud fra, at du ved, at du skal trykke tastkombinationen: |**CTRL**| |**SHIFT**| |**ESC**|.

Der findes en del karakterer (bogstaver, tal eller tegn) i datamatens arbejdslager. For at få udskrevet et af dem på skærmen må man bruge nøgleordet *chr$( ),* idet man i parentesen sætter tallet for den karakter, man ønsker udskrevet. Tallet går fra 32 til 255.

Reset datamaten og indtast:

```
print chr$(25Ø)
```

På skærmen ser du nu karakteren 250, som er en mand, der spadserer til højre. Hvis du nu vil se alle datamatens karakterer og deres numre på skærmen, skal du indtaste følgende program:

```
1Ø for n=32 to 255: print n;chr$(n);
2Ø next n
run
```

Hvis du vil slå en eller flere karakterer op til brug for et program, du laver, så har du hele sættet i Appendiks III bagest i bogen.

LOCATE

Denne ordre, der betyder *anbring, sæt eller placer,* bruges til at flytte karaktermarkøren til et bestemt sted på skærmen. Karaktermarkøren, der viser det sted, hvor du nu vil have skrevet noget, starter normalt i skærmens øverste venstre hjørne, der bærer betegnelsen 1,1 i det system, man kalder xy-koordinaterne. X er den vandrette position og Y er den lodrette position. I Mode 1 rummer skærmbilledet 25 linjer, hver på 40 karakterer (eller anslag). Hvis du ved xy-koordinatsystemet ville anbringe en karakter midt på første linje i Mode 1, skulle du bruge koordinaterne 20,1. For at se dette skal du indtaste:

```
mode 1

1Ø locate 2Ø,1
2Ø print chr$ (25Ø)
run
```

For at bevise, at det virkelig er øverste linje, kan du indtaste:

```
border Ø
```

Nu vil *BORDER* blive sort, og du vil se den lille mand på midten af øverste linje.

I Mode 0 er der fortsat 25 linjer, men kun 20 anslag pr. linje. Hvis du nu indtaster:

```
mode Ø
run
```

vil du se den lille mand dukke op i skærmens øverste højre hjørne. Det skyldes, at x-koordinaten 20 er det sidste anslag på linjen i Mode 0.

I Mode 2 er der stadigvæk 25 linjer, men 80 anslag på hver. Hvis du anvender det samme program, vil du sandsynligvis kunne gætte, hvor manden vil dukke op. Prøv at indtaste:

```
mode 2
run
```

Gå nu tilbage til Mode 1 ved blot at indtaste: *mode 1*

Nu kan du selv eksperimentere lidt med *locate* og *chr$( )*, idet du varierer tallene i de to udtryk, indtil du har lært at placere karakterer et hvilket som helst sted på skærmen. Fx vil du kunne tegne en pil midt på skærmen, hvis du indtaster:

```
locate 2Ø,12:print chr$ (24Ø)
```

Det skyldes, at 20 er den horisontale x-koordinat (i rækken 1 til 40)
at 12 er den lodrette y-koordinat (i rækken 1 til 25)
og at 240 er karaktersymbolet for pil (i rækken 32 til 255)

Hvis du vil have karaktersymbolet 250 skrevet igen og igen over hele skærmen, skal du indtaste følgende program:

```
5 cls
1Ø for x = 1 to 39
2Ø locate x,2Ø
4Ø print chr$ (250)
5Ø next x
6Ø goto 5
run
```

Tryk |**ESC**| to gange for at bryde programmet.

For at fjerne den foregående karakter fra skærmen, før den næste karakter vises, skal du indtaste:

```
4Ø print " "; chr$(25Ø)
```

(Denne nye linje 40 erstatter automatisk den linje, der tidligere blev indtastet som linje 40).

Indtast nu

```
run
```

For at forbedre karakterens bevægelse hen over skærmen, kan du tilføje denne linje:

```
3Ø call &bd19
```

Dette program kan forbedres yderligere ved at indføje forsinkende løkker og ved at bruge andre karakterer. Indtast nu:

list

Indsæt nu følgende linjer i programmet:

```
60 FOR N=1 TO 300:NEXT N
65 FOR X=39 TO 1 STEP -1
70 LOCATE X,20
75 CALL &BD19
80 PRINT CHR$(251)
85 NEXT X
90 FOR N=1 TO 300:NEXT N
95 GOTO 10
RUN
```

Prøv dette lille interessante program. Vi har tilføjet nogle yderligere ordrer, som vil blive forklaret senere i bogen. Lige for øjeblikket kan du nøjes med at indtaste:

```
10 MODE 1
20 LOCATE 21,14:PRINT CHR$(244)
30 TAG
40 FOR X=0 TO 624 STEP 2
50 MOVER -16,0
60 IF X<308 OR X>340 THEN Y=196:GOTO 90
70 IF X<324 THEN Y=X-104:GOTO 85
80 Y=536-X
85 SOUND 1,0,20,7
90 MOVE OX,OY:PRINT " ";:OX=X:OY=Y
100 MOVE X,Y
110 IF (X MOD 4)=0 THEN PRINT CHR$(250);
    ELSE PRINT CHR$(251)
120 FOR N=1 TO 4: CALL &BD19:NEXT N
130 NEXT X
140 TAGOFF
150 GOTO 20
RUN
```

PLOT

Mens locate-ordren skulle placeres karaktermarkøren, skal ordren *PLOT* placere grafikmarkøren, der modsat karaktermarkøren ikke er synlig. Ved grafik bruger man pixelkoordinater - en pixel eller et billedpunkt er en meget lille bestanddel af skærmbilledet. Der er derfor plads til hele 640 vandrette og 200 lodrette pixel på et skærmbillede. Modsat ved *locate* ændrer xy-koordinaterne sig ikke efter, om det er Mode 0, 1 eller 2, du bruger. xy-koordinaterne ved grafik starter i skærmens nederste venstre hjørne, hvor vi har punktet 0,0.

Reset datamaten og gør et lille beskedent grafikforsøg. Indtast:

plot 32Ø,2ØØ

Du fik en lille prik midt på skærmen. Prøv nu at ændre Mode ved at indtaste:

mode Ø
plot 32Ø,2ØØ

Nu ser du, at prikken fortsat er i skærmens centrum, men er blevet lidt større. Skift nu Mode igen ved at indtaste:

mode 2
plot 32Ø,2ØØ

Prikken er fortsat i centrum, men er nu blevet meget mindre. For at vænne dig til denne ordre bør du nu *plotte* prikker mange steder på skærmen, mens du skifter Mode. Når du er træt af det, går du tilbage til Mode 1 og visker skærmen ren ved at indtaste:

mode 1

DRAW

Ordren *DRAW* (tegn) bruges til at tegne en linje fra grafikmarkørens nuværende position. For at se dette i praksis vil vi med det efterfølgende program tegne et rektangel. Vi starter med at anbringe grafikmarkøren ved hjælp af en plot-ordre. Så trækker vi med *DRAW* en linje fra grafikmarkørens position opad mod øverste venstre hjørne, derfra videre til højre hjørne o.s.v.

Reset datamaten og indtast:

```
5 cls
1Ø plot 1Ø,1Ø
2Ø draw 1Ø,39Ø
3Ø draw 63Ø,39Ø
4Ø draw 63Ø,1Ø
5Ø draw 1Ø,1Ø
6Ø goto 1Ø
run
```

Tryk to gange på |**ESC**| for at slippe ud af programmet. Tilføj nu følgende linjer, så du får tegnet et nyt rektangel inden i det første:

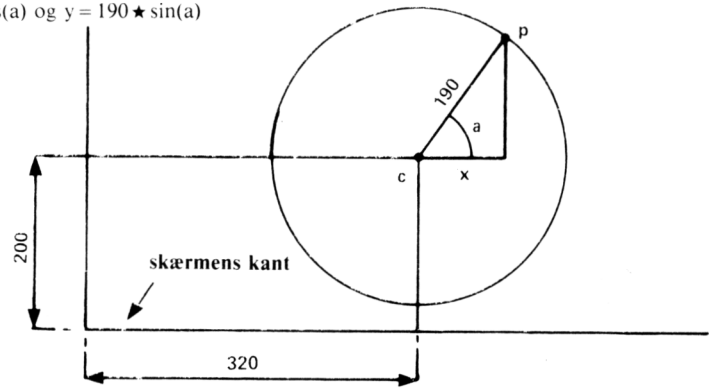
```
6Ø plot 2Ø,2Ø
7Ø draw 2Ø,38Ø
8Ø draw 62Ø,38Ø
9Ø draw 62Ø,2Ø
1ØØ draw 2Ø,2Ø
2ØØ goto 1Ø
run
```

Tryk to gange |**ESC**| for at undslippe programmet.

CIRKLER

Cirkler kan enten laves med *PLOT* eller *DRAW*. En af metoderne er at plotte xy-koordinaterne for hvert eneste punkt langs cirklens omkreds. Hvis du ser på tegningen her, kan du se, at punktet p på cirklen kan plottes med xy-koordinaterne, sådan:

x = 190 ★ cos(a) og y = 190 ★ sin(a)



NEW

Vi er nu begyndt at bruge ordren *new,* før vi indtaster et program. Denne ordre fortæller datamaten, at den skal rydde sit arbejdslager fuldstændigt. Det svarer omtrent til reset af datamaten, men ved *new* bliver skærmen ikke visket ren, kun arbejdslageret. Dette er nyttigt, idet du måske fortsat ønsker at se det gamle program på skærmen, mens du skriver det nye (og evt. bedre).

Tilbage til cirklerne. I det foregående program *plottede* vi punkterne på en cirkel med nederste venstre skærmhjørne som udgangspunkt. Hvis vi nu ønsker at anbringe en cirkel i skærmens centrum, må vi først plotte cirklens centrum med koordinaterne 320,200 og derefter anbringe alle punkterne i forhold til dette centrum ved at tilføje centrumspositionens koordinater. Et program, der skulle tegne en cirkel, skulle så se således ud:

```
new
5 cls
1Ø for a = 1 to 36Ø
15 deg
2Ø plot 32Ø,2ØØ
3Ø plot 32Ø + 19Ø ★ cos(a),2ØØ + 19Ø ★ sin(a)
4Ø next
run
```

Cirklens radius kan mindskes ved at mindske tallet 19Ø, der drejer sig om antallet af pixel. Hvis du vil se en anderledes cirkel, der er tegnet uden gradsordren *DEG* , så fjern linje 15 ved simpelthen at indtaste:15

Hvis du vil se en solid cirkel tegnet med linjer fra centrum, så rediger linje 30, idet du erstatter ordet *plot* med ordet *draw*. Prøv dette med og uden linje 15.

Bemærk, at i dette program skriver vi *next* i stedet for *next a*. Det kan sagtens lade sig gøre, hvis der kun er eet *for,* idet datamaten så selv finder ud af, hvad det er for et *for,* dette *next* henviser til i den såkaldte for-next-løkke. Men i et program, hvor der er utallige for-next-løkker, kan man forvirre datamaten totalt, hvis man glemmer at hæfte det rigtige navn efter next.

ORIGIN

I det foregående program brugte vi plot-ordren til at plotte centrum i en cirkel og tilføjede xy-koordinaterne fra denne centrumsposition. I stedet for at tilføje disse koordinater kan vi bruge ordren *ORIGIN* (d.v.s. oprindelse). Denne ordre vil fastlægge cirklens centrum og derefter fastsætte xy-koordinaterne til alle omkredsens punkter (i skridt på hver een grad) udfra dette oprindelige punkt. For at se dette indtaster du:

```
5 CLS
10 FOR A = 1 TO 360
15 DEG
20 ORIGIN 320,200
30 PLOT 190★COS(A),190★SIN(A)
40 NEXT
RUN
```

Her kan du igen ændre linjerne 15 og 30 ved at fjerne ordren *deg*, eller du kan tegne en solid cirkel ved at fjerne *draw*. Hvis du vil tegne fire mindre cirkler på skærmen, kan du indtaste følgende program:

```
5 CLS
10 FOR A = 1 TO 360
15 DEG
20 ORIGIN 196,282
30 PLOT 50★COS(A),50★SIN(A)
40 ORIGIN 442,282
50 PLOT 50★COS(A),50★SIN(A)
60 ORIGIN 196,116
70 PLOT 50★COS(A),50★SIN(A)
80 ORIGIN 442,116
90 PLOT 50★COS(A),50★SIN(A)
100 NEXT
RUN
```

Her kan du igen fjerne linje 15 og ændre linjerne 30,50,70 og 90 ved at bruge ordren *DRAW*.

GOSUB RETURN

Hvis du i et program har en række instrukser, der skal udføres et antal gange, kan du indsætte dem som *subrutiner,* som programmet kan hente igen og igen ved hjælp af ordren *gosub* og et linjenummer

Afslutningen på en *gosub*-rutine markeres med instruktionen *return*. Når datamaten møder denne ordre, vil den søge tilbage til den programordre, der fulgte umiddelbart efter gosub-ordren.

I det foregående program brugte vi ordren *plot 5Ø★cos(a),5Ø★sin(a)* fire gange. Det kunne vi have klaret med en subrutine, det blev sat i gang med en gosub-ordre. Prøv at indtaste følgende:

```
5 CLS
10 FOR A = 1 TO 360
15 DEG
20 ORIGIN 196,282
30 GOSUB 120
40 ORIGIN 442,282
50 GOSUB 120
60 ORIGIN 196,116
70 GOSUB 120
80 ORIGIN 442,116
90 GOSUB 120
100 NEXT
110 END
120 PLOT 50*COS(A),50*SIN(A)
130 RETURN
RUN
```

Bemærk, at instruksen *end* bruges i linje 110. Hvis den ikke var der, ville datamaten af sig selv efter linje 100 fortsætte med linje 120, som der jo kun var brug for, hvis der var givet en gosub-ordre.

Som afslutning på dette afsnit har vi et lille program, der indeholder en del programmeringsordrer og nøgleord, som du nu burde forstå:

```
NEW
10 MODE 0:BORDER 6:PAPER 0:INK 0,0
20 GOSUB 160:FOR X = 1 TO 19:LOCATE X,3
30 PEN 15:PRINT" ":CHR$(238)
40 FOR T = 1 TO 50:NEXT T:SOUND 2,(X + 100)
50 NEXT X:GOSUB 160:FOR B = 3 TO 22
60 LOCATE 20,B:PEN 7:PRINT CHR$(252)
70 CLS:GOSUB 160:NEXT B
80 SOUND 2,0,100,15,0,0,1
90 GOSUB 160:BORDER 16,24:LOCATE 20,25
100 PEN 14:PRINT CHR$(253);
110 FOR T = 1 TO 1000:NEXT T
120 BORDER 6:GOSUB 160:FOR F = 3 TO 24
130 LOCATE 10,(25-F):PEN 2
140 PRINT CHR$(144):CLS:GOSUB 160
150 SOUND 7,(100-F),5:NEXT F:GOTO 10
160 LOCATE 10,25:PEN 12
170 PRINT CHR$(239):RETURN
RUN
```

LYD

Lydeffekter skabes af en højttaler, der er indbygget i datamaten. Hvis du anvender MP1-netdelen og et almindeligt fjernsyn, bør du skrue helt ned for tv-lyden.

På datamaten kan lydniveauet reguleres ved hjælp af knappen mærket *VOLUME* helt ude på højre endeflade. Du kan også overføre lyden til en forstærker, fx et stereoanlæg ved hjælp af I/O-kontakten i venstre side af datamatens bagflade. På denne måde kan du lytte til de lyde, datamaten skaber, i stereo enten gennem højttalere eller hovedtelefoner.

Ordet envelope bruges i det følgende for at beskrive, hvad der sker med en lyd som funktion af tiden. Fx er styrke-envelopen en beskrivelse af, hvordan tonens styrke forløber. Dette kan fx være hurtigt op på maksimal styrke, hurtigt ned på en mellemstyrke og derefter langsom udklingen. Ordren *SOUND* har syv parametre. Kun de to første skal altid anvendes. De øvrige fem er det frivilligt, om du vil bruge. Ordren indskrives som:

kanalstatus, toneperiode, varighed, styrke, styrkens envelope, tonens envelope og støjperiode.

I de følgende eksempler vil vi bruge 1 som kanalstatus - med andre ord dens referencenummer.

TONEPERIODE

Hvis du kigger i Appendiks VII, vil du se, at det midterste c har en toneperiode på 478. Indtast nu:

```
new
1Ø sound 1,478
run
```

så vil du høre en kort tone, der er det midterste c gengivet i 0,2 sekunder.

VARIGHED

Hvis du ikke fastlægger varigheden af en lyd, vil den vare i 0,2 sekunder. Varighedens enheder er på 0,01 sekund. Hvis du vil have en tone, der varer i 1 sekund, skal du altså bruge tallet 100, og hvis den skal vare i to sekunder tallet 200. Prøv at indtaste:

```
1Ø sound 1,478,2ØØ
run
```

Nu vil du høre det midterste c i to sekunder.

STYRKE

Dette tal fastlægger startstyrken på en tone. Tallet er i rækken fra 0 til 7. Hvis du imidlertid fastlægger en styrkeenvelope, udvides rækken til at omfatte 0 til 15. Hvis du ikke angiver noget tal, går datamaten ud fra, at du ønsker styrke 4. Prøv at indtaste:

```
1Ø sound 1,478,2ØØ,3
run
```

Nu vil du høre en meget kraftigere lyd.

STYRKEENVELOPE

Ordren for styrkeenvelope er *env*. Denne har normalt fire parametre. De sidste tre parametre dukker op i enhvert af de fem envelopeafsnit, der står til rådighed. Vi bruger kun en af dem her. De fire parametre ved *env* er: envelopenummeret, antallet af skridt, størrelsen af skridtene og skridttiden.

ENVELOPENUMMERET

Dette tal gives til en bestemt envelope, således at denne kan bruges i en lydkommando. Tallet kan vælges mellem 0 og 15.

ANTALLET AF SKRIDT

Dette bruges i sammenhæng med skridttiden. Fx har du måske lyst til at bruge 10 skridt af hver 1 sekunds varighed. I dette tilfælde er antallet af skridt 10. Skridtenes antal kan vælges mellem 0 og 127.

SKRIDTENES STØRRELSE

Hvert skridt kan variere i størrelse fra niveau 0 til 15 i forhold til det sidste skridt. De 15 styrkeniveauer er de samme som i lydkommandoen. Men skridtene kan reguleres fra -128 til + 127, så du kan ikke blot variere udstrækningen op og ned på klar og selvfølgelig vis, men også få nogle ejendommelige lydeffekter frem ved at bruge tal, der rækker ud over 15.

SKRIDTTIDEN

Dette tal fastlægger tiden mellem skridtene i enheder på 0,01 sekunder (d.v.s. hundrededele af sekunder). Rækkevidden går fra 0 til 255. Den længst mulige tid mellem to skridt er derfor 2,55 sekunder.

Du kan eksperimentere med styrkeenvelopen ved at indtaste følgende program:

```
5 env 1,1Ø,1,1ØØ
1Ø sound 1,284,1ØØØ,1,1
run
```

Linje 10 fastlægger en lyd, der skal have en toneperiode på 284 (det internationale a), skal vare i 10 sekunder med en startstyrke på 1 og skal bruge styrkeenvelope nr. 1, der er vist i linje 5, og består af 10 skridt, som øger styrken for hvert skridt med 1 pr. sekund (100x0,01 sekunder).

Prøv nu at ændre linje 5 på hver af følgende måder og *run* hver gang for at høre virkningen af ændringer i envelopen.

```
5 env 1,1ØØ,1,1Ø
5 env 1,1ØØ,2,1Ø
5 env 1,1ØØ,4,1Ø
5 env 1,5Ø,2Ø,2Ø
5 env 1,5Ø,2,2Ø
5 env 1,5Ø,15,3Ø
```

og prøv så til slut dette:

```
5 env 1,5Ø,2,1Ø
```

Du vil bemærke, at halvvejs gennem lyden forbliver lydniveauet konstant. Det skyldes, at antallet af skridt var 50, og at tiden mellem hvert skridt var 0,1 sekund. Derfor var tidsrummet, hvori lydens styrke varierede, kun på fem sekunder, men varigheden af lyden i ordren *lyd* i linje 10 var 10 sekunder (tallet 1000).

Prøv selv at eksperimentere for at se, hvilke former for lyd du kan frembringe.

TONENS ENVELOPE

Ordren for tonens envelope er *ent*.

Den har normalt fire parametre. De sidste tre parametre dukker op i enhver af de 5 envelopeafsnit, der er til rådighed. Vi bruger kun en af disse her. Yderligere forklaring fås i kapitel 6. Her er de fire parametre: Envelopenummer, antal skridt, skridtets toneperiode og skridttid.

ENVELOPE NUMBER

Dette er nummeret, der gives til en envelope, så den kan bruges i en lydkommando. Du kan bruge numre mellem 0 og 15.

SKRIDT ANTAL

Dette bruges sammen med skridttiden. Fx har du måske lyst til at bruge 10 skridt af hver 1 sekunds længde. Du kan bruge mellem 0 og 239 skridt.

SKRIDTETS TONEPERIODE

Toneperioden for hvert skridt kan variere mellem -128 og + 127. Negative skridt øger tonefrekvensen (gør tonerne højere). Den korteste toneperiode er 0. Det må man huske, når man opbygger en toneenvelope. Toneperiodernes fulde omfang vises i Appendiks VII.

SKRIDTTID

Dette tal fastlægger tiden mellem skridt i enheder på 0,01 sekund (d.v.s. hundrededele af sekunder). Tallet for skridttiden kan vælges mellem 0 og 255. Den længst mulige tid mellem skridtene er derfor 2,55 sekunder.

For at eksperimentere med toneenvelope kan du indtaste dette program:

```
5 ent 1,1ØØ,2,2
1Ø sound 1,284,2ØØ.15,Ø,1
run
```

Linje 10 fastsætter en lyd med en toneperiode på 284 (det internationale a), der skal vare i to sekunder med en startstyrke på 15 (maksimum), uden styrkeenvelope (vist ved 0) og med toneenvelope nummer 1.

Linje 5 definerer toneenvelope nr. 1, der består af 100 skridt, som øget toneperioden med 2 (mindsker frekvensen) hver 0,02 sekunder.

Nu kan du ændre linje 5 på hver af følgende måder og hver gang køre programmet for at høre virkningen af ændringer i toneenvelope:

```
5 ent 1,1ØØ,-2,2
5 ent 1,1Ø,4,2Ø
5 ent 1,1Ø.-4,2Ø
```

Udskift nu ordren *sound* og toneenvelopen ved at indtaste:

```
5 ent 1,2,17,7Ø
1Ø sound 1,142,14Ø,15,Ø,1
15 goto 5
run
```

Tryk to gange på |**ESC**| for at bryde programmet. Nu kan du sætte styrke, styrkeenvelope, toneenvelope og sound-ordren sammen for at skabe forskellige lyde. Begynd med at indtaste:

```
new
5 env 1,1ØØ,1,3
1Ø ent 1,1ØØ,5,3
2Ø sound 1,284,3ØØ,1,1,1
run
```

Udskift nu linje 10 ved at indtaste:

```
1Ø ent 1,1ØØ,-2,3
```

Erstat nu alle linjerne med:

```
5 env 1,1ØØ,2,2
1Ø ent 1,1ØØ,-2,2
2Ø sound 1,284,2ØØ,1,1,1
run
```

Prøv selv at lave nogle variationer

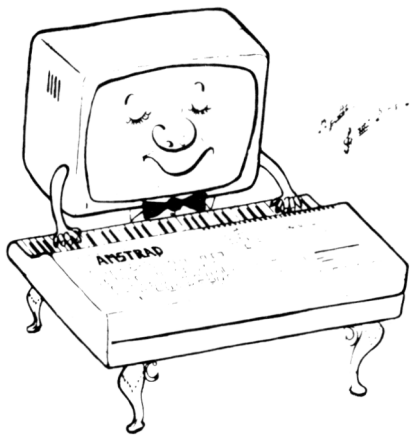
STØJ

Støj kan tilføjes efter en *sound* ordre. Støjen kan vælges fra 1 til 31. Prøv dette ved at føje et støjtal til i enden af en *sound* ordre, mens du fortsat bruger *en*velope ordren.

Erstat linjerne 5 og 20 med:

```
5 env 1,1ØØ,3,1
2Ø sound 1,2ØØ,1ØØ,1,1,1,5
run
```

Prøv her igen at frembringe usædvanlige lyde ved at ændre på styrkeenvelopen og *sound*-ordren - med og uden støj.



1. Opstarten:

For jer, der sprang begyndervejledningen over, kommer her lidt detaljer om, hvordan datamaten opstilles, igangsættes og bruges. Hvis du finder ordvalget lidt forvirrende, bør du gå tilbage til startfasen i indføringskurset.

Emner, der dækkes i dette kapitel:

★ Skik og brug i denne brugervejledning
★ Igangsættelsen
★ Tastaturindøvningen

Uanset, hvor meget du kender til programmering og datamater, bør du sætte dig ind i disse få instrukser om, hvordan man opstiller datamaten. Hvis du lige har åbnet kassen og ikke kan vente med at komme i gang, så vil dette kapitel fortælle dig alt, hvad du har behov for at vide for at kunne tilfredsstille din første nysgerrighed og komme i gang med at bruge BASIC. Dette afsnit er beregnet på brugere, der har et vist kendskab til omgangen med datamater. Begyndere bør starte med vejledningen foran i bogen.

VIGTIGT - du SKAL læse dette:

Terminologien

For at tydeliggøre i teksten, når der henvises til taster på datamatens tastatur, og når der henvises til tekst, der udgør en del af et program, vil vi i denne brugervejledning bruge følgende regler:

|ENTER| : Taster, der ikke har en tilsvarende trykt karakter på skærmen, vises i denne form, omgivet af parenteser.

QWERTYUIOP : Taster, der har en tilsvarende trykt karakter på skærmen, vises i denne form uden parenteser.

10 FOR N = 1 to 1000 : Tekst, der ses på skærmen eller skal indtastes fra tastaturet, vises i denne form eller som udprintninger fra en printer (skriver). Bemærk forskellen mellem nul (Ø) og bogstavet O.

Det forventes, at du afslutter hver programlinje eller hver direkte ordre med at trykke på tasten **|ENTER|**, og dette vil ikke blive gentaget i de programlistninger, der ses i resten af denne vejledning.

Yderligere forventes det, at du indtaster *run* efter hvert program, du har indskrevet. BASIC omdanner alle nøgleord, der indtastes med små bogstaver til STORE BOGSTAVER, når et program LISTes. De eksempler, der vises i denne bog, er med STORE BOGSTAVER, idet det er således, du vil se programmet, hvis du LISTer det. Hvis du indskriver med små bogstaver, vil du lettere være i stand til at få øje på fejl, eftersom de fejlstavede nøgleord ikke vil blive vist af datamaten med STORE BOGSTAVER, når du LISTer programmet.

1.1.1.LUK KASSEN OP!

Farveskærmen

VIGTIGT

Se venligst under **opstillingen** foran i bogen, hvor der er detaljerede instrukser i, hvordan man tilslutter datamaten.

Når datamaten er korrekt forbundet og opstillet som vist i figur 1, så tænd for skærmen og derefter for datamaten, idet du benytter skydeknappen ude i højre sideflade. Efter ca. 30 sekunders opvarmningstid vil skærmen vises dette billede:

```
Amstrad 64K Microcomputer  (v1)
©1984 Amstrad Consumer Electronics plc
        and Locomotive Software Ltd.

BASIC 1.0

Ready
```

Dette er kendt under forskellige navne: Reset-, Godmorgen- eller Opvågningsmeldingen. Det viser, at datamaten er fuldstændigt stillet i starttilstand - det er en tilstand, der indtræffer, når man tænder for datamaten, laver reset via tastaturet, eller når man trykker på tre taster i rækkefølge og på samme tid. De tre taster er:

|**CTRL**| |**SHIFT**| og |**ESC**|. Prøv nu dette, før du har indskrevet noget som helst.

Du kan justere lysstyrken, **BRIGHTNESS**, på skærmen. Farven er forudindstillet fra fabrikken, så hvis du ønsker at ændre den viste farve (guldbogstaver på en blå baggrund), må du gøre det ved hjælp af ordrer i et program. Så må du springe frem til grafikindføringen og BASIC-nøgleordene i Kapitel 8. Hvis du ikke har noget imod at springe lidt frem i forløbet, så får du her et kort program, der vil vise dig en af de bedste og mest læselige kombinationer af farver for en skærmtekst, der anvender højopløsning med 80 anslag (karakterer) pr. linje.

Indskriv:

1Ø MODE 2
2Ø INK 1,Ø
3Ø INK Ø,13
4Ø BORDER 13

...men du kan selvfølgelig også indskrive ovennævnte direkte som enkeltudtryk.

Hvis ovenstående program ikke siger dig noget, eller du ikke kan indtaste det korrekt, så bør du gå tilbage til vort INDFØRINGSKURSUS FOR BEGYNDERE i starten af bogen. Du vil opdage, at 80 karakterers teksttilstanden er klart det bedste skærmbillede, hvis du er ved at udvikle programmer. Hvis du har lyst, kan du gemme ovenstående program på starten af en tom kassette, så du ikke skal indtaste det, hver gang du tænder. SAVE-instrukserne står i Kapitel 2.

PAS PÅ! Skærmens høje lysstyrke gør, at du må være forsigtig med ikke at overbelaste dine øjne, når du sidder nær den. Derfor bør du ikke anvende større lysstyrke end nødvendigt under de herskende belysningsforhold. I det øjeblik du føler, at det påvirker dine øjne, så sluk for skærmen og foretag dig noget andet. Men for helt at afværge det, bør du følge disse råd:

1.Arbejd altid med tilstrækkeligt lys til, at du uden besvær kan læse denne vejledning. Hvis du læser denne vejledning i skæret fra skærmen, må vi på det bestemteste opfordre dig til at finde en anden lyskilde.
2.Brug den mindst mulige skærm-lysstyrke, der tillader dig at se, hvad du laver.
3. Sæt dig så langt væk fra skærmen som muligt.

En bordlampe, der stilles ved siden af skærmen vil hjælpe med at mindske belastningen på dine øjne. Hvis du vel at mærke har placeret den bag skærmen, så den ikke skaber reflekser i denne.

1.1.2 Den grønne skærm

GT64-skærmen har tre kontrolknapper nederst på forsiden. Med disse kan brugeren regulere lysstyrken, kontrasten og lodret hold (**BRIGHTNESS, CONTRAST** og **V**ertical **HOLD** hedder knapperne). Det er muligt, at du må justere HOLD i starten for at hindre, at skærmbilledet ruller, men derefter skulle det ikke mer være nødvendigt. Derimod kan der være behov for af og til at justere lysstyrke og kontrast, eftersom disse ting skal indrette sig efter de skiftende lysforhold i omgivelserne.

Den grønne skærm er en såkaldt monokrom monitor, der varierer lysstyrken i karaktererne for at give kontrast mellem de forskellige dele af skærmbilledet. Startmeldingen på foregående side er derfor den samme som ved farveskærmen, bortset fra, at den vises i lyst grøn på en mere afdæmpet grøn baggrund.

Skønt for megen brug af en monokrom monitor kan føre til overanstrengte øjne, gør skærmbilledets noget *blødere* udseende det lettere at arbejde i længere tid med sådan en skærm end med en farveskærm. En monokrom monitor er selv den dyreste farvemonitor overlegen, hvis man er i stand til at drage fuld nytte af 80 karakterers teksttilstanden, eftersom opløsningsgraden (evnen til at vise små billedenheder tæt ved hinanden, uden at de flyder ud) bestemt er langt bedre på en monokrom skærm.

Juster **BRIGHTNESS**, indtil du har tilstrækkelig lysstyrke på din skærm, uden at karakterne bliver overdrevent udflydende.

Hvis du vil se datamatens 80 karakterers skærmbillede i udfoldelse, så indtast det følgende korte program i din CPC64. Det giver dig valg af skærmbillede:

```
10 REM SAET SKAERM FORMAT
20 FOR N=0 TO 26
30 MODE 2
40 INK 1,N
50 INK 0,(26-N)
55 BORDER N
60 LOCATE 15,12:PRINT "TRYK PAA EN BOGSTAVTAST FOR AT SKIFTE
   SKAERM FORMAT"
70 A$=INKEY$
80 IF A$="" GOTO 70
90 NEXT
100 GOTO 20
RUN
```

Ovenstående illustrerer et andet vigtigt punkt ved skik og brug i denne brugervejledning. Visse programlistninger *løber over* d.v.s. fortsætter på næste linje. Det er vigtigt at bemærke, at når vi viser listningerne, så er de ekstra mellemrum, der står i starten af den nye linje, ikke nødvendige ved programindskrivningen. De står der bare for at give et pænere satsbillede i bogen.

Dette program viser ikke alle de mulige kombinationer af gråtoner (farver), men det giver dig et billede af, hvad der er til rådighed. Hvis du finder en farvekombination, du kan lide, så stands datamaten ved at trykke to gange på tasten |**ESC**|, så vil det resultere i et *Break* i programmet.

Fremover vil bogen ofte henvise til farvemulighederne. Programmer, der viser interessante kombinationer af farver og grafik, giver muligvis helt usynlige resultater på en monokrom skærm, skønt vi har gjort os de største anstrengelser for at frembringe en farveskala, der giver en progressiv skala af tilsvarende gråtoner. Dette beskrives mere udførligt i Kapitel 5.

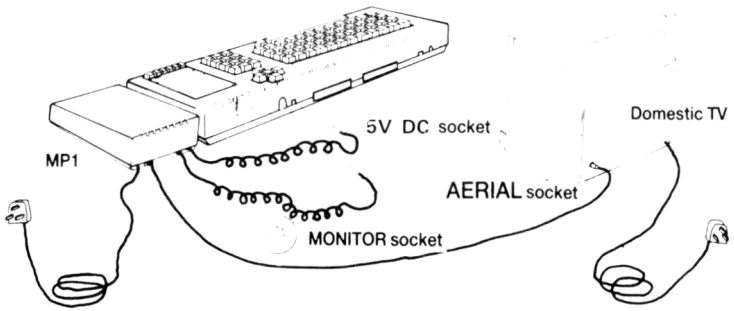
Fordelen ved den monokrome skærm er, at den giver et klarere og mindre trættende skærmbillede, mens man udvikler programmer. Men hvis du nu eller senere bliver bidt af programmeringsdjævlen, så er der nok ikke meget håb om, at du ikke før eller siden lader dig friste af farvemuligheden!

1.1.3 MP1 TV-modulatoren/netdelen

MP1 er et ekstraudstyr, som du kan få lyst til at anskaffe dig, hvis du p.t. anvender AMSTRAD CPC464-datamaten med GT64 grøn skærm. Med MP1 kan du anvende dit eget farvefjernsyn som farveskærm og kan derved til fulde nyde AMSTRAD CPC464-datamatens farvemuligheder.

VIGTIGT

Se venligst under **opstillingen** foran i bogen, hvor der er detaljerede instrukser i, hvordan man tilslutter datamaten.



*Se på side G1.4, hvorledes du forbinder MP1 med datamaten og din TV-antenne, og hvor*der er givet i et program.

Eftersom signalet fra din datamat passerer gennem en del omformning undervejs til fjernsynet (det moduleres og demoduleres), kan der optræde nogen forvrængning. Derfor kan resultatet ved brugen af et tv som skærm aldrig blive helt så godt som det, du får ved brugen af en direkte tilsluttet skærm (monitor). Hvis dit fjernsyns kvalitet ikke er den for en AMSTRAD CPC464-datamat ønsket kan du fx opleve, at en 80 karakterers tekst (Mode 2) ikke giver et fuldstændigt krystalklart resultat. I så tilfælde må du nøjes med Mode 1, når du skal indskrive tekst.

1.2 De første skridt

Du er nu *tilsluttet* og klar til at anvende datamaten. Den venter blot på dit input:

Opstartmeldingen, d.v.s. den velkomsttekst, der står på skærmen, når du tænder, er den eneste indbyggede tekst i AMSTRAD CPC464-datamaten. Al anden tekst skal du generere gennem instrukser, du indkoder i datamaten.

Hvis du allerede er bekendt med BASIC-programmeringssproget, har du sikkert allerede indskrevet et kort program for at gøre dig bekendt med datamaten. AMSTRAD BASIC er velbekendt på mange måder, og for at få sat dig i gang, vil vi vise dig et kort program, du kan indtaste for at se alle de karakterer, der er *indbygget* i datamaten. Det er det såkaldte karaktersæt - en vending, der beskriver den fuldstændige samling af medfødte bogstaver, tal, tegn og andre former for symboler, der kan udskrives på skærmen ved indtastninger.

Nogle af de karakterer, du ser, kan ikke fås direkte ved et tasttryk, men kan kun fremkaldes ved at indskrive ordren *PRINT CHR$ ( ),* hvor karakterens nummer (kodetal) indskrives i parentesen. Dette beskrives senere i denne vejledning

Årsagen til denne *særhed* er, at hver enhed i karaktersættet er gemt i datamaten som den dataenhed, der hedder en byte. Som du vil se, når du arbejder dig igennem Appendiks II, har en byte 256 forskellige værdimæssige kombinationsmuligheder. Men eftersom datamaten skal og må bruge mindst een hel byte for hver karakter, den gemmer på, (idet det jo faktisk er den mindste værdienhed, CPC464 kan genkende) kan vi jo lige så godt bruge alle 256 muligheder i stedet for at nøjes med de ca. 100 standardkarakterer, som en skrivemaskine kan klare.

Nogle af karaktererne er international standard. De kaldes ASCII-karaktererne, et datamatudtryk, der er en forkortelse for American Standard Code for Information Interchange (eller på dansk: Amerikansk Standard Kode for Informations Udveksling).

Det er først og fremmest et system, der skal sikre, at data, der sendes fra en fjernskriver eller datamat til en anden, også bliver forstået af modtageren. Her har du faktisk det eneste helt fælles ved datamater, så vi råder dig kraftigt til at sætte dig ind i alle sider af ASCII, som du vil finde i Appendiks III. Her finder du også CPC464s egne ekstra karakterer og deres kodetal.

Nogle af de øvrige karakterer, der ikke direkte kan printes ved et tasttryk, kan du få frem ved en kombination af kontroltasten |**CTRL**| og andre taster. Men det skal du ikke tænke for meget på nu, for så længe du ikke fuldt ud forstår kontroltastens funktion, kan du gøre mere skade end gavn ved at bruge den tilfældigt.

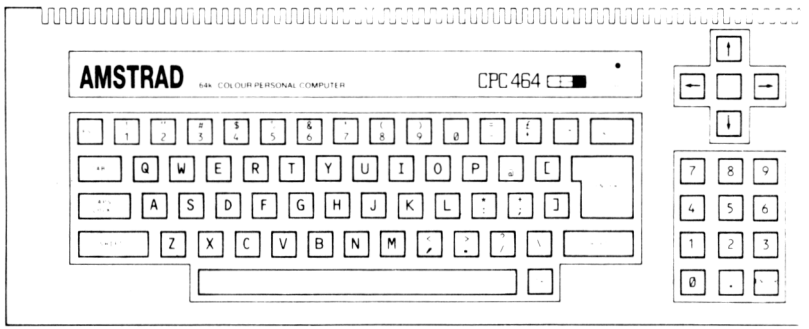
1.2.2

For at se nøjagtigt, hvorledes disse karakterer ser ud på skærmen, skal du indtaste det følgende program, der dels vil tilfredsstille din nysgerrighed og dels vise dig lidt om CPC464. Programmet skal også hjælpe med at overbevise dig om, hvor let programmeringens kunst egentlig er, og det skal også vise dig, at blot du kan fremkalde velkomstbilledet, når du tænder for datamaten, skulle du ikke kunne løbe ind i maskinelle problemer. CPC464 venter blot på, at du skal indkode de rette oplysninger til databehandling.

(Hvis du laver en fejl, mens du indtaster dette program, må du springe frem til 1.2.7 for at se, hvorledes du kan rette fejlen uden at starte helt forfra med programmet).

Når du indtaster programmet på næste side, så er det ligegyldigt, om du bruger små eller store bogstaver. Datamaten skal nok selv finde ud af det. Derimod er det vigtigt, at du anvender mellemrum og tegn, nøjagtigt som de står, eftersom AMSTRAD BASIC tillader brugen af *reserverede* ord indeni variabelnavne. Se dem i Appendiks VIII.

Tastaturets såkaldt *fysiske* udseende fremgår af figur 4. Når vi omtaler det som *det fysiske tastatur* skyldes det, at mange af tasterne kan omdefineres af brugeren, så de i stedet for de påtrykte tegn og symboler frembringer andre ting, når man trykker på dem. Dette beskrives nærmere i senere afsnit.



Figur 4: CPC464s tastatur

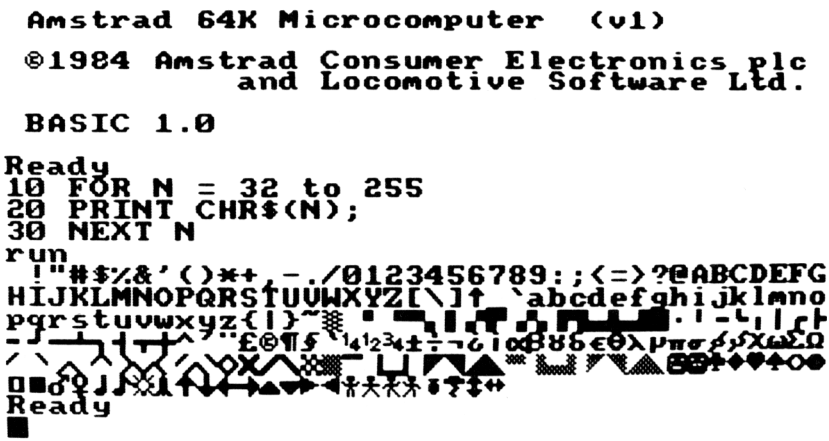
Hvis du trykker på |**ENTER**|, sender du en direkte kommando eller en programlinje ind til behandling i datamaten med en ordre til straks at udføre den, hvis det var en direkte kommando, eller gemme den som en del af et program, hvis den havde et nummer.

|**ENTER**| kaldes sommetider for vogn retur eller simpelthen returtangenten. Det er en terminologi hentet fra dataterminalerne, der var en videreudvikling af fjernskriverne. Termen har overlevet og har sin egen karakterkode i ASCII-sættet. Her hedder |**ENTER**| CR. Indtast nu:

```
10 FOR N=32 TO 255
20 PRINT CHR$(N)
30 NEXT N
RUN
```

1.2.3

Se nu hvad, der er sket på skærmen:



Datamaten fik ved det lille program, du netop har indtastet, ordre til at vise hele sit karaktersæt. Hvis den ikke har gjort det, så har du lavet en fejl, som du ikke har bemærket. Gå frem til 1.2.7 for at se, hvorledes problemet kan løses.

Lad os formode, at alt gik som planlagt. Nu kan vi så se nærmere på, hvad der ligger bag de karakterer, som skærmen viser. Det vil hjælpe dig til nærmere at forstå, hvorledes CPC464 meddeler sig gennem sine karakterer.

Det første, du bør bemærke, er, at datamaten ikke blev beordret til at PRINTe "abcdefghij....osv). Vi sagde til den:

PRINT CHR$(N)

N er et nemt forkortelsesord for *variabel.* Det er rent tilfældigt, at det var bogstavet N, der valgtes. Det skyldes udelukkende, at matematikere elsker at bruge det i en sammenhæng som denne. En variabel er en bid datainformation, som *varierer* alt efter, hvilke instrukser der er givet i et program.

Et tal som 5 er bestemt. Det optræder mellem tallene 4 og 6 og er derfor ikke nogen variabel. Bogstavet N er også bestemt. Det er et bogstav fra alfabetet. Hvordan kunne datamaten så vide, at bogstavet N i dette tilfælde ikke var en fast og bestemt ting, men en variabel. Jo, se, i datamatens regelsæt står, at hvis den skal opfatte N som et bogstav, skal det stå mellem anførselstegn - altså "N" - i så tilfælde ville den have svaret med meldingen *Syntax error* på dansk syntaksfejl, for det ville i dette tilfælde have været i strid med datamatens regelsæt *syntaksen.*

Når vi derfor anvender N, som vist i programmet, fortæller vi datamaten, at der er tale om en variabel med navnet N. Datamatens BASIC-regler siger, at når den har udtrykket *FOR,* skal den forvente en variabel - så derfor formoder den, at det, der følger efter *FOR,* må være variablen.

Vi har også fortalt datamaten, at variablen N strækker sig fra 32 til 255 (N = 32 to 255). Det er en sekvens (rækkefølge), der starter ved 32 og slutter ved 255. Eftersom vi nu har fastsat variablen og dens udstrækning, skal vi nu instruere datamaten om, hvad den så skal gøre med denne viden. Det sker i næste linje:

2Ø PRINT CHR$(N);

Her får datamaten at vide, at den skal omdanne talværdien af det, der er blevet lagt i N til den karakter, der svarer til tallet. Der er her tale om en BASIC-funktion, der udfører netop denne opgave. Nu kan datamaten se efter i sit lager (sin ROM) for at finde en karakter, der svarer til Ns værdi og PRINTe den på skærmen.

Det semikolon, der står til sidst i linjen, er en ordre til datamaten om ikke at lave *vogn retur* og *ny linje,* som den ellers automatisk ville gøre (d.v.s. skrive næste karakter under den anden på næste linje). Vi ønsker nemlig at se karaktererne i een lang linje, der løber videre på næste, når der ikke er mere plads på en linje.

Den næste linje fortæller datamaten, at når den har udført den pålagte opgave med det første tal i sekvensen (32), så skal den gå tilbage til linjen med *FOR* og gøre det igen med den værdi udtrykket *NEXT* nu tildeler variablen N. Dette forløb kaldes en *løkke* og er en af de vigtigste og mest grundlæggende sider ved programmering af datamater og ved databehandling.

Eftersom en FOR-løkke er et af de mest grundlæggende træk ved databehandling, vil du se den dukke op i alle programmeringssprog i en eller anden form. Dens formål er at spare programmøren for lange og trættende indtastninger af en og samme ordre, og du vil hurtigt selv komme til at bruge den i din egen programmering.

Når denne FOR-løkke når til den øverste grænse, vi har sat for den (255), slutter dette forløb, og datamaten kigger på den næste linje efter linje 30. Men eftersom der ikke er nogen næste linje, standser den simpelthen og kommer tilbage til sin klar-tilstand med ordet *Ready.* Dette ord viser os, at datamaten er parat til et nyt program eller til en gentagelse af det gamle ved hjælp af ordren *RUN.* Programmet ligger sikkert opbevaret i datamatens arbejdslager og vil blive der, indtil du giver datamaten andre ordrer eller slukker for den. For når du tager strømmen, taber du alle de data, der står i datamaten, medmindre du har gemt dem på et kassettebånd.

Dette program viser på enkel vis en grundlæggende side af databehandlingen: At alt, hvad en datamat udfører, har noget at gøre med tal. Datamaten har fremvist alfabetet og en hel række andre karakterer, men den har gjort det ved hjælp af tal, der helt præcist peger på de tegn, der ligger som en slags billeder i dens ROM. Det, der populært sagt sker, er, at datamaten på skærmen viser det billede, der peges på. Når du fx trykker på tasten A, så beder du ikke datamaten om at vise et A på skærmen, næh, du giver den ved hjælp af tastaturet en talværdi og beder den kigge i sit lager efter denne talværdi, der indeholder oplysninger om, hvordan bogstavet A skrives på skærmen. Hver eneste karakter har et kodetal, og du kan finde alle disse kodetal i denne vejlednings Appendiks III.

1.2.4

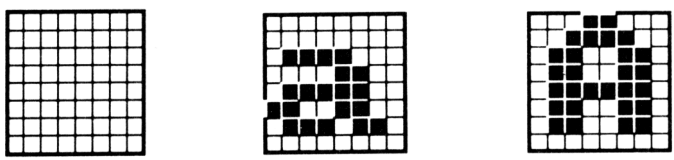
Bliv endelig ikke bekymret, hvis du ikke forstår alle de tekniske udtryk, der bruges på denne side. Det er vigtigt, at vi fuldt ud får forklaret, hvorledes datamaten behandler dine instrukser og kommer til de ønskede resultater, men det er højst sandsynligt, at kun teknisk indstillede mennesker helt og fuld forstår de mere tekniske sider af sagen. Hvis du finder denne side for tung at komme igennem, så spring den roligt over og gå videre med afsnittet 1.2.5.

Koden for bogstavet A er 97. Datamaten forstår end ikke tallet 97. Det skal først oversættes fra menneskets decimale talsystem til et talsystem, datamaten kan opfatte - dette kaldes maskinkode, og de principper, der ligger til grund for dette, er dækket i Appendiks II.

Første gang, du forsøger at oversætte vort vanlige talsystem til den *heksadecimale kode,* man bruger ved datamater, vil du sikkert opfatte det som noget meget, meget svært forståeligt. Det falder os naturligt at tænke på tal i enheder af ti, eftersom vi har ti fingre og ti tæer. Det heksadecimale system, der har enheder af 16, vil forekomme en lige så unaturligt som at prøve at spise med kniv og gaffel i de modsatte hænder.

Det kræver en lignende grad af tankemæssig behændighed at forstå hex (som heksadecimalt forkortes). Men når du først gør det, vil mange af de ting, du opfatter som underligt ved databehandling, let og elegant falde på plads for dig. Talsystemets elegante opbygning vil blive soleklar.

Når datamaten har oversat et tryk på tasten A til et tal, den forstår, ser den i den del af sit lager, som tallet henviser til, og resultatet bliver en serie tal, der udgør karakteren A. Det betyder, at den karakter, du ser på skærmen, er opbygget af en datablok, der ligger i datamatens lager som et mønster, sammensat af tal. På figuren til venstre ser du det net, karakteren opbygges på:



Figur 5: Tomt karakternet, lille a, stort A

Nettet kan opfattes som et skakbræt af vandrette linjer og lodrette søjler. En karakter opbygges af punkter på dette net. Hvis du forestiller dig nettet som en række lamper, der er tændt og slukket, så har du i nettet til venstre kun slukkede lamper, mens du i de to andre har en række tændte og slukkede. Sådan arbejder datamaten faktisk. De data, der ligger i dens lager, fortæller den, om en prik (eller en byte) er tændt eller slukket. Sagt på en anden måde, om det er 0 eller 1.

Hver karaktercelle er opbygget af otte rækker gange otte søjler. Hvis du ikke finder den karakter, du skal bruge i CPC464s sæt, kan du selv lave en ved brug af nøgleordene *SYMBOL* og *SYMBOL AFTER* som beskrevet i Kapitel 8.

Brugerdefinerede karakterer laves af enhver kombination af 0 til 64 prikker, opstillet på enhver mulig vis. Så hvis man anvender alle de muligheder, som netværket rummer, er der plads til mange flere karakterer end de 255, der ligger i CPC464. Hvis du hertil lægger, at man kan sammenstille hele blokke af karakterer, så de danner større karakterer, så bliver mulighederne med brugerdefinerede karakterer bogstaveligt taget uendelige.

Bagest i bogen, i appendiks IX, kan du iøvrigt finde et lille program, der giver dig danske karakterer på datamaten.

1.2.5 Men tilbage til programmet!

Resultatet af det første program, du indtastede, blev ret rodet af udseende. I toppen af skærmen har du fortsat datamatens velkomsthilsen. Hvis vi viskede skærmen ren, før vi startede databehandlingen, ville det se meget pænere ud. Det gør vi ved at tilføje een linje til programmet.

Indtast følgende linje, der hvor markøren står (markøren er den firkantede blok, der står umiddelbart til venstre under ordet *Ready,* og hvis du ikke ved det, hvorfor i alverden er du så her i stedet for i Indføringskurset i starten af bogen??):

```
5 CLS
RUN
```

Nu vil du se, hvor pænt skærmen bliver visket ren, før datamaten begynder at skrive karaktersættet i øverste venstre hjørne.

Dette viser også en af de mest forstående sider ved BASIC-sproget: Den orden, hvori du indtaster linjerne, er totalt ligegyldig. Datamaten skal nok få en nummereret linje ind på den rigtige plads - også selv om du ikke har kaldt programmet frem på skærmen ved hjælp af ordren *LIST.*

For at kontrollere, at datamaten virkelig har gjort dette, kan du nu prøve at indtaste *LIST.*

1.2.6 LISTning

Du kan let se, hvad en datamat har gemt i sit arbejdslager ved at bede den om at LISTe. Hvis du skrev LIST for lidt siden, viste skærmen følgende:

```
5 CLS
10 FOR N=32 TO 255
20 PRINT CHR$(N);
30 NEXT N
Ready
```

Dette program vil forblive i CPC464s arbejdslager, indtil du enten:

- ★ Slukker
- ★ Laver RESET - ved at trykke |**CTRL**| |**SHIFT**| |**ESC**| i denne rækkefølge og holde dem nedtrykket, indtil Reset indtræffer.
- ★ LOADer eller RUNer et program fra dataoptageren.
- ★ Indtaster NEW |**ENTER**|, som resetter alle variabler og visker datamatens programlager rent uden at ændre på ting som skærmformat (Mode) eller valgte farver.

Nu vil det være en god ide, hvis du programmerede en af datamatens funktionstaster, så et enkelt tryk på denne medførte udførelse af ordren: |**ENTER**|CLS:LIST|**ENTER**| - det vil lette dig enormt ved indtastningen af programmer. For at gøre dette, skal du indtaste:

```
KEY 138,CHR$(13)+"CLS:LIST"+CHR$(13)
```

Tryk nu på punktumtasten på det lille taltastatur ude til højre. Du har nu programmeret denne tast. Du kan forudprogrammere ialt 32 taster på denne vis, og du kan vælge enhver tast, du måtte ønske at forvandle til en anden karakter end den, der er trykt på den. Se nærmere om TAST-kommandoerne i Kapitel 8.

Hvis det er et langt program, du er ved at udvikle, så definer tasten således:

```
KEY 138,CHR$(13) + "CLS:LIST"
```

så kan du med denne tast indføre den nummerrække, du ønsker, eller - hvis du trykker to gange på den - få en fuld listning.

Når du eksperimenterer med farver, kan du komme ud for en farvesammenstilling, der gør det umuligt at læse skærmen, fordi baggrunden og teksten har samme farveværdi. Dette kan du sikre dig mod ved følgende forprogrammering af en tast:

```
KEY 139,CHR$(13) + "MODE2:INK1,Ø:INKØ,9" + CHR$(13)
```

...nu behøver du blot at trykke på den mindste af de to ENTER-taster (den på det lille taltastatur), så vil du komme tilbage til en farve- sammensætning, der gør teksten synlig (uden at du mister det program, du har i arbejdslageret).

De koder, der bruges på de brugerdefinerbare taster, bliver slettet samtidig med resten af datamatens data, når du slukker for datamaten. De er jo indtastet på samme vis som programinstrukserne. Så når du har fundet frem til en række koder, du har brug for, er det en god ide at indsætte dem i et program og at gemme det på et kassettebånd, så du blot når som helst kan spille det over i datamaten.

1.2.7 Redigering

Det er menneskeligt at fejle, og du vil uundgåeligt lave fejl, når du indtaster et program. Vi byder velkommen til dette afsnit til alle, der sprang frem hertil fra afsnit 1.2.2!

CPC464 prøver at gøre det så let som muligt at rette de fejl, du begår, men skal på samme tid søge at undgå de problemer, der kan opstå, hvis du ved en fejltagelse *skriver oveni* karakterer, som du ikke ønskede ændret.

Det tastbundt, der styrer markørens bevægelser, giver dig midler til at trække datamatens opmærksomhed hen på den del af skærmbilledet, du ønsker at ændre.

Når du indtaster forkert i en nummereret linje, fx:

```
1Ø FOR N = 332 TO 255
```

så har de flere mulige udveje:

1. Du kan trykke på |**ENTER**| og genindtaste hele linjen. Den ukorrekte linje vil så blive slettet fra datamatens lager, idet den nye linje skrives oveni, fordi den har samme linjenummer.

2. Du kan trykke på tasten med venstrepilen og flytte markøren tilbage til den ukorrekte indtastning:

10 FOR N=3[3]2 TO 255

Bemærk, at karakteren under markøren vises inverteret - altså i kontrast til den øvrige tekst. Med andre ord, karakteren, der normalt har samme farve som markøren, har nu fået samme farve som baggrunden (PAPER), således, at den kan ses gennem markøren, der nu står oveni den.

Tryk nu på tasten |**CLR**| (en forkortelse for clear eller visk), og den karakter, der står i markøren, vil nu forsvinde, mens de øvrige rykker op og udfylder mellemrummet i linjen:

1Ø FOR N = 32 TO 255

Tryk igen |**ENTER**|, og den rettede linje vil være den, datamaten husker. Du behøver ikke at flytte markøren hen i slutningen af linjen. Datamaten sluger hele linjen, uafhængigt af markørens placering.

3.Du kan også flytte markøren hen til karakteren umiddelbart til højre for den, du ønsker at slette:

10 FOR N=33[2]2 TO 255

Tryk nu på tasten |**DEL**| (forkortelse for delete eller slet), og karakteren umiddelbart til venstre for markøren slettes. Markøren trækker linjen afsted bag sig uden at påvirke den karakter, der står indeni den.

Tryk nu |**ENTER**|, og datamaten sluger den rettede linje.

1.2.8 Eftertanke

De forudgående metoder er udmærkede, hvis du får øje på fejlen, før du når til enden af linjen og taster |**ENTER**|. De fleste fejl optræder imidlertid, uden at du bemærker det, og dukker først op, når du siger RUN til programmet. Så svarer datamaten med en fejlmelding (Appendiks VIII).

En del fejl vil få datamaten til at fremvise fejllinjen med redigeringsmarkøren placeret over første tegn til venstre. Hvis det er tilfældet, så kan du bruge en af de ovennævnte metoder.

Hvis fejlen ikke bliver fremhævet for dig på denne måde, bliver du nød til at LISTe programmet for at finde årsagen til problemet og rette fejlen.

1.2.9 Redigering med Copymarkør

Først LISTer du programmet (lad os gå ud fra, at det er det lille prøveprogram, der kan være på eet skærmbillede):

```
5 CLS
10 FOR N=32 TO 255
20 PRINT CHRS(N)
30 NEXT N
```

Fejlen ligger i linje 20, hvor der står S i stedet for $. ($-tegnet fortæller datamaten, at den skal betragte karaktererne umiddelbart efter som tekst og ikke som nummeriske data). Du kan nu enten genindtaste linje 20 fra begyndelsen og indsætte den i programmet med |**ENTER**|, eller du kan bruge skærmredigering som følger:

Hold på tasten |**SHIFT**| (enten den i højre eller venstre side af tastaturet) og tryk derefter på opadpilen.

Nu kan du enten taste den een linje ad gangen, eller du kan holde på den, indtil den automatiske repetition sætter i gang og flytter den for dig. I det øjeblik, du flytter fingeren fra tasten, standser markøren, og med lidt øvelse vænner du dig hurtigt til virkningen. Hvis du kører forbi linjen, kan du køre nedad igen med nedpilen.

Denne handling får en copymarkør til at skille sig fra hovedmarkøren. (De to markører ser ens ud). Copymarkøren flyttes til den linje, du ønsker at ændre. Begynd med copymarkøren anbragt over linjens første karakter.

```
5 CLS
10 FOR N=32 TO 255
20 PRINT CHR$(N);
30 NEXT N
Ready
```

Hvis du ved hjælp af piletasterne flyttede hovedmarkøren til den linje, du ønskede at ændre, uden at holde på |**SHIFT**|, ville datamaten ikke anerkende dette som en gyldig handling, eftersom den kun accepterer de karakterer, der indskrives direkte efter hovedmarkøren.

Hvis du forsøger at overskrive en linje på denne vis, kan du let komme ud af fælden ved at trykke tasten |**ESC**|, FØR du trykker på |**ENTER**| eller en funktionstast, der indeholder chr$(13). Hvis du ved en fejl indtaster kommandoordet *NEW* og derefter trykker |**ENTER**|, så mister du hele dit program, så pas endelig på.

Når du er begyndt at indtaste en linje og så prøver at gå ud af den, uden at trykke |**ENTER**|, vil datamaten hyle, når du gør noget ulovligt. Men så snart du er undsluppet problemet ved at trykke |**ENTER**|, vil du ikke tabe noget af programmet, medmindre den første karakter, du indtastede, svarede til et linjenummer i dit program. Så vil din fejl nemlig erstatte denne linje, og så må du skrive den om.

Når copymarkøren er korrekt placeret, trykker du på tasten |**COPY**|, indtil copymarkøren når frem til den karakter, du ønsker at ændre:

```
5 CLS
10 FOR N=32 TO 255
20 PRINT CHR$(N)
30 NEXT N
Ready
20 PRINT CHR
```

nu slipper du |**COPY**| og taster $ - som vil vise sig under hovedmarkøren, der følger med en plads til højre som sædvanlig.

Flyt nu copymarkøren forbi S-fejlen, mens du trykker på tasten |**SHIFT**| og tryk så een gang på højrepilen. Nu standser copymarkøren over (. Slip |**SHIFT**| og hold på |**COPY**| igen, indtil du når ud forbi enden af linjen. Tryk |**ENTER**| og den rettede linje i bunden af skærmen erstatter den ukorrekte linje i LISTningen.

Du kan kombinere disse metoder ved simpelthen at lave **COPY** af hele linjen med fejl. Derefter redigerer du den med hovedmarkøren, idet du bruger piletasterne og |**CLR**| og |**DEL**| uden |**SHIFT**|, før du trykker |**ENTER** ved afslutningen. Hvis du holder på |**CTRL**| og enten venstre- eller højrepilen, flytter du i eet skridt markøren hele vejen til venstre eller højre i den linje, du retter. Prøv at øve dig lidt i det, så det bliver let for dig.

Endelig kan du rette ved at indtaste:

```
EDIT 2Ø
```

så svarer datamaten med:

```
20 PRINT CHR$(N);
```

Nu bruger du simpelthen piletasterne sammen med |**CLR**| og |**DEL**|, som vist ovenfor, og når du er tilfreds med resultatet, trykker du igen |**ENTER**| som før. Hvis du bliver fanget i en fælde, slipper du ud med |**ESC**| og kan LISTe igen. Linjen, hvori du trykkede |**ESC**|, vil ikke blive overskrevet.

LIST nu igen programmet, og du vil se den rettede version. Hvis den stadigvæk ikke er rigtig, så prøv igen!

Hidtil er vi kun lige begyndt at se på fundamenterne i CPC464. For at kigge på de to andre funktionstilstande må du indtaste:

```
MODE Ø
RUN
```

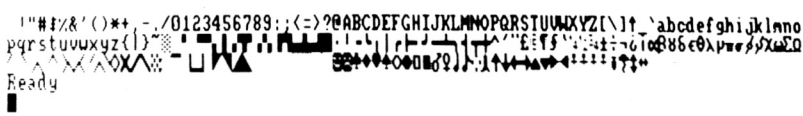
Bemærk, at skærmen først tømmes, og at dit program nu vises med 20 karakterer lange linjer på skærmen:

For at komme tilbage til det oprindelige skærmbillede, taster du:

MODE 1
RUN

Og vi er tilbage, hvor vi begyndte. For at se programmet med 80 tegns skærm, taster du:

MODE 2
RUN



Nu har vi gjort en begyndelse, og du skulle være kommet ud over den første nysgerrighed. Trænede brugere vil allerede være i gang med at overføre deres yndlingsprogrammer til denne særlige BASIC-dialekt, mens mindre erfarne brugere bør fortsætte gennem startafsnittene for at få et overblik over de BASIC-træk, der er specielle ved denne datamat.

2 Dataoptageren

LOADning og dataoptagerens funktioner

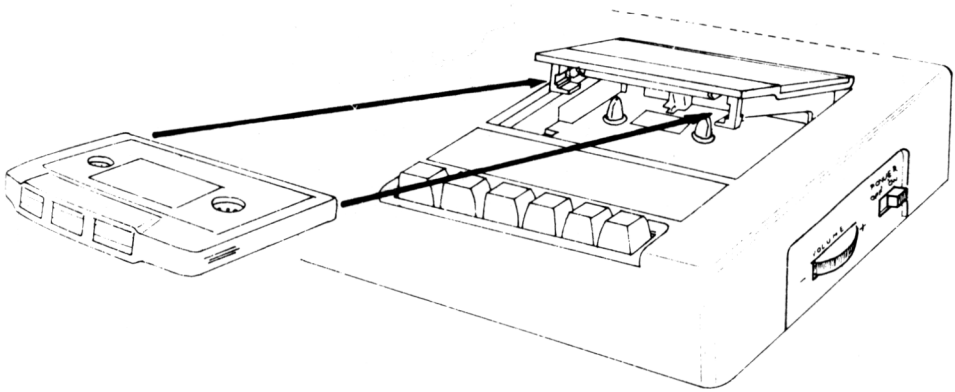
Emner der dækkes i dette kapitel:

★ Ligheder og forskelle mellem dataoptagere og lydbåndoptagere
★ LOADning og RUNning af programmer fra kassette - velkomstbåndet
★ De valgfrie båndhastigheder
★ Hvordan programmer gemmes på kassette
★ Indlæsningsfejl

AMSTRAD CPC464-datamatens arbejdslager kan kun opbevare data, så længe der er strøm på datamaten, og denne er tændt. Datamaten har med andre ord en flygtig hukommelse. Hvis du ønsker at opbevare data eller programmer, når der ikke er strøm på, må du overføre dem til en båndkassette eller et andet varigt lagermedium (fx diskettesystemet, der er ekstraudstyr til CPC464).

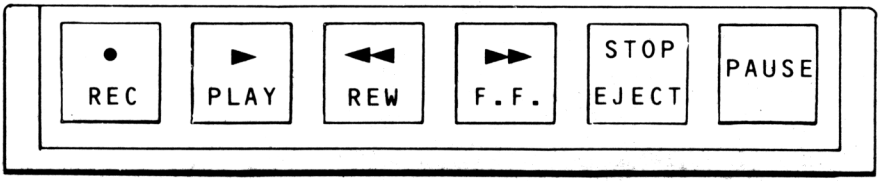
2.1 Optagerknapperne

På højre side af tastaturenheden finder du dataoptageren, som er vist på figur 2.1. Mekanikken i denne er omtrent som i en almindelig kassettebåndoptager - bortset fra, at den elektronik, der kontrollerer signalerne, er specielt egnet til at optage og gengive datamatsignaler.



Figur 2.1: Den rigtige måde at indsætte kassetten i dataoptageren

På samme vis ligner knapperne til dataoptageren dem, du finder på en vanlig kassettebåndoptager.



Figur 2.2: CPC464s dataoptagerknapper

Bemærk, at du skal trykke hårdere på disse taster end på selve datamatens tastatur.

|**REC**| - optageknappen fungerer sammen med |**PLAY**| og optager data, når den gennem et program har fået ordre til det. Du kan ikke nedtrykke denne knap, medmindre der er en kassette i optageren (og denne kassette ikke har fået sikkerhedstapperne fjernet), ligesom lågen over kassetten skal være lukket. Hvis du vil bruge denne funktion, trykker du først |**REC**| og holder på knappen, mens du trykker på |**PLAY**|. Datamaten vil overføre data til kassetten, når den af et program får besked på det - eller når du indskriver en SAVE ordre direkte fra tastaturet.

|**PLAY**| - afspilningsknappen bruges til enten at LOADe eller RUNe et program fra kassetten. Datamaten vil så læse data fra kassetten, enten når den af et program får besked på det - eller ved en direkte kommando fra tastaturet. Både |**REC**|- og |**PLAY**|-knappen springer automatisk op, hvis et kassettebånd løber ud.

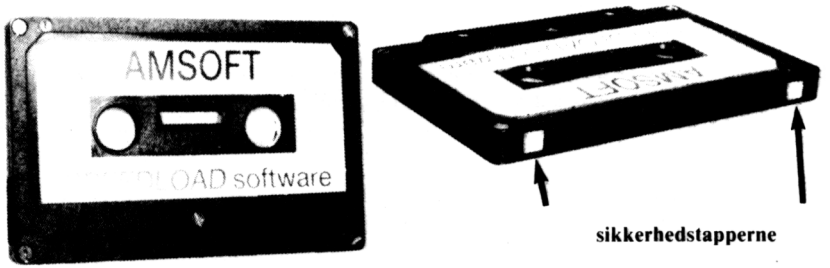
|**REW**| - tilbagespolingsknappen har ikke nogen mekanisk stopmekanisme, der får den til at springe op, når båndet løber ud. For ikke at beskadige den, bør du ikke lade den stå og løbe, så risikerer du, at motoren løber varm og bryder sammen.

|**F.F.**| - knappen til hurtig fremspoling. Ejheller denne knap har automatisk stop. Så heller ikke denne bør du glemme at trykke op, når båndet er løbet ud.

|**STOP/EJECT**| - denne knap standser enhver båndkørsel og får alle optagerens knapper til at springe op i udgangspositionen. Hvis man slipper knappen og trykker igen, springer låget op. Kassetten kan ikke fjernes fra optageren, før denne er standset.

|**PAUSE**| - denne knap giver en mekanisk pause sammen med knapperne |**PLAY**| eller |**REC**| og |**PLAY**|. Knappen bør ikke bruges til at lave en pause under indlæsning eller udlæsning af data, idet der så vil optræde en læsefejl. Alle pauser under afspilning og indspilning styres ved interne instrukser fra CPC464-programmellet, og derfor benyttes den mekaniske pausemulighed næsten ikke.

2.2 Skrivebeskyttelse (figur 2.3)



sikkerhedstapperne

For at hindre, at en optagelse slettes ved en fejltagelse, er alle kassetter forsynet med såkaldte sikkerhedstapper, som vist på fotoet. Disse tapper kan brækkes med en kuglepen, en saks eller lignende. Når de er borte, kan |**REC**|-knappen ikke nedtrykkes på din dataoptager. Forsøg ikke at bruge magt, så risikerer du at beskadige den.

Hver side har sin egen tap. Hvis du vil beskytte begge sider, må du altså fjerne begge tapper.

Hvis du senere vil optage på en kassette, der er beskyttet på denne måde, skal du sætte et stykke selvklæbende tape over hullet, så virker |**REC**| igen.

2.3 LOADning fra kassette

Figur 2.1 viser den korrekte måde at indsætte kassetten på. Båndet skal være kørt helt tilbage (fra højre til venstre spole), og hvis det ikke er det, må du først trykke |**REW**|, indtil det er sket. Hvis båndet ved et uheld er løbet lidt ud af kassetten, må du forsigtigt spole det ind igen ved at sætte en kuglepen ind i et af spolehullerne. Ellers risikerer du at ødelægge båndet og dine data.

Mens du let kan afspille en musikkassette, hvor båndet har fået et knæk, kan et sådant knæk ødelægge dine data. Skulle du komme til at beskadige et bånd, men alligevel har kunnet overføre indholdet til datamaten, så skynd dig at lave en sikkerhedskopi på et andet ubeskadiget bånd og smid den beskadigede kassette væk, så du ikke ved en fejltagelse kommer til at bruge den.

2.4 Velkomstbåndet

Med din AMSTRAD CPC464-datamaten fulgte et bånd, der indeholder en del demonstrationer af datamatens lyd- og grafikmuligheder samt af dens indbyggede programmel - AMSTRAD BASIC - og maskinens operativsystem (MOS).

En del af dette MOS er selve det kassettestyrende programmel, som bl.a. rummer nogle korte ordrer til at indlæse og udlæse data og programmer fra båndet. De ordrer, du hyppigt træffer på er *LOAD* og *RUN* (INDLÆS et program og KØR et program).

For at gøre brugen af datamaten så let som mulig rummer CPC464 en del specialfunktioner, der gør det lettere for dig at indtaste noget, indtil du har fået tilstrækkelig øvelse ved tastaturet. Hvis du lige har tændt for datamaten, ser du på dens *opvågningsbillede* og dens klarmelding *Ready*.

Indsæt nu tapen, som vist på figur 2.1, stil båndtælleren på 000 med den lille knap ved siden af den. Tryk på tasten |**CTRL**|, og mens du holder den nedtrykket, trykker du på den lille tast |**ENTER**| i nederste højre hjørne af taltastaturet. Så svarer datamaten med:

RUN"
Press PLAY then any key:

eller på dansk *KØR - tryk SPIL og derefter en hvilken som helst tast.*

Det, du ser, er et eksempel på forprogrammering af en tast, som omtalt i Kapitel 1, hvor vi instruerede datamaten om at udføre en hel række kommandoer ved et enkelt tryk på en tast (engelsk KEY).

Du kunne selvfølgelig godt have indtastet instruksen *RUN"* og trykket på |**ENTER**|, men det er jo noget nemmere med blot to tasttryk.

De fleste gange udfører de to |**ENTER**|-taster samme funktion, men der er tilfælde, hvor man kan ændre funktionen på den mindste af de to taster af praktiske grunde. Emnet *de brugerdefinerbare taster* dækkes mere indgående senere i bogen.

Tryk nu på dataoptagerens tast |**PLAY**|, indtil den låser. Optageren starter ikke, før du trykker på en anden tast. Det er en god vane i dette tilfælde at bruge den store |**ENTER**|-tast. Det er ret almindeligt i databehandling at finde udtrykket "press any key". Det dækker over, at man har givet programmet ordre til at afvente en besked om at gå videre med en række ting, i stedet for at man selv skal indtaste alle de nødvendige ordrer.

Det korrekte ville være at sige: *tryk på en hvilken som helst tast, undtagen* |**SHIFT**| |**CAPS LOCK**| |**CTRL**| |**RSC**| *eller nogen af tasterne på dataoptageren* - men af forenklingsgrunde må man altid gå ud fra, at visse ting er kendt i forvejen. Hvis du ser vendingen i denne vejledning, i et program fra AMSOFT eller andre programsælgere, så gå automatisk ud fra, at det er det, vendingen betyder.

Tasttrykket vil ikke resultere i, at der viser sig en karakter på skærmen, men dataoptageren begynder at køre. Hvis den ikke gør det, må du prøve at trykke på en anden tast og se efter, at du ikke er kommet til at trykke på pauseknappen.

Hvis du trykker på mere end en tast, vil datamaten ignorere det. Den reagerer ikke på tasttrykket, når den er gået igang med LOADningen.

Læg mærke til, at du ikke skrev navnet på det program, du ville have indlæst. Hvis der ikke står mere på en linje, hvor man har skrevet *RUN"*, vil datamaten udsøge sig det første program, der står på båndet og begynde indlæsningen. Når datamaten har fundet det først korrekt optagne program, kommer der en melding på skærmen. I dette tilfælde står der:

Loading WELCOME 1 block 1

Det er en meddelelse til dig om, at datamaten har fundet den første af den række blokke, der udgør programmet WELCOME 1. Hvert program optages i datablokke, der er ca. 2K-byte lange. De indlæses en for en i datamaten, og denne viser dig på skærmen nummeret på den blok, der i øjeblikket indlæses. Efter hver blok holder båndet en lille pause, før den næste blok begynder. Når den kommer, skifter blok-tallet på skærmen.

Hvis datamaten under indkørslen finder dårligt optagne data, vil den vise en fejlmelding (se dem i Appendiks VIII), som fortæller om problemet. Generelt kan du ikke gøre andet end at forsøge at køre programmet igen, indtil det indlæses uden fejl. Vi går nu ud fra, at båndet kører. Læs nu instrukserne på skærmen, så klarer Velkomsttapen resten...

2.5 Almindelig og hurtig kørsel

Din CPC464 har to hastigheder på dataoptageren: Den *sikre* på 1000 baud (databits pr. sekund) og den *hurtige* på 2000 baud. Den hurtige indlæser og udlæser med andre ord data dobbelt så hurtigt som den sikre, men som navnet siger, sker det med en mindre sikkerhedsmargin, og man risikerer derfor undertiden, at data falder ud undervejs (det kan skyldes for dårlig båndkvalitet, eller at tonehovedet på den optager, der er brugt, står lidt anderledes end på din). Med andre ord: Det man sparer i tid, risikerer man at sætte til igen, fordi man skal indkøre programmet en gang til.

Hvis du bruger bånd af en rimeligt god kvalitet, og kun ind- og udlæser på din egen dataoptager, kan du imidlertid være rimeligt sikker på, at den hurtige metode virker de allerfleste gange. Det samme gælder også ved de fleste bånd, du kan købe med færdige programmmer. AMSTRAD råder dog programfirmaerne til at optage programmerne med begge hastigheder på deres bånd.

Datamaten læser automatisk den fart, hvormed båndet er indspillet. Når du skal gemme et program - med ordren *SAVE* - må du fortælle datamaten, om du ønsker at bruge den hurtige metode, ellers vil datamaten automatisk vælge den sikre.

Valget af den hurtige sker ved, at du, når datamaten melder *Ready,* skriver:

SPEED WRITE 1

og hvis du vil tilbage til den sikre fart, kan du enten lave Reset på datamaten (hvorved du mister alle data), eller du kan, når datamaten melder *Ready,* indtaste:

SPEED WRITE Ø

2.6 Optagelse af programmer og data

BASIC har flere ordrer, der drejer sig om opbevaring af data på en kassette. Vi giver her et kort resume af dem sammen med nogle eksempler.

2.6.1 SAVE "filnavn"

Den mest ligetil metode til at gemme data på en kassette er at benytte ordren *SAVE,* når CPC464 melder *Ready,* efter at den har udført eller LISTet et program. Vi vil som eksempel anvende det korte program, der viste karaktererne i Kapitel 1.

Det "filnavn", der er nævnt i overskriften, er programmets navn. Det kan være 16 karakterer langt og kan bestå af enhver kombination af de karakterer, du kan finde på tastaturet, inklusive mellemrum. Hvis du forsøger at skrive et navn på mere end 16 karakterer, vil datamaten bortkaste den 17. og de efterfølgende karakterer. Vi går ud fra, at du har programmet i datamatens arbejdslager. Så indtaster du:

SAVE "KARAKTERER"

Datamaten svarer med at skrive:

Press REC and PLAY then any key:

eller *tryk REC og PLAY og derefter en eller anden tast.* Så begynder båndet at køre, og datamaten gemmer programmet under navnet "Karakterer".

VIGTIGT

Datamaten kan ikke mærke, om du virkelig har trykket på de rette knapper på dataoptageren, så hvis du ikke trykkede |**REC**|, men blot |**PLAY**|, så begynder båndet at køre, og det vil se ud, som om dataoptageren SAVEr. Det er altså noget, du selv må sikre dig.

PAS PÅ: Hvis du ved en fejltagelse trykker |**REC**| og |**PLAY**|, når du blot vil afspille et bånd, så sletter du det program, der måtte stå på båndet. Hvis du ikke standser den ved at trykke |**ESC**|, så vil dataoptageren fortsætte helt til båndets afslutning og slette alt på det, fordi datamaten ikke har kunnet søge sig et program at standse ved. Hvis du er bange for at tabe data ved slig fummelfingrethed, så gør det til en vane at bryde sikkerhedstapperne på kassetten, når du har optaget et program på den. Det er altid lettere at sætte klæbestrimmel over hullerne, hvis du skal optage noget nyt på den, end at skulle til at begynde helt forfra med indtastning af et program.

Der er fire måder, hvorpå filer kan SAVEs på CPC464. Vi har lige set på den generelle metode. Her kommer de mere specielle:

2.6.2 SAVE ''filnavn'', A

Fremgangsmåden er den samme som den ovennævnte, bortset fra efternavnet *,A* der er en instruks til datamaten om, at SAVE programmet eller datafilen i form af en ASCII-tekstfil, snarere end den kortere form, datamaten automatisk bruger.

Denne metode bruges ved filer, der stammer fra tekstbehandlingsprogrammer eller andre seriøse programme, og vi vil senere drøfte brugen af den, når vi kommer til denne programtype.

2.6.3 SAVE ''filnavn'', P

,P fortæller datamaten, at den skal beskytte (engelsk Protect) de data, der står på kassetten. Det medfører, at fremmede ikke uden videre kan læse disse data blot ved at RUNne programmet og standse programkørslen med |**ESC**|. Programmer, der er gemt med denne metode, kan kun udføres med kommandoerne *RUN* eller *CHAIN*. Hvis du forventer senere at få brug for at rette eller ændre programmet, bør du gemme en kopi i ubeskyttet form.

2.6.4 SAVE ''filnavn'',B, startadresse, længde (,startpunkt)

Denne metode gør det muligt for dig, at lave et binært SAVE, hvorved hele blokke af data, der ligger i datamatens RAM, gemmes på kassetten, nøjagtig som de står i det indre lager. For at kunne gøre det må man fortælle datamaten helt præcis, hvor starten er på det lagerafsnit, du ønsker at gemme, hvor langt det er, og hvilken adresse der er startpunktet, hvis filen skal RUNne som et program.

Formålet med denne binærmetode er at give dig mulighed for at gemme et skærmbillede direkte på kassetten. Det kan fx bruges til at opbygge et *titelbillede* i lange programmer, så man får brugt den lange ensformige ventetid, mens programmet LOADer.

2.6.5 Unavngivne filer og CAT

Hvis du vil SAVE en fil uden at give den et navn, taster du:

SAVE ''''

så vil BASICen gemme den som en *unavngiven fil.* På en kassette kan man gemme så mange programmer med samme navn (inkl. unavngivne filer), som man måtte lyste - dette i modsætning til et diskettesystem, hvor det kræves, at hver fil har sit helt eget navn.

Du vil hurtigt fare helt vild i dine programmer, hvis du ikke giver dem navne, der kan minde dig om, hvad de indeholder, og vi råder dig også til at sætte numre eller datoer på navnet, så du udfra disse kan se, hvilke der indeholder de nyeste udgaver af dine programmer og datafiler.

Man kan se, hvad en kassette indeholder, ved at bede om et katalog. Det sker med ordren *CAT*. Så svarer datamaten med den sædvanlige ordre om at trykke |**PLAY**| og en anden tast.

Nu afleverer datamaten en liste over båndets indhold. Den viser alle filnavne med STORE BOGSTAVER, antallet af blokke i filen og en enkelt karakter, der viser filens art:

$ er et almindeligt BASIC-program
% er et beskyttet BASIC-program
★ er en ASCII-tekstfil
& er en binær fil

Et *OK* i slutningen af linjen viser, at filen er læsbar, og at den ville være blevet indlæst, hvis du havde bedt datamaten om det. Hvis du udfører CAT-funktionen, mens der er et program i datamatens arbejdslager, påvirkes dette ikke i mindste måde.

2.7 Læsefejl

Hvis du under LOADningen af et program eller en datafil ser teksten *Read error* (læsefejl) på skærmen, vil datamaten fortsætte med at læse blokkene den finder efter fejlen, men den vil ikke forsøge at LOADe dem, medmindre den identificerer en af dem som blok 1 i det program, den forgæves søgte at indlæse.

Dette betyder, at efter en sådan melding kan du godt standse kassetten med de sædvanlige tryk, spole tilbage til starten og forsøge en ny indlæsning af programmet eller filen. Det kan meget vel ske, at datamaten har held med et nyt indlæsningsforsøg.

Læsefejl kan skyldes forskellige ting. Den mest almindelige grund er, at båndet er blevet beskadiget - det kan være krøllet, strakt, ridset eller på anden måde fysisk skadet.

Fejlen kan også være elektrisk. Den kan opstå, hvis du slukker for datamaten, mens spille- eller optageknapperne var nedtrykkede. Hvis dette er tilfældet, kan båndet få en skade. Når disse knapper er nedtrykket, holdes båndet nemlig trykket an mod tonehovedet. Når du slukker for strømmen, kan der komme et lille strømstød fra tonehovedet. Dette stød afsætter sig som et signal på båndet og ødelægger totalt de data, der er på dette lille sted.

Et bånd, der i længere tid står stille og trykket an mod tonehovedet, kan krølle eller blive strakt. Hvis du derfor forlader din datamat midt under brugen, og du ikke ved, om du er borte i længere tid, så sørg for, at udløse |**REC**| og |**PLAY**| - hvis altså der ikke er tale om, at du venter på, at en LOAD eller SAVE skal blive færdig.

Læsefejl kan også indtræffe, hvis du brugte pauseknappen under en ind- eller afspilning af et bånd, eller hvis båndet oprindeligt blev indspillet på en anden CPC464, hvor tonehovedet ikke var korrekt indstillet.

Læsefejl opstår af og til helt uden forklaring. Båndkassetter blev ikke oprindeligt skabt til opbevaring af data, og derfor har de på flere punkter nogle svagheder, der gør, at de ikke kan leve op til den standard, man møder på professionelle båndstationer til data.

På trods af dette er kassetten særdeles velegnet til jobbet som standard-lagermedium for datamater i den lavere prisklasse. De fysiske begrænsninger i størrelsen på de magnetiske partikler på båndets overflade sammenkædet med den hastighed, hvorunder båndet passerer tonehovedet, sætter visse grænser på den fart, hvorved data kan overføres mellem bånd og datamat. At forsøge at presse farten op over de tolerancegrænser, systemet har, vil gøre driften upålidelig - især når man anvender de metoder, der er gængse ved massefabrikation af forholdsvis billige programmer.

BEMÆRK at kassetter, der indeholder programmer fra andre typer datamater, ikke kan indlæses på CPC464. Det kan godt være, at programmerne ligner CPCens, det kan også godt være, at de lyder ligesådan, når du afspiller dem over en båndoptagers højttaler, men de vil ikke LOADe og RUNe på CPC464. Hvis du trods dette skulle finde programmer, der synes både at LOADe og RUNe - på trods af, at de stammer fra andre datamatfabrikater - vil det interessere os overmåde meget. Det vil i så fald glæde os, hvis du vil meddele os alle mulige detaljer omkring den pågældende datamat- og programtype

2.8 Kassetteovervejelser

Skønt dataoptageren vil acceptere næsten enhver type kassetter på C90 eller derunder, bør du kun anvende C12 (6 minutter på hver side) eller højst C30 (15 minutter på hver side). Programmer, der gemmes i enden af meget lange bånd, er svære at finde, medmindre du er fornøjet ved at sidde og vente på det (og kan huske det filnavn, du gav programmet) - eller du omhyggeligt har brugt båndtælleren og har anvendt tallet fra den i dit katalogiseringssystem (altså i filnavnet). Hvis du ikke ønsker at risikere at indspille hen over et program på en meget lang kassette, så må du være meget omhyggelig med at lokalisere startpunktet og passe meget på med, at du ikke kommer til at optage oveni andre programmer, du ønsker at beholde.

I det store og hele er det mindre besværligt at anvende korte kassetter, hver med så få programmer som muligt. C12-kassetter er relativt billige - og hvis du af en eller anden grund skulle beskadige dit bånd, er det mindre ærgerligt at bortkaste en lille kassette end en stor.

Slutteligt bedes du erindre, at kommercielt programmel (altså færdigindspillede programmer fra programhuse) næsten altid er beskyttet af copyright-lovene. Du bør ikke forsøge at kopiere eller på anden vis mangfoldiggøre sådanne programmer bortset naturligvis fra den slags programmer, der ligefrem råder dig til at lave en sikkerhedskopi. Men også i dette tilfælde *kun* til din egen private brug.

At kopiere de programmer, som andre mennesker skal leve af at sælge, om det så bare sker til *min ven,* er ikke blot en moralsk forkastelig handling (hvis mindste konsekvens er, at ingen til slut gider skrive gode programmer til programhusene), men ovenikøbet strafbart. Det har adskillige mennesker både her i landet og i udlandet mærket på deres pengepung, når deres *piratvirksomhed* er blevet afsløret af programmets ophavsmand eller licensindehaver.

3 BASIC-skolen

En kort introduktion til programmer, skrevet med AMSTRAD BASIC

Emner, der dækkes i dette kapitel

★ Syntaksreglerne og beskrivelse af syntaksen
★ PRINT-kommandoer, strømme og skærmformattering
★ ZONEr

3.1 Grundlæggende BASIC

I Appendiks II viser vi det grundlæggende forhold mellem CPC464s indbyggede BASIC og datamatens indre operationer. Hvis du ikke tidligere har prøvet at programmere en datamat, skal du ikke være nervøs. Vi prøver at hjælpe dig frem ad vejen så blidt som muligt - men det har af og til været nødvendigt at gå ud fra visse forudsætninger, som kan virke forvirrende på en begynder. Hvis det er tilfældet med dig, foreslår vi, at du prøver at kigge i de mange bøger, der er skrevet for begyndere i programmeringens kunst.

Du skulle kunne arbejde dig gennem dette kapitel, idet du prøver de simple øvelser, uden at du behøver at forstå alt. Men - jo flere regler du lærer dig, des lettere vil det være for dig.

BASIC er det sprog, som er indbygget i din CPC464. Det er der, når du tænder, og giver sig til kende med ordet:

Ready

BASIC er det nemmeste datasprog at lære. Det er opbygget med klart definerede ord og grammatik, og det arbejder helt logisk, når du blot overholder reglerne.

ARMSTRAD BASIC kan udføre de kommandoer, som er beskrevet i kapitel 8. Hver af kommandoerne identificeres af et eller flere nøgleord, og en kommando kan have et antal parametre, hvoraf nogle kan være valgfrie. Generelt kan en parameter være et udtryk, som indeholder konstanter, variabler og funktioner. Kombinationer af tal og bogstaver kaldes for strenge, og forskellige former for taldata kan bruges inklusive decimaltal, hexadecimale tal og binæretal.

Filer på kassetten behandles *sekventielt* d.v.s. en efter en - og ikke *frit*, hvor en fil kan vælges direkte imellem mange andre, uden at man først passerer gennem de uønskede filer.

3.2 Et BASIC programs opbygning

Programinstruktionerne præsenteres for BASIC i linier. En linie kan indeholde flere kommandoer, som adskilles af koloner, kun begrænset af, at linielængden ikke må overstige 255 karakterer. En karakter er et tal, et bogstav, et mellemrum eller et tegn. Når du arbejder i kommandotilstanden, indskriver du linier på tastaturet, og disse må ikke begynde med et linienummer. I programtilstanden indlæses linier fra det eksisterende program i hukommelsen, og disse udføres strengt i rækkefølge efter det første tal på linien.

AMSTRAD BASIC tillader dig at tilføje eller fjerne linier fra programmet og at ændre eksisterende linier, når du arbejder i kommandotilstanden. Inden programmet køres med RUN, eller udskrives med LIST, vil BASICen sørge for, at linierne står i rækkefølge efter linienummer, uanset hvilken rækkefølge linierne blev indlæst i.

3.3 Indlæsning af linier

BASIC accepterer op til 255 karakterer lange liner, som afsluttes af |**ENTER**|. Mens du indtaster en linie, er det muligt at rette den indskrevne linie og at kopiere karakterer fra andre steder på skærmen. Se kapitel 1, sektion 1.2.7.

Alle nøgleord skal adskilles med en separator: Dette kan være mellemrum, en matematisk operator (+,- etc.), eller andre genkendelige karakterer. Dette er, fordi det er tilladt at bruge reserverede ord som dele af variabelnavne. Selvfølgelig kan et reserveret ord ikke være et variabelnavn uden at være *omsluttet* af noget, så datamaten ikke kan genkende ordet.

Nøgleord kan indskrives med store eller små bogstaver.

Kommandoen PRINT kan forkortes til et spørgsmålstegn ?, og når denne form bruges, er separatoren ikke nødvendig. Matematiske operatorer (**+-★/MOD**) adskiller også nøgleordene, så det følgende er tilladt, selv om det ikke anbefales, da det kan føre til dårlige vaner, når man indtaster programmer:

```
for n= 1 to 50:?n:next
```

På samme måde kan en enkel apostrof ' (|**SHIFT**| 7) bruges som erstatning for REM i REM-sætninger.

Flere mellemrum end nødvendig bliver ladt i fred og kan bruges til at formatere programlisten, ved fx at vise løkkers omfang.

3.4 TERMINOLOGI

For at kunne beskrive BASIC kommandoer og nøgleord, må en formel, men simpel terminologi bruges. Hver kommando er beskrevet, som den skal se ud, når den indtastes på tastaturet, med variabler og valgfrie muligheder vist med forskellige typer parenteser. Dette henviser til den efterfølgende beskrivelse.

Disse muligheder er repræsenteret med forskellige navne, og omsluttet af vinkelparenteser. Fx er der nogle steder brug for et tal. Dette vises således:

«numerisk udtryk»

Nogle kommandoer har ikke brug for parametre. Dette kan fx være STOP kommanoden, som har formen:

STOP

Hvor der er valgfrie muligheder i en definition, er den valgfrie mulighed omsluttet af firkantede parenteser. Hvis fx et numerisk udtryk er valgfrit, vil det se således ud:

|«numerisk udtryk»|

Hvis en valgfri del kan gentages (så den kan bruges et antal gange, inkl. ingen gange), vil en stjerne ★ stå efter den sidste firkantede parentes:

«ciffer»|«ciffer»|★

En sådant udtryk kan være:

34, eller
344, eller
345678 etc

På mange steder bruges en liste over parametre adskilt med kommaer. En kort form, som bedst illustreres med et eksempel, bruges:

«liste over:«udtryk» betyder: «udtryk»,|,«udtryk»|★ eller:
«liste over:|#|«tal»betyder: |#|«tal»|,|#|«tal»|★

et eksempel på dette er:

3,4 eller
3,4,4 eller
3,4,5,6,7,8 etc.

listen kan være et enkelt objekt. Hvis listen indeholder mere end et objekt, må hvert ekstra objekt foregås af et komma, eftersom dette angiver grænsen mellem objekter, som datamaten skal behandle enkeltvis.

Tal kan udtrykkes på flere måder:

a. «uskalerede tal» er tal uden tierpotens.

b. «skalerede tal» er tal, som skal opløftes til en potens, på formen.

2E4 (2 gange 10 opløftet til fjerde potens). Eksponenten kan være enten positiv eller negativ.

c. «tal fra andre talsystemer» er tal, som enten er binære eller hexadecimale. Se Appendiks II

Decimal form	100
Hexadecimal form	&64 eller &H64 .....(H er valgfrit)
Binær form	&X1100100 .....(X er tvunget)

3.5 Øvelse gør mester - udskrivning med PRINT

For at demonstrere, hvordan denne terminologi virker, er her nogle eksempler på brugen af BASIC.

En BASIC kommando, som bruger det meste af terminologien, er PRINT kommandoen. En kommando er et BASIC nøgleord eller en BASIC sætning, der kan bruges enten som direkte kommando eller som program. En funktion kræver en kommando for at gå igang fx:

PRINT FRE(''')

For at få CPC464 til at give dig et svar på et spørgsmål, skal du fortælle den tre ting:

1) Hvor du vil have svaret til at stå - på skærmen, på printeren eller andetsteds?

2) Du må give datamaten nogle data at arbejde med.

3) Du må fortælle datamaten, hvad den skal gøre med dataene.

PRINT bruges til at fortælle datamaten, at den skal putte resultatet ind på en bestemt strøm. En strøm kendetegnes med et tal fra 0 til 9, dette tal beskrives i afsnittet om BASICen, som et «strømudtryk». Tallet er *navnet* på den strøm, der skal bruges.

0...7 er tekststrømme til tekstvinduer, der tidligere er sat op med WINDOW kommandoen.

8 er parallelprinterporten og kan kun bruges hvis en Centronics kompatibel printer er tilsluttet korrekt.

9 er en kassette udskrivningsfil, som skal åbnes korrekt på et tidligere tidspunkt i programmet.

En koncis form på PRINT er da (bortset fra PRINT USING):

PRINT |# «strømudtryk»||«udskrivningsliste»|

...indtil videre. De firkantede paranteser betyder, at du ikke behøver at give strømnummeret, og du behøver heller ikke give PRINT kommandoen noget at skrive ud. (Hvis du ikke giver PRINT noget at skrive ud, vil kommandoen blot give en tom linie - prøv det). Hvis du ikke sender udskriften til en bestemt strøm, vil CPC464 antage, at du mener strøm #0. Dette er skærmen lige efter den, hvor markøren står nu. Prøv denne linie (husk at trykke |**ENTER**| når du har skrevet linien, så datamaten ved, at den skal i arbejde):

PRINT ''HALLO''

Datamaten skriver:

HALLO

Bemærk, at citationstegnene fjernes fra udskriften. Citationstegn bruges af BASIC til at se, hvor teksten begynder og slutter.

Indtast nu:

PRINT # 0,"HALLO"

Resultatet er det samme.

Men prøv:

PRINT # 4,"HALLO"

...nu skriver datamaten teksten øverst til venstre på skærmen, fordi det er første ledige plads i strøm nummer 4. Denne dækker hele tekstarealet, medmindre den tidligere er indstillet med WINDOW kommandoen. Startpositionen for al tekst i en skærm-strøm er øverste venstre hjørne. Og strøm 4 er hidtil ubrugt. Den besked, datamaten skriver, når den tændes, bruger strøm 0, så teksten blev sendt til strømmen efter de karakterer, som allerede var der.

Denne mulighed i AMSTRAD BASIC er særligt brugbar, eftersom den tillader opbygningen af komplekse skærmbilleder ved hjælp af simple kommandoer og definitioner, der hjælper en til at opbygge et tydeligt skærmbillede.

BASIC vil udskrive alt, hvad du sætter mellem citationstegn, uden at undersøge, hvad det er. Reserverede ord kan således stå i printlisten.

Prøv:

PRINT "4★4"

datamaten skriver

4★4

For at få datamaten til at udføre multiplikationen 4 gange 4 skal operatoren og tallene stå til rådighed for datamaten, og resultatet skal sendes til en PRINT strøm.

PRINT 4★4

...og BASIC giver svaret

16

Bemærk, at tallet skubbes en kolonne frem fra venstre margen, eftersom BASIC reserverer plads til et eventuelt minustegn, som står foran et negativt resultat.

PRINT kommandoen har mange andre former, idet der kan bruges en fuld formateringsfacilitet til udskrivning af tal.

«print listen» i print kommandoen henviser til en liste over ting, der skal udskrives. Dette kan være tal, en variabel eller et streng udtryk, som er en tidligeret defineret strengvariabel (fx HALLO$), eller alt andet, som omsluttes af citationstegn.

PRINT USING sætter tal ind i et fast format til printning. Kolonner rettes ind, og uønskede rester bortkastes.

3.6 PRINT USING formatet og ZONER

Når du tænder datamaten, vil BASIC sætte skærm zonebredden til 13 kolonner. Når den næste PRINT instruktion bruger et komma , vil næste ting blive tabuleret til næste, nærmeste zone. Hvis der er færre kolonner tilgængelige på en linie end krævet i ZONE-kommandoen, vil BASIC sætte næste udskrift på en ny linie. BASICen vil ikke "knække" en udskrift over kanten på en linie.

Hvis du ikke bruger noget USING format, vil BASIC skrive positive tal ud med et mellemrum foran, og negative tal vil blive udskrevet med et - foran. Alle tal efterfølges af et mellemrum. Decimalpunktum bruges ikke, hvis et tal ingen decimaler har.

AMSTRAD BASIC bruger ikke |**TAB**|-tasten til at tabulere i kolonner - eftersom der hersker stor uenighed om meningen og virkemåden i de forskellige BASIC-dialekter. Hvis du trykker på |**TAB**|-tasten, vil du få en pil pegende til højre udskrevet. Dette er det samme som |**CTRL**| og I-tasten samtidig, udover dette har tasten ingen funktion i AMSTRAD BASIC.

3.7 PRINT TAB («heltalsudtryk») («udskriftsliste»)

Virkningen illustreres bedst med et eksempel. Indtast dette og se resultatet:

```
5 MODE 2: INK 1,0:INK 0,9
10 FOR N=1 TO 5
20 ZONE 40
30 PRINT TAB (N*4) "HEJ",N
40 NEXT
```

Dette viser både ZONE sammen med komma, og TAB funktionen i arbejde. Kør det igen med linie 10 ændret til:

```
10 FOR N=-5 TO 5
```

TAB instruktionen flytter starten af udskriften frem med antallet af mellemrum, som angivet i «heltalsudtryk». (ZONE kan indstilles i området 1 til 255. Se definitionerne i kapitel 8).

PRINT USING formen bruges til at formattere resulterne af beregninger, hvor resultater i reelle tal vil give et uforudsigeligt antal decimaler. Det er en kompliceret facilitet, som bedst forstås gennem praktiske forsøg, eftersom den tekniske form:

PRINT |#«strømudtryk»,||«print liste»||«using format»||«separator»|

retfærdigvis må beskrives som *ikke særlig brugervenlig,* især da «using format» igen kan deles op i:

USING «strengudtryk»;[«using liste»]

hvor «using liste» yderligere inddeles i

«udtryk»[separator»«udtryk»] ★

Prøv følgende:

```
PRINT 123.456, USING "###.##";4567.896
```

resultatet er:

123.456 %4567.90

Dette viser flere ting. For det første viser det, at ting, der bliver udskrevet før USING, bliver ladt i fred. For det andet reserverer USING det antal pladser, der skal bruges til udskriften af det følgende tal (det følgende kan være en variabel). Hvis tallet før punktummet har flere cifre, end der er reserveret plads til, vil tallet alligevel blive udskrevet, men med % foran for at vise, at tallet var for stort. Dernæst viser det sig, at kommaet får udskriften af tallet efter 123.456 til at hoppe frem til næste zone. Hvis kommaet havde været et semikolon, ville tallet have været blevet udskrevet umiddelbart bagefter 123.456, kun med et enkelt mellemrum imellem. Tal adskilles altid med et mellemrum, når de bliver skrevet ud på samme linie - af åbenlyse årsager!

Læg også mærke til, at udtrykket bliver afrundet,idet de tiloversblevne decimaler ikke bare smides væk.

Prøv dette:

```
PRINT 123.456, USING "#####.##+";4567.899
```

og fortegnet kommer bagefter det formatterede tal. Et minus vil automatisk komme foran et negativt tal.

PRINT USING er en meget brugbar facillitet, når man skal lave en eller anden form for tabel. Du vil altid blive advaret, hvis formatet er for restriktivt (med tegnet%).

4 Variabler, operatorer og data

Hvordan man styrer informationerne i et BASIC program

I dette kapitel behandles:

★ Hvordan man træner sig
★ Variabeltyper: Reel, heltal og streng
★ Operatorer, logiske udtryk
★ Indicerede variable
★ DATA

4.1 Kan du se det reserverede ord

Bemærk (hvis du ikke allerede har gjort det), at kommandoer og andre reserverede ord i AMSTRAD BASIC adskilles fra den øvrige del af et program ved mellemrum, tegnsætning, numeriske operatorer etc. Programmerne er lettere at læse og fjerne fejl fra, fordi de altid bliver udskrevet med STORE BOGSTAVER, selv om du har lov at indskrive indskrive nøgleord med STORE BOGSTAVER eller små bogstaver. Hvis en kommando ikke bliver udskrevet med store bogstaver, er det, fordi den ikke er blevet skrevet rigtigt ind, og derfor kører programmet ikke.

AMSTRAD BASIC tillader dig at indbygge nøglord i variabelnavne. En variabel i AMSTRAD BASIC er et navn, som du giver til et bestemt element. Det kan være så simpelt som et bogstav (variabler skal altid starte med et bogstav - ikke med et tal), selv om det er lettere at læse et langt program, hvis du bruger variabelnavne, som viser, hvad der foregår:

```
SVAR=4*4: PRINT SVAR
```

Variabelnavne i AMSTRAD BASIC kan have op til 40 tegn (det første skal være et bogstav), og alle 40 er af betydning. Variabelnavne må ikke indeholde mellemrum, så vil BASIC kun læse de bogstaver (eller tal), som står før det første mellemrum, og derefter skrive:

```
Syntax error
```

Som viser, at der er skrevet en forbudt rækkefølge af karakterer ind (se appendik VIII). Hvis du vil bruge navne med flere ord, så skriv et punktum . hvor du ville have skrevet et mellemrum. Alle de almindelige former for indicerede variable er tilladt, når du blot husker, at en indiceret variabel skal dimensioneres med DIM.

4.2 Genveje

Det er besværligt at skrive PRINT hver gang, så du kan skrive ? i stedet, BASIC vil da forstå, at dette betyder PRINT (så længe det ikke står inde i et udtryk, mellem citationstegn ""). Bemærk, at når du bruger ?, er det ikke nødvendigt at anvende mellemrum sådan som ved PRINT. Hvis du skriver linien som et program og ikke som en direkte kommando:

```
10?4★4
run
```

er svaret fortsat det samme, men prøv at LISTe dette en-linies program og se mirakler:

```
list
10 PRINT 4★4
```

BASIC har også indsat et mellemrum før det ændrede spørgsmålstegnet. Prøv også følgende:

```
10?"HALLO"
```

Du kan også blive doven og droppe det sidste citationstegn i en linie, som du også kan se af den lille **[CTRL]** **[ENTER]** forkortede kassette- og RUN-rutine, som skriver RUN" på skærmen. Det samme virker ved programlinier, men det er ikke nogen god vane at få, for, hvis du senere går tilbage til linien og føjer noget til, vil du sikkert glemme at indsætte det afsluttende citationstegn.

4.3 Flere kommandoer pr. linie og blandede beregninger

Du kan udføre et antal operationer på en enkelt BASIC linie - faktisk så mange du ønsker indtil den maksimale linielængde på 255 karakterer. Som sædvanlig skal sætningerne adskilles af kolon :

```
?2★8/5+5-4★777E9/3
```

giver

```
-1.036E+12
```

Det er meget vigtigt, at du forstår den orden, BASICen følger ved udførelsen af matematiske beregninger (+-★/ MOD etc), ellers vil du begå elementære fejl. Rækkefølgen er:

↑ Potensopløftning - tallet opløstes til en given 10'er potens
- Fortegnsminus (det minus, som bruges foran et negativt tal)
★ Multiplikation
/ Division
/ Heltalsdivison: Resultatet bliver afkortet til heltalsdelen af kvotienten. Resten bortkastes af BASIC
+ Addition
- Subtraktion

Alt det, der omsluttes af parenteser (), behandles allerførst, og hvis indholdet af en parentes i sig selv er en blandet beregning, så vil denne blive behandlet i samme orden, som angivet ovenfor, inkl. eventuelle ekstra parenteser inden i parentesen. Du skal altid have lige mange højre og venstreparanteser i et sådant udtryk, ellers vil du få Syntax error.

4.4 At komme igang

Vi er kommet et stykke vej, siden vi startede med PRINT sætningen tilbage i afsnit 3.5. Du skulle nu have fået tilstrækkeligt med grundlæggende regler for AMSTRAD BASIC til, at du nu skulle kunne dykke ned i noget, der ligner rigtig programmering, i modsætning til de simple pseudo-lommeregner forsøg, vi har foretaget indtil dette øjeblik. BASIC nøgleordene vil blive introduceret, når det er nødvendigt, kig i kapitel 8 for at se den alfabetiske liste og beskrivelse, hvis du ikke forstår brugen ud fra den sammenhæng, vi her har.

Mange BASIC nøgleord har navn efter det, de gør. Fx betyder GOTO 50 *gå til* linie 50 og forsæt programafviklingen derfra. END betyder slutning, og BASIC returnerer til direkte kommandoer med et *Ready,* straks den træffer på et END, også selv om det måtte være i første programlinie.

Når du arbejder med direkte kommandoer, kan du indtaste et antal ordrer ved at adskille de enkelte ordrer med kolon. Men, så snart du udfører linien (ved at trykke |ENTER|, bliver instruktionerne behandlet og linien slettet. Du kan altid bruge linien igen, hvis den stadig er på skærmen, ved hjælp af copy-markøren.

4.5 Betingelser og logiske sætninger

BASIC gør meget brug af datamatens evne til at gøre simple ting meget hurtigt - igen og igen uden at kede sig. Der findes en del kommandoer, som kan hjælpe en med at opstille den slags programmer (løkker): Kommandoer, der starter, udfører og finder ud af, hvornår løkken skal stoppes, hvis en forud opstillet betingelse opfyldes.

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

Nu får du et kort program, der kan vise dig disse operatorer, baseret på et emne, der står vore hjerter nær. Hvis du ikke allerede kigger på datamatens startmelding:

```
Amstrad 64k Microcomputer (v1)
c 1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd

BASIC 1.Ø

Ready
```

...så tryk på |**CTRL**| |**SHIFT**| og |**ESC**| tasterne, idet du holder dem nede samtidig. Datamaten vil da starte helt forfra med ovenstående besked. Fortsæt nu med at indtaste følgende (hvis du skal rette en skrivefejl, kan du læse om det i kap. 1.27):

```
10 INPUT "HVAD TJENER DU";LOEN
20 IF LOEN <100000 THEN GOTO 30 ELSE 40
30 PRINT "BED OM MERE":END
40 PRINT "KOEB EN STOERRE BIL"
```

Kør dette program ved at indtaste:

```
RUN
```

Og selvfølgelig |**ENTER**|. Datamaten spørger dig:

```
HVAD TJENER DU?
```

Bemærk, hvordan datamaten automatisk skriver spørgsmålstegnet, når den vil have dig til at give nogen oplysninger. Svar på spørgsmålet med tal, ikke bogstaver, dollartegn eller kommaer og giv datamaten svaret med |**ENTER**| tasten.

Føj linie 5 nedenfor til ved simpelthen at skrive den ind, når du er færdig med ovenstående program og der igen står Ready på skærmen:

```
5 CLS
```

Kør nu programmet igen, så skærmen bliver slettet. Hvis dit første svar var mindre end 100000, så skriv denne gang et beløb over 100000, så du kan se forskellen. Udvid nu det oprindelige program ved at tilføje linie 50 nedenfor:

```
5 CLS
10 INPUT "HVAD TJENER DU";LOEN
20 IF LOEN <100000 THEN GOTO 30 ELSE 40
30 PRINT "BED OM MERE":END
40 PRINT "KOEB EN STOERRE BIL"
50 IF LOEN >300000 THEN PRINT "OG FIND EN GOD REVISOR"
run
```

Kommandoen END i linie 30 stopper programmet og returnerer til Ready. Eftersom der ikke er nogen END i linie 40, vil datamaten fortsætte med at undersøge, om variablen LOEN er større end 300000, og hvis dette er tilfældet, vil programmet give dig endnu et godt råd!

Fortsæt med at føje en linie 60 til:

```
5 CLS
10 INPUT "HVAD TJENER DU";LOEN
20 IF LOEN <100000 THEN GOTO 30 ELSE 40
30 PRINT "BED OM MERE":END
40 PRINT "KOEB EN STOERRE BIL"
50 IF LOEN >300000 THEN PRINT "OG FIND EN GOD REVISOR"
60 IF LOEN >250000 THEN PRINT "...OG STIK MIG EN HALVTREDSER"
run
```

Læg mærke til at de sammenlignende operationer (< og >) virker som begrænsere, der markerer grænserne på tallene, og at du ikke behøver at skrive et mellemrum før eller efter dem. Hvis du gør det, vil mellemrummet blive sprunget over af BASIC. Hvis du svarer 260000, når du kører dette program, vil du se, at programmet passerer gennem linie 50, som om den ikke var der, for til gengæld at blive fanget af linie 60.

På dette tidspunkt skal du lave et program selv, ved hjælp af de emner, vi har behandlet indtil nu. Læg mærke til den måde dette program er vokset frem på. De fleste programmer udvikler sig på denne måde. Dette introducerer den måske vigtigste tanke bag programmering:

4.6 Udvikling: Programmernes opståen

Den måde, hvorpå BASIC tillader dig at lade programmer vokse efterhånden som du arbejder dig frem, er måske den mest bekvemme detalje ved dette sprog. Folk, der går ind for *ordentlig programmering,* vil hævde at dette fører til rodet og ustruktureret programmering, hvor programmerne er klistret sammen i den rækkefølge, man har fået ideerne. Realister vil anse dette for den bedste måde at fastholde begynderes interesse, da disse derved får mulighed for at gøre fremskridt i små, nemme etaper.

Lad os se på vort eksempel igen. Føj nu en linie 70 til, som får programmet til at starte forfra igen efter at have ventet længe nok til, at du får læst skærmen:

```
5 CLS
10 INPUT"HVAD TJENER DU";LOEN
20 IF LOEN <100000 THEN GOTO 30 ELSE 40
30 PRINT "BED OM MERE":END
40 PRINT "KOEB EN STOERRE BIL"
50 IF LOEN >300000 THEN PRINT "OG FIND EN GOD REVISOR"
60 IF LOEN >250000 THEN PRINT "...OG STIK MIG EN HALVTREDSER"
70 for n=1 to 900: next n:goto 5
run
```

Læg mærke til at den nye linie er skrevet med små bogstaver. Det er for at minde dig om, at AMSTRAD BASIC kan skelne mellem et variabelnavn og et nøgleord. Tryk på **|ESC|** to gange for at komme ud af programkørslen, og udskriv programmet med LIST. Læg mærke til, at datamaten har ændret nøgleordene til store bogstaver, mens variablen n bliver ved med at være med et lille bogstav.

Linie 70 er en pauseløkke, idet datamaten tæller fra 1 til 900, før den udfører den næste sætning -.GOTO 5. Derved starter programmet af sig selv op igen. Den eneste måde, man kan standse programmet, er ved at trykke |**ESC**| to gange på |**ESC**| knappen. Hvis du kun trykker en gang, stopper programmet, trykker du en gang til, kommer du helt ud af programmet, der dog ikke slettes fra datamatens hukommelse.

Medmindre du trykker |**ESC**|, mens programmet er i gang med pausen på linie 70, vil programmet stoppe omgående, da det ikke laver noget, når det venter på indtastninger. Linien, som datamaten ventede i, vil stå lige bagefter ordet Break. Du kan fortsætte med **CONT** fra den linje, hvor programmet stoppede. Du kan ikke miste et program med |**ESC**| - kun med **NEW** eller ved reset med tasterne |**CTRL**| |**SHIFT**| og |**ESC**|.

Derfor skulle det ikke være nødvendigt med nogen indbygget sikkerhed mod utilsigtigt sletning af programmet. Når du sletter et program, er det tabt, det kan ikke reddes. Gem det derfor på kassette, hvis du er tvivl, om du vil bruge det senere.

4.7 Flere variabler, og strenge

Det vigtigste ved databehandling er variabler. Hvis datamaten kun arbejder med tekst, er den ikke andet end en elektrisk skrivemaskine. Husk, at hvis en del af et matematisk udtryk er variabelt, så vil resultatet også være variabelt.

Variabler har tre vigtige træk: Et navn, en type, og en "opbygning". Navnene har vi omtalt tidligere (kap. 4.1) - typerne er valgfrie, så vi kunne definere en variabel med reglerne fra afsnit 3.4 som:

«navn»|**«type markør»**|

Typemarkørerne er:

% for heltalsvariabler, når alt til højre for decimal punktummet skal slettes. Heltal bruger mindre plads i hukommelsen, og derfor kan programmer, der kun behøver heltal arbejde hurtigere, hvis variablerne defineres med DEF som heltal. kommandoen DEFINT sætter variabler til at være heltal. Heltal kan ligge mellem -32768 og +32767.

! Erklærer en variabel til at være reel - hvilket betyder at tallet kan have decimaler. Variabler er normalt reelle, så du har kun brug for at definere reelle variabler, hvis du har brugt DEFINT til at lave dem til heltal. Reelle variabler kan bruges i området fra 2.9E-39 til 1.7E+38.

$ Bruges til at markere strengvariabler, hvor indholdet kan være en blanding af tal og bogstaver. Med andre ord en tilfældig blanding af karakterer, som omsluttes af citationstegn ''''. Fx:

NAVN$ = ''HANS NIELSEN''

Hvis vi bruger dette i vort udviklingseksempel fra før, kan vi fx føje en linie 6 til og ændre linie 60:

```
5 CLS
6 INPUT ''HVAD HEDDER DU'';NAVN$
10 INPUT ''HVAD TJENER DU'';GAGE
20 IF GAGE <100000 THEN GOTO 30 ELSE 40
30 PRINT ''BED OM MERE'':END
40 PRINT ''KOEB EN STOERRE BIL''
50 IF GAGE >300000 THEN PRINT ''OG FIND EN GOD REVISOR''
60 IF GAGE >250000 THEN PRINT ''
...OG STIK MIG EN HALVTREDSER '';NAVN$
70 FOR n=1 to 900: NEXT n:GOTO 5
run
```

Læg mærke til, at der er skrevet et mellemrum efter HALVTREDSER, ellers ville navnet komme til at stå lige bagefter ordet halvtredser. Prøv det, hvis du ikke tror på os! Semikolonnet (;) i slutningen på PRINT og INPUT sætninger får datamaten til at fortsætte på samme linie, næste gang den skal skrive noget ud.

Vi kan også arbejde med heltal ved at føje en linie 61 til. Skriv den blot ind. Datamaten skal nok få den til at stå det rigtige sted:

```
5 CLS
6 INPUT ''HVAD HEDDER DU'';NAVN$
10 INPUT ''HVAD TJENER DU'';GAGE
20 IF GAGE <100000 THEN GOTO 30 ELSE 40
30 PRINT ''BED OM MERE'':END
40 PRINT ''KOEB EN STOERRE BIL''
50 IF GAGE >300000 THEN PRINT ''OG FIND EN GOD REVISOR''
60 IF GAGE >250000 THEN PRINT ''
..OG STIK MIG EN HALVTREDSER '';NAVN$
61 DAGL.GAGE. = GAGE/365: PRINT ''DET ER #'';DAGL.GAGE.;' kr om dagen''
70 FOR n=1 to 5000: NEXT n:GOTO 5
run
```

Bemærk at pausen i linie 70 er blevet udvidet til 5000, nu da der er mere at læse på skærmen. Resultatet af beregningen af dagløn er rodet - du kan lige så godt afrunde det til et heltal. Føj linie 62 til..

```
5 CLS
6 INPUT "HVAD HEDDER DU";NAVN$
10 INPUT "HVAD TJENER DU";GAGE
20 IF GAGE <100000 THEN GOTO 30 ELSE 40
30 PRINT "BED OM MERE":END
40 PRINT "KOEB EN STOERRE BIL"
50 IF GAGE >300000 THEN PRINT "OG FIND EN GOD REVISOR"
60 IF GAGE >250000 THEN PRINT "
...OG STIK MIG EN HALVTREDSER ";NAVN$
61 DAGL.GAGE = GAGE/365:
PRINT "DET ER #";DAGL.GAGE.;" kr om dagen"
62 HELTALS.DAGL.% = DAGL.GAGE.:PRINT
"eller ";HELTALS.DAGL.%;" hvis du ikke er bekymret for klatter"
70 FOR n = 1 to 5000: NEXT n:GOTO 5
```

Kør dette

Bemærk at du skal huske at fortsætte med at bruge typemarkøren%, eftersom det er muligt at have en reel og en heltalsvariabel med samme navn, hvor kun % tegnet er til forskel. Læg også mærke til at datamaten klipper linier over, der er for lange (Det sker med alle tre skærmformater). Brug MODE 2 til at indtaste lange programmer, da det er meget lettere at læse programmer, der ikke hele tiden fortsætter på næste linie.

For at få MODE 2 indtaster du:

```
MODE 2
```

Og for at få sort på hvid skærm, som er lettere at læse på CTM640, indtaster du følgende tre direkte kommandoer:

```
INK 1,0
INK 0,13
BORDER 13
```

Skriv nu programmet ud med LIST igen.

4.8 Hvordan man får skærmen pæn

En del af udviklingsprocessen for et program, er at rydde det op en gang imellem. Det første man kan gøre, er at give linierne nye numre, med RENUM kommandoen. Når datamaten skriver Ready, skriver du:

```
RENUM
```

Og prøv så at LIST'e programmet igen:

```
10 CLS
20 INPUT "HVAD HEDDER DU";NAVN$
30 INPUT "HVAD TJENER DU";GAGE
40 IF GAGE<100000 THEN GOTO 50 ELSE 60
50 PRINT "BED OM MERE":END
60 PRINT "KOEB EN STOERRE BIL"
70 IF GAGE>300000 THEN PRINT "OG FIND EN GOD REVISOR"
80 IF GAGE>250000 THEN PRINT "OG STIK MIG EN HALVTREDSER ";NAVN$
90 DAGL.GAGE = GAGE/365:
PRINT "DET ER ";DAGL.GAGE;"kr om dagen"
100 HELTALS.DAGL% = DAGL.GAGE:PRINT
"eller ";HELTALS.DAGL%;" hvis du ikke er bekymret for klatter"
110 FOR n = 1 to 5000: NEXT n:GOTO 10
```

Alle linienumre er blevet rundet op - incl dem, der bliver henvist til inde i programmet. Det ville ikke være meget værd, hvis BASIC ikke holdt øje med alle linienumrene, og gav nye numre til alle samtidig.

Nu vil vi rydde op i det, der kommer på skærmen, når programmet kører. Det første vi gør, er at sætte løkken på linie 110 ud af kraft, ved at lave den om til en REM-sætning. REM betyder REMark = bemærkning på engelsk.

```
10 CLS
20 INPUT "HVAD HEDDER DU";NAVN$
30 INPUT "HVAD TJENER DU";GAGE
40 IF GAGE<100000 THEN GOTO 50 ELSE 60
50 PRINT "BED OM MERE":END
60 PRINT "KOEB EN STOERRE BIL"
70 IF GAGE >300000 THEN PRINT "OG FIND DIG EN GOD REVISOR"
80 IF GAGE >250000 THEN PRINT "OG STIK MIG EN HALVTREDSER ";NAVN$
90 DAGL.GAGE = GAGE/365:
PRINT "DET ER ";DAGL.GAGE;"kr om dagen"
100 HELTALS.DAGL% = DAGL.GAGE:PRINT
"eller ";HELTALS.DAGL%;
hvis du ikke er bekymret for klatter"
110 REM:FOR n = 1 to 5000: NEXT n:GOTO 10
```

Når man skriver REM foran en linie, vil resten af linien blive opfattet som en kommentar, som BASIC ikke skal tage sig af. BASIC stopper da programmet, istedet for at starte det op pany hele tiden. Indtast nu

```
15 mode 1
```

Linie 15 vil sørge for at skærmen har 40 karakterer pr linie, når dette program kores. Når man udfører MODE kommandoen, vil skærmen altid blive slettet. Linie 10 er da unodvendig, men vi lader den blive alligevel. Kør nu programmet, og svar:

```
HVAD HEDDER DU? HANS
HVAD TJENER DU? 400000
KOEB EN STOERRE BIL
OG FIND EN GOD REVISOR OG STIK MIG EN HALVTREDSER HANS
DET ER 109.589041 kr om dagen
eller 110
hvis du ikke er bekymret for klatter
```

Det er ikke særlig kønt:

Tilføj..

```
25 PRINT: PRINT
```

```
85 PRINT
```

Og ret linie 100 til:

```
100 HELTALS.DAGL% = DAGL.GAGE:PRINT
"eller ";HELTALS.DAGL%;"Hvis du ikke":PRINT
"er bekymret for klatter"
```

Kør dette program, og du vil se at datamaten har skrevet "Hvis du ikke" på en linie ovenfor. Tilføj linie 120 for at få Ready beskeden endnu længere ned:

```
120 ?:?:?:?:?
```

Kør så programmet, eller fjern Ready helt, ved at indtaste:

```
120 GOTO 120
```

Når programmet nu kører, kan du kun stoppe det med |**ESC**|. Husk at ? er en forkortelse for PRINT. Prøv nu LIST:

```
10 CLS
20 INPUT "HVAD HEDDER DU";NAVN$
25 PRINT: PRINT
30 INPUT "HVAD TJENER DU";GAGE
40 IF GAGE<100000 THEN GOTO 50 ELSE 60
50 PRINT "BED OM MERE":END
60 PRINT "KOEB EN STOERRE BIL"
70 IF GAGE >300000 THEN PRINT "OG FIND EN GOD REVISOR"
80 IF GAGE >250000 THEN PRINT "OG STIK MIG EN HALVTREDSER ",NAVN$
85 PRINT
90 DAGL.GAGE = GAGE/365:
PRINT "DET ER ";DAGL.GAGE;"kr om dagen"
100 HELTALS.DAGL% = DAGL.GAGE:PRINT
"eller ";HELTALS.DAGL%;" hvis du ikke":PRINT
"er bekymret for klatter"
110 REM:FOR n = 1 to 5000: NEXT n:GOTO 10
120 GOTO 120
```

4.9 LOCATE

Indtil nu har du skrevet de fleste BASIC kommandoer ved hjælp af den universale BASIC grammatik, der kan forstås af de fleste maskiner, som bare taler BASIC. LOCATE er speciel for AMSTRAD BASIC (og flere andre), og bruges til at place tekstmarkøren et bestemt sted på skærmen:

LOCATE 10,4

Placerer tekst markøren i kolonne 10, 4 linier fra toppen af skærmen. Hvis du prøver at gøre dette som en direkte kommando, vil markøren flytte sig rigtigt nok, men beskeden Ready vil gøre at en ny linie bliver startet, således at markøren ender på begyndelsen af en ny linie igen. Tilføj linie 16:

```
16 LOCATE 10,4
run
```

Læg mærke til at det første spørgsmål starter som beskrevet. Næste linie ser ud som den plejer, helt ude i venstre side af skærmen, fordi linie 25 indsætter et par tomme linier. Tryk |**ESC**| to gange, og føj linie 26 til:

26 LOCATE 10,4

Kør programmet (med RUN). Nu ligger andet spørgsmål oveni det første. Nu kan du fortsætte med at placere teksten lige hvor du vil have det, ved at bruge koordinaterne på planlægningsblokken, som du finder i appendiks 6.

Hvis du vil have alle spørsmål og svar til at komme på samme linie, så skal du skrive CLS: før hver LOCATE kommando. På den måde bliver der ikke nogen rester tilbage fra en gammel linie, når den nye skrives.

Bemærk at koordinaterne til LOCATE kan være variabler, således at programmet selv styrer placeringen af teksten. Nu har vi slidt løn programmet op, så næste eksempel vil bygge på et andet emne. Hvis du vil gemme hvad du nu har lavet, så skal du gøre det med kassette kommandoerne i kapitel 2. Det kan være du ligefrem synes om programmet nu, så det kan være du vil gemme det, til at sammenligne med, og til senere udvidelse, med de kommandoer, som vi nu vil vise dig:

4.10 IF ... THEN

Denne kommandos brug er ligefrem, og præcis som den skrives. IF «logisk udtryk» THEN GOTO «linienummer» er en af mange former på denne kommando. På dansk ville den hedde HVIS.... SÅ.

IF checker om resultatet af det «logiske udtryk» er sandt - i så fald bliver det, der står efter THEN udført. IF kommandoen kan bygges ind i løkker, der gentager sig selv (rekursive løkker). Nulstil datamaten ved at trykke |**CTRL**| |**SHIFT**| |**ESC**| og indtast:

```
1 MODE 1
10 AMSTRAD=0
20 PRINT "AMSTRAD CPC464 personlig farve
datamat"
30 AMSTRAD=AMSTRAD +1
40 IF AMSTRAD <10 THEN GOTO 20
```

Kør dette, og se at PRINT sætningen i linie 20 bliver gentaget indtil betingelsen i linie 40 bliver opfyldt. Derved "løkker" linie 40 tilbage til linie 20. Kan du se fornuften i udtrykket variabel, når værdien af AMSTRAD skifter hvergang programmet gennemløber løkken.

Hvis du vil se hvad der sker med værdien af AMSTRAD, i løbet af dette program, så kan vi indføre en linie 35.

```
35 LOCATE 1,2: PRINT AMSTRAD:LOCATE 1,AMSTRAD
run
```

Hvis det gik for hurtigt, kan du sætte farten ned ved at indsætte en pauseløkke:

```
36 for n = 1 to 500:next
```

Indfør nu lidt farve (hvis du har en farvemonitor), ved at indtaste:

```
34 BORDER AMSTRAD
```

Denne linie skifter farven på kanten af skærmen, så den sættes til værdien af AMSTRAD med linie 30. Udskriv programmet igen:

```
1 MODE 1
10 AMSTRAD = 0
20 PRINT "AMSTRAD CPC464 personlig farve
datamat"
30 AMSTRAD = AMSTRAD + 1
34 BORDER AMSTRAD
35 LOCATE 1,20:PRINT AMSTRAD:LOCATE 1,AMSTRAD
36 FOR n = 1 TO 500:NEXT
40 IF AMSTRAD <10 GOTO 20
run
```

Du vil selvfølgelig gerne se alle de farver, som AMSTRADen kan lave, så lav linie 40 om til:

```
40 IF AMSTRAD<26 GOTO 20
```

RUN programmet, og du vil se alle de tilgængelige farver startende med den mørkeste og sluttende med den lyse hvide. Du kan lave beskeden om kantfarven mere nyttig ved at tilføje ordet "kantfarve" i linie 35:

```
35 LOCATE 1,20:PRINT "kantfarve ";AMSTRAD:
LOCATE 1,AMSTRAD
run
```

Når vi nu er ved kanter, så prøv følgende, når programmet er færdigt, og har skrevet Ready:

```
BORDER 14,6
```

Du vil nu se kanten skifte mellem farverne 14 og 6. Du bliver nødt til at vente til næste kapitel, med at få mere forklaring på hvordan grafik og farver virker. For at afslutte dette afsnit, kan du indtaste:

ink 1,18,16

og derefter..

speed ink 1,5

Find nu en hovedpinepille, og sluk datamaten. Husk at SAVE programmet, hvis du vil imponere dine venner.

4.11 Indicerede variable

Nogle programmer kræver en masse lagerpladser, for at kunne virke. Dette er også meget godt, men det er sommetider svært at holde styr på hvilke variabler, der skal bruges hvor. BASIC klarer denne situation ved hjælp af indicerede variable (eller "vektorer","matricer").

Hvad er en indiceret variabel? Spørger du. Det er grundlæggende set en gruppe af variabler, som alle arbejder under det samme navn.

Den bedste måde at forklare dette, er at tage et eksempel. Se på et program, der efterligner et kortspil, og bl.a. med omdeling af tilfældige kort. Det samme kort må ikke blive delt ud to gange, så derfor må vi huske hvilke kort, der allerede er delt ud, og hvilke, der endnu mangler. Den nemmeste måde at gøre dette, er at give hvert kort en variabel, og så give variablen en værdi efter placeringen af kortet. Fx 1 for at vise at kortet er delt ud, og nul til at vise at kortet stadig ligger i bunken.

Dette vil give 52 variabler (med mindre du bruger et sæt snydekort!), og du skal kunne huske hvilken variabel, der hører til hvilket kort. Det er her den indicerede variabel kan være nyttig.

Først skal vi give variablen et navn, fx STABEL. For nu at få adgang til et bestemt element i variablen, skal vi bare give den et nummer. Så hvis de første tretten elementer er hjerter, så er sekseren repræsenteret med STABEL(6), tieren med STABEL(10) og kongen med STABEL(13). Kan du se logikken?

Du kan ikke, desværre, bare blive ved med at bruge nye elementer, uden at give datamaten advarsel om hvor mange du vil bruge, så den kan reservere plads. Kommandoen DIM bruges til dette.

Denne kommando DIMensionerer den indicerede variabel, d.v.s. sætter dens størrelse op. I vores tilfælde skal vi bruge 52 kort. Ordren lyder:

DIM STABEL(52)

Dette fortæller datamaten at den skal reservere plads i hukommelsen, til 52 variabler (53 i virkeligheden, da element 0 også findes).

Nu kan vi skrive en subrutine, der kan uddele kort:

```
10 DIM STABEL(52)
20 FOR X = 1 TO 52
30 LET STABEL(X) = 0
40 NEXT X
....
1000 KORT = RND(52) + 1
1010 IF STABEL(KORT) = 1 THEN GOTO 1000
1020 STABEL(K0RT) = 1
1030 RETURN
```

Bemærk at DIM sætningen er den første i programmet. Dette skyldes at et array kun kan dimensioneres en gang. Det kan ikke laves om i løbet af programmet.

Linierne 20 til 40 sætter alle elementerne i variablen STABEL til 0. Subrutinen, der starter i linie 1000 vælger da et tilfældigt kort ud, og checker at det ikke allerede er delt ud. Hvis det er delt ud, så vælges et andet, indtil der findes et, som stadig er i bunken. Rutinen ændrer da værdien af det rigtige element i variablen, så det kan ses at kortet nu er blevet delt ud, og derefter returnerer programmet fra subrutinen.

Indicerede variabler kan godt have mere end en dimension. Dette gøres simpelthen ved at føje flere tal til i DIM sætningen. Fx kan en 10★10★10 variabel (eller matrice) sættes op med:

```
DIM MATRICE(10,10,10)
```

Denne teknik er nyttig til at dele data ind i mindre "sæt" indenfor en gruppe. I vort eksempel kan vi dele STABEL ind i fire sæt, hver på tretten kort, til at repræsentere hver farve:

```
DIM STABEL(4,13)
```

Hvis vi nu vil have fat i klør fire, som kunne have været element 43 i vor oprindelige matrice, skal vi blot kigge på element (2,4), idet vi går ud fra at klør er anden række i vor nye matrice.

Matricer behøver ikke kun at blive brugt til at gemme tal-data, de kan også bruges til at styre strenge. En anvendelse af dette kunne være at gemme navnene på nogle mennesker, som har bestil billetter til et teater eller et fly.

4.12 DATA

Denne kommando arbejder sammen med kommandoen READ, og kan bruges til at putte data ind i et program. De nødvendige data skrives på en linie med hvert element adskilt fra de andre, med et komma. Foran hele listen skal der stå en DATA kommando. DATAene kan nu bruges i rækkefølge, med en READ sætning.

Et eksempel er:

```
10 READ X,Y,Z
20 PRINT X;" + ";Y;" + ";Z;"  =  ";X + Y + Z
30 DATA 1,3,5
```

DATAene kan være enten numeriske, strenge eller en blanding af begge. Du behøver ikke være bange for at dataene skal fylde mere end en linie, du skal bare starte forfra med en ny linie med DATA kommandoen i begyndelsen. Når datamaten møder en READ sætning, leder den efter DATA sætninger i programmet, uanset hvor de står. Vær sikker på at du har DATA nok til dine READ sætninger, ellers vil der opstå fejl.

Den eneste måde du kan forstyrre datamatens indlæsning i rækkefølge, er at bruge RESTORE kommandoen. Dette sætter "vejskiltet" til at pege på begyndelsen af dataene igen, således at de samme data kan læses flere gange, hvis det er nødvendigt. Dette program eksempel viser brugen af READ,DATA og RESTORE kommandoerne:

```
10 FOR C = 1 TO 5
20 READ X$
30 PRINT X$;" ";
40 NEXT C
50 RESTORE
60 GOTO 10
70 DATA DAVS,HVORDAN,HAR,DU,DET
```

Tryk |**ESC**| to gange for at stoppe programmet.

Læg mærke til at selvom DATAlinien står allersidst i programmet, så kan den stå lige hvor det passer dig.

DATA kommandoen behøver ikke at blive brugt til data, der skal skrives ud. Tal i datasætninger kan fx læses ind i lyd kommandoen SOUND. Indtast:

```
10 FOR n = 1 TO 30
20 READ s
30 SOUND 1,s,40,5
40 NEXT n
50 DATA 100,90,100,110,120,110,100,0
60 DATA 130,120,110,0,120,110,100,0
70 DATA 100,90,100,110,120,110,100,0
80 DATA 130,0,100,0,120,150
```

Hvis du ikke kan høre noget, så skru op for volumenkontrollen i højre ende af datamaten.

For at afslutte denne korte BASIC introduktion, er her et program, der gør det muligt for dig at spille blackjack på CPC464. Det viser brugen af mange ting i BASIC, og det skulle kunne forstås, da variablerne har fået navne, der siger hvad variablerne er. Du kan kvikke det op med lyd og grafik, og du kan udvikle programmet videre, sådan som BASIC programmer kan, fra et simpelt skelet.

Spillets mål er at komme så tæt på en total på 21, ved at lægge kortværdierne sammen, og derefter for "huset" at forsøge at komme dette nærmest eller over, uden at overstige 21. Når du har indlæst linie 1, så brug AUTO til at lave resten af linienumrene automatisk.

```
1 REM BLACKJACK
10 REM BEGYNDELSE
20 YC = 2:CC = 2
30 ESSER = 0
40 CESSER = 0
50 S = 0
60 T = 0
70 DIM HUS$(4)
80 HUS$(1) = "KLØR"
90 HUS$(2) = "HJERTER"
100 HUS$(3) = "SPAR"
110 HUS$(4) = "RUDER"
120 CLS
130 DIM STABEL(52)
140 FOR X = L TO 52
150 STABEL(X) = 0
160 NEXT X
170 REM UDDEL TO KORT TIL HVER SPILLER
180 LOCATE 10,3
190 PRINT"DIG";SPC(15);"HUSET"
200 LOCATE 3,5
210 GOSUB 740
220 S = S + F
230 IF F = 11 THEN ESSER = ESSER + 1
240 LOCATE 3,6
250 GOSUB 740
260 S = S + F
270 IF F = 11 THEN ESSER = ESSER + 1
280 LOCATE 24,5
290 GOSUB 740
300 T = T + F
310 IF F = 11 THEN CESSER = CESSER + 1
320 LOCATE 24,6
330 GOSUB 740
340 T = T + F
350 IF F = 11 THEN CESSER = CESSER + 1
360 REM INDTAST MULIGHED VEND (V) ELLER STÅ (S)
370 X$ = INKEY$:IFX$<>"S" AND X$<>"V" THEN 370
380 IF X$ = "S" THEN 560
390 LOCATE 3,YC + 5
400 YC = YC + 1
```

```
410 GOSUB 740
420 S = S + F
430 IF F = 11 THEN ESSER = ESSER + 1
440 REM CHECK POINT OG ESSER
450 IF S<22 THEN 370
460 IF ESSER = 0 THEN 500
470 ESSER = ESSER-1
480 S = S-10
490 GOTO 450
500 LOCATE 12,19
510 PRINT "DU HAR TABT"
520 PRINT:PRINT"IGEN? (J/N)"
530 X$ = INKEY$:IF X$<> "J" AND X$<>"N" THEN 530
540 IF X$ = "J" THEN RUN
550 END
560 IF T>16 THEN GOTO 700
570 CC = CC + 1
580 LOCATE 24,CC + 4
590 GOSUB 740
600 T = T + F
610 IF F = 11 THEN CESSER = CESSER-1
620 IF T<21 THEN 670
630 IF CESSER = 0 THEN 670
640 CESSER = CESSER-1
650 T = T-10
660 GOTO 620
670 LOCATE 12,19
680 PRINT "DU VANDT"
690 GOTO 520
700 LOCATE 12,19
710 IF T<S THEN 680
720 PRINT "HUSET VINDER"
730 GOTO 520
740 REM DEL KORT
750 LET KORT = INT ( RND(1)*52 + 1)
760 IF STABEL(KORT) = 1 THEN GOTO 750
770 STABEL(KORT) = 1
780 F = KORT-13*INT(KORT/13)
790 IF F = 0 THEN F = 13
800 IF F = 1 OR F>10 THEN GOTO 850
810 PRINT F;" AF ";
820 IF F>10 THEN F = 10
830 PRINT HUS$(INT((KORT-1)/13) + 1)
840 RETURN
850 IF F = 11 THEN PRINT "KNAEGT AF ";
860 IF F = 12 THEN PRINT "DAME AF ";
870 IF F = 13 THEN PRINT "KONGE AF ";
880 IF F<>1 THEN GOTO 820
890 F = 11
900 PRINT "ES AF ";
910 GOTO 830
```

Tryk V for at vende et kort, eller S for at stå, og spille mod huset. Vi påstår ikke at dette er det sidste ord inden for kortspil på CPC464, men det vil give dig lidt at rode med og udbygge med grafik og lyd.

4.13 Logiske udtryk

En stor forskel mellem en datamat og en lommeregner er datamatens evne til at behandle udtryk, som fx dem man bruger i IF.. THEN sætninger. For at kunne dette, behandler logik operatorerne de tal, som de bliver sat til at arbejde på, som binære tal, hvor operatoren så arbejder på de enkelte bits. Beskrivelsen er helt logisk (hrmm), men det er ikke særlig svært at beskrive logikken i simple vendinger.

De to halvdele af et logisk udtryk betegnes "argumenter". Et logisk udtryk indeholder:

«argument|«logisk operator»|

hvor:

«argument» er: NOT «argument»
eller: «numerisk udtryk»
eller: «sammenlignende udtryk»
eller: «logisk udtryk»

Begge argumenter til en logisk operator bliver lavet om til heltal, og Error 6 bliver resultatet, hvis et argument ikke kan passe ind i heltalsområdet.

De logiske operatorer (i rækkefølge efter udførelses rækkefølge), og deres virkning på hver bit er:

AND Resultatet er 0 medmindre begge argument bits er 1
OR Resultatet er 1 medmindre begge argument bits er 0
XOR Resultatet er 1 medmindre begge argument bits er ens

AND er den mest brugte af disse, og den betyder ikke "læg sammen"

PRINT 10 AND 10

Giver svaret 10.

PRINT 10 AND 12

Giver svaret 8.

PRINT 10 AND 1000

Giver svaret 8 igen.

Det er fordi tallene 10 og 100 er omformet til binær representation:

```
      1010
1111101000
```

Operationen AND checker to bits af gangen, og når biten både foroven og forneden er 1, er resultatet 1:

0000001000

..som ved omformning til decimaltal giver resultatet 8. Alt dette betyder at den logiske operator AND bruges til at undersøge om to betingelser er tilstede samtidig. Her er et selvforklarende eksempel:

```
10 CLS:INPUT "DAGENS NUMMER";DAG
20 INPUT "MAANEDENS NUMMER";MAANED
30 IF DAG=24 AND MAANED=12 GOTO 50
40 GOTO 10
50 PRINT "GOD JUL"
```

OR virker også på bits, hvor resultatet er 1 medmindre begge bits fra argumentet er 0, så er resultatet 0. Hvis vi bruger samme tal som til AND eksemplet:

```
PRINT 1000 OR 10
1002
```

Bit vis:

```
      1010
1111101000
```
giver svaret:

```
1111100010
```

Og i program eksemplet:

```
10 CLS:
20 INPUT "MAANEDENS NUMMER";MAANED
30 IF MAANED=12 OR MAANED=1 OR MAANED=2 GOTO 50
40 CLS:GOTO 10
50 PRINT "DET MAA VAERE VINTER"
```

```
10 CLS
20 INPUT "MAANEDENS NUMMER";MAANED
30 IF NOT (MAANED=6 OR MAANED=7 OR MAANED=8) GOTO 50
40 CLS: GOTO 10
50 PRINT "DET KAN IKKE VAERE SOMMER"
```

Den sidste vigtige ting at overveje, er det faktum, at du kan sætte et antal logiske betingelser sammen for at gøre betingelsen endnu sværere:

```
10 CLS:INPUT "DAGENS NUMMER";DAG
20 INPUT "MAANEDENS NUMMER";MAANED
30 IF NOT(MAANED=12 OR MAANED=1) AND DAG=29 GOTO 50
40 CLS:GOTO 10
50 PRINT "DETTE ER HVERKEN DECEMBER ELLER
JANUAR, MEN DETTE KAN VÆRE ET SKUDAAR"
```

Resultatet af en sammenlignende operation er enten -1 eller 0. Bit repræsentationen for -1, er alle bits af heltallet 1; For 0 alle bits af heltallet 0. Resultatet af den logiske behandling af sådan to argumenter vil give enten -1 (for sandt), eller 0 for falsk.
Undersøg dette ved at føje linie 60 til ovenstående program.

```
60 PRINT NOT(MAANED=12 OR MAANED=1)
70 PRINT (MAANED=12 OR MAANED=1)
```

Og når programmet kører, kan du fx indtaste 29 for dagen, og 5 for måneden, dette vil give svaret i linie 50, og de logiske resultater vil linierne 60 og 70 udskrive nedenunder.

Endelig XOR (eXclusive OR) giver resultater der er sande, så længe begge paramtre er forskellige. Det følgende viser alle disse muligheder i en ''sandhedstabel''; som er en nem måde at illustrere hvad der sker i en bitvis logisk operation.

Argument A Argument B	1 0	0 1	1 1	0 0
AND result OR result XOR result	0 1 1	0 1 1	1 1 0	0 0 0

5 GRAFIK KURSUS

Hvordan man finder vej i grafikken og farverne på CPC464:

Emner, der behandles i dette kapitel:

★ Skærmtilstande og punktgrafik
★ Farverne
★ INK,PAPER og PEN
★ At tegne linier
★ Vinduer

5.1 Muligheder, der knytter sig specielt til CPC464

Beskrivelsen og mulighederne i AMSTRAD BASIC CPC464 har indtil videre bygget på en industri standard. Det meste af det aritmetriske kan køre på næsten alle BASIC datamater. De BASIC kommandoer, der styrer grafikken (og tekstmarkøren), er derimod specielle for CPC464.

Som sædvanlig vises ordrerne med de specielle datamat-skærms skrifter, yderligere eksempler og en præcis gennemgang af kommandoerne findes i Kapitel 8.

5.1.1. Farveudvælgelse

Sort, d.v.s. ingen farve opfattes som en farve i alle følgende beskrivelser af styringen af farverne.

Kanten af skærmen kan sættes til to af alle de tilgængelige farver, uanset hvilken indstilling skærmen har. De bliver ikke ændret af MODE kommandoen. De kan sættes til at blinke, eller til en enkelt fast farve.

Antallet af tilgængelige farver, der kan vises samtidig er afhængigt af hvilken MODE skærmen er i. Hver INK (blæk) kan sættes til to farver, d.v.s. til at blinke, eller til en enkelt, d.v.s. konstant farve. Tekst "papiret",tekst "pennen" og grafik "pennen" kan så sættes til en ledig farve.

5.1.2 Gennemsigtig skrift og forholdet mellem PEN, INK og PAPER.

Undtagen i situationer hvor der er specificeret blinkende farver, bruges der to blækfarver (INK), når der skrives på skærmen; den ene bestemmer farven på pennen, mens den anden bestemmer farven på papiret.

NB. Tallet, som forbindes med PAPER kommandoen, er den INK farve, som er givet til det tal, IKKE den farve som står skrevet i appendiks VI. På samme måde er tallet, som forbindes med PEN kommandoen, den INK farve som er givet til den PEN, og ikke den farve, som står i appendiks VI.

PAPER tallet bliver sat til 0, hvis det ikke angives, mens PEN tallet bliver sat til en hvis det ikke bliver givet. For at sætte INK for PAPER nummer 0 til grøn, som er farve 9, skal du indtaste:

INK 0,9

På samme måde skal du gøre, hvis du vil sætte INK farve for PEN nummer 1 til sort, som er farve nummer 0:

INK 1,0

Hvis du sætter PAPER til samme INK som PEN, fx INK 0,0, vil skærmen blive helt sort.

Teksten kan sættes til at være gennemsigtig (markøren er altid gennemsigtig), eller mat, idet du bruger en af de kontrolkarakterer, som findes udover BASIC grafikkommandoerne. Når karaktererne er sat til at være transparente, kan du enten ignorere papirfarven og skrive oveni grafikken, eller du kan skrive oveni baggrunden. Dette korte program illustrerer virkningen:

|CTRL| |SHIFT| |ESC|

```
10 MODE 1
20 INK 2,19
30 DRAW 200,200,2
40 LOCATE 1,21
50 PRINT "1 NORMAL"
60 PRINT CHR$(22)+CHR$(1)
70 ORIGIN 0,0
80 DRAW 500,200,2
90 LOCATE 12,18
100 PRINT "2 GENNEMSIGTIG"
110 PRINT CHR$(22)+CHR$(0)
120 LOCATE 22,15
130 PRINT "3 NORMAL IGEN"
```

Den første DRAW kommando, i linie 30, blev udført før datamaten blev sat til at skrive gennemsigtigt, i linie 60. Anden DRAW kommando kom efter CHR$(22) + CHR$(1), som kommanderer datamaten til transparent skrift. Læg mærke til hvordan de overlappende punkter har skiftet farve, og hvordan (i transparent tilstanden) INKens farve i karakterens "celle" er blevet overskrevet helt.

Prøv at bytte om på linie 60 og linie 110, og se hvordan dette påvirker skærmbilledet. En fuld oversigt over disse ekstra kommandoer findes i appendiks VI.

5.2 Skærm MODE

Skærmen kan indstilles til tre forskellige formater. Tekst og grafik virker forskelligt i disse MODEs:

a) Normal

MODE 1: 40 kolonner ★ 25 linier, 4 blæk (INK) farver til tekst 320 ★ 200 punkter, som hver kan vælge farve ud af 4 mulige

b) Mangefarve MODE

MODE 0: 20 kolonner ★ 25 linier, 16 blæk (INK) farver til tekst 160 ★ 200 punkter, som hver kan vælge farve ud af 16 mulige

c) Højopløsnings MODE

MODE 2: 80 kolonner ★ 25 linier, 2 blæk (INK) farver til tekst 640 ★ 200 punkter, som hver kan vælge farve ud af 2 mulige

Som du kan se, er forskellen at føle i antallet af vandrette "elementer" på skærmen. Du må ikke blande disse sammen med de små linier på TV skærmen, som er noget indbygget i selve TVet (eller monitoren).

Hver af de tre forskellige MODEs bliver styret af kommandoen MODE, og kun en MODE kan bruges af gangen når du bruger BASIC. Når du skifter MODE slettes skærmen - incl al tekst og grafik (samme virkning som CLS og CLG kommandoen), men selve programmet lades uberørt.

Du kan skifte MODE med en BASIC kommando, fra et program eller som direkte kommando.

5.2.1 MODE 0 er den mangefarvede grafik skærm.

16 af de 27 tilgængelige farver kan vises samtidig, og hvert individuelt element på skærmen kan programmes til sin egen farve. Der er 160 punkter vandret og 200 punkter lodret på skærmen. En oversigt over dette gitter findes i Appendiks VI.

I MODE 0 er der 20 karakterer på hver af 25 linier.

5.2.2 MODE 1 er standard MODE.

MODE 1 er sat til at komme frem automatisk, når du tænder for CPC464. 4 af de 27 farver kan bruges samtidig, selvom du kan skifte hurtigt igennem alle 27, hvis du vil. Skærmen er 320 punkter bred og 200 punkter høj. En oversigt findes i appendiks VI.

I MODE 1 er der 40 karakterer på hver af 25 linier.

5.2.3 MODE 2 er højopløsnings MODE

MODE 2 tillader to farver at blive vist samtidig, og denne bruges primært til at vise 80 karakterer pr. linie - dette gør et program meget lettere at skrive, eftersom du kan se meget mere af programmet med et enkelt blik.

MODE 2 giver 640 punkter vandret og 200 punkter lodret.

5.2.4 Prøv dette

Med CPC464 helt tom |CTRL| |SHIFT| |ESC|, prøv dette program:

```
5 REM GRAFIK DEMONSTRTION
10 MODE 1
15 INK 2,0
16 INK 3,6: REM SAETTER FARVEN SOM BRUGES
I LINIE 90
17 BORDER 1: REM MOERK BLAA
20 CLG: REM RYD SKAERMEN
30 B% = RND★5 + 1 :REM SAET TILFAELDIGE HELTAL OP
40 C% = RND★5 + 1
50 ORIGIN 320,200 :REM PLACER GRAFIK UDGANGSPUNKT
60 FOR A = 0 TO 1000 STEP PI/30
70 X% = 100*COS(A)
80 MOVE X%,X% :REM FLYT GRAFIK MARKOER
90 DRAW 200★COS(A/B%),200★SIN(A/C%),3
:REM TEGN LINIE
91 IF INKEY$<>"" THEN 20
100 NEXT: REM TILBAGE TIL 60
MEDMINDRE AFBRUDT I 91
110 GOTO 20
```

Kør nu dette program. Tryk på en tast, for at få et andet mønster. Dette viser flere vigtige muligheder i CPC464: CPC464 skriver jævnt på skærmen uden skarpe kanter, og programmellet indeholder kommandoer, som kan få meget avancerede virkninger frem uden meget arbejde. REM sætningerne bruges kun til at hjælpe dig, du behøver ikke at skrive dem ind, for at programmet skal virke.

Læg mærke til at flere af linienumrene viser at linierne er sat ind senere. Vi kunne selvfølgelig have ryddet listen op med RENUM kommandoen, men det kan bruges til at vise dig, hvordan programmer udvikles udfra et oprindeligt skelet, hvis vi lader numrene være. Du kan jo selv skrive RENUM..

Gem dette program på kassette fx:

SAVE "GRAFIK 5.5.84"

Denne grafik demonstration viser tegninger i forskelligt farvede interferens mønstre.

```
new
10 A$ = INKEY$:REM TRYK PAA EN TAST FOR AT
STARTE ET NYT MOENSTER
20 IF A$ = '''' THEN 10
30 CLS
40 M = INT(RND * 3):REM VAELG ET TILFAELDIGT
TAL MELLEM 0 OG 3
50 IF M>THEN 40: REM PROEV IGEN HVIS TALLET
ER STØRRE END 3
60 MODE M
70 I1 = RND * 26: REM VAELG EN TILFÆLDIG INK FARVE
80 I2 = RND * 26
90 IF ABS(I1-I2)«5 THEN 70
100 INK 0,I1:INK 1,I2
110 S = RND * 5 + 3
120 ORIGIN 320,-100
130 FOR X = -1000 TO 0 STEP S
140 MOVE 0,0
170 DRAW -X,300: DRAW 0,600
180 A$ = INKEY$
190 IF A$>>''''THEN 30: REM FORSTYR LOEKKEN VED
AT TRYKKE PAA EN TAST
200 NEXT X
210 GOTO 10
```

Dette og det foregående program viser simple matematiske metoder på en farverig og visuel måde. Begge laver grundlæggende set summer af tilfældige tal for at sikre at hvert mønster af forskelligt på en eller anden måde.

Din CPC464 er fint elektronisk grafpapir, og en af klassiske geomtriske former er sinusfunktionen:

```
10 REM TEGN SINUSBOELGE
20 MODE 2
30 INK 1,21
40 INK 0,0
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR N = 0 TO 720
90 Y = SIN(N)
100 PLOT N * 640/720,198*Y,1
110 NEXT
```

PLOT-sætningen i linie 100 er den del af programmet som tegner funktionen. Den laver et punkt for hver beregning der gøres i FOR-NEXT løkken (linierne 80-110).

CPC 464 har mange simple og kraftfulde kommandoer - du kan gøre programmet mere virkningsfuldt bare ved at skrive:

15 BORDER 6,9

RUN dette. Kanten skifter nu farve mellem farve 6 og farve 9. Blinkhastigheden er sat af datamaten. For at få programmet til at køre konstant, stopper du programmmet med |**ESC**| |**ESC**|, og tilføjer:

120 GOTO 50

Læg mærke til at den blinkende kant ikke stoppede, da programmet stoppede. Dette skyldes at kanten styres uafhængigt af resten af programmet. For at få kanten til at stå stille, skal du trykke |**ESC**| to gange, og derefter ændre linie 15 til:

15 BORDER 2

Kør programmet, så stopper blinkeriet.

For at ændre farven på kurven og baggrunden, skal du ændre farven på INK i linie 30 og 40. Når du LISTer programmet skal det se således ud:

```
10 REM TEGN SINUSBOELGE
15 BORDER 2
20 MODE 2
30 INK 1,2
40 INK 0,20
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR N=0 TO 720
90 Y=SIN(N)
100 PLOT N*640/720,198*Y,1
110 NEXT
120 GOTO 50
```

Tallet 1 i slutningen af PLOT sætningen i linie 100 fortæller datamaten at den skal tegne kurven i farve angivet som INK nummer 1 i linie 30. Læs definitionen på PLOT sætningen i oversigten over nøgleord i kapitel 8. Der vil du se præcis hvordan denne sætning virker.

Hvis du ser nærmere på kurven på skærmen, så vil du se, at det ikke er en kontinuerlig linie, men at den er knækken over i mange fine dele. Den mindste del af dette er sådan et "punkt", som vi har beskrevet tidligere.

5.2.5 Grafikmarkøren og linietegning

Du har nu prøvet flere af de måder du kan oversætte et program til skærmbilleder. Flere af kommandoerne har fået en chance for at vise hvad de duer til. Når du skal tegne linier på skærmen, er der flere vigtige ting at tage hensyn til, for at undgå forvirring.

Det første du skal holde øje med, er indstillingen af program hukommelsen. Datamaten husker farveindstillingerne, selv efter en NEW kommando. Hvis du vil slette alt, skal du trykke |**CTRL**| |**SHIFT**| |**ESC**|. Husk at SAVE, hvis der er noget du skal bruge senere, inden du bruger denne kommando!

Du kan bevise det ovenstående bare ved følgende:

NEW:CLS

Efter at du har brudt ud af ovenstående program. Indtast nu:

DRAW 100,100

DRAW instruktionen tegner en ret linie fra sidste placering af GRAFIK-MARKØREN til den x,y koordinat, som du giver datamaten (100,100). GRAFIK-MARKØREN er usynlig og angiver det punkt, hvor næste grafikoperation vil starte.

For at finde ud af hvor det er, skal du bruge funktionerne XPOS og YPOS. Indtast:

PRINT XPOS

Svaret er 100

(som er det samme for YPOS på dette punkt)

Læg mærke til at hvis teksten går ned til bunden af skærmen og får skærmen til at rulle op, vil grafikken følge med. Grafik markøren bliver hvor den er. Prøv det - hold pil-ned knappen nede, indtil billedet forsvinder ud foroven, og spørg da XPOS og YPOS hvor de er nu. Grafik markøren bliver i hukommelsen.

Hvis du vil bestemme farven på en linie, der skal tegnes, så skal du tilføje farvens nummer i slutningen af DRAW kommandoen (Se beskrivelsen af PLOT kommandoen efter programmet på forrige side, det virker på samme måde).

Du skal først have givet en INK-farve, og husk at du kan kun bruge det antal INKs og farver, som er tilladt i den skærm MODE du er i nu. For at se dette kan du prøve:

```
10 MODE 1
20 INK 0,10
30 ORIGIN 0,0
40 INK 1,26
50 INK 2,0
60 DRAW 320,400,1
70 DRAW 640,0,2
```

Her er et eksempel på et program, der bruger alle de emner vi har nævnt indtil nu, og prøver et par mere. Se hvordan den første linie (10) sætter farverne, sådan at programmet vil give de rigtige resultater:

```
10 INK 0,0:INK 1,26:INK 2,6:INK 3,18: BORDER 0
20 REM DETTE PROGRAM TEGNER MOENSTRE
30 MODE 1:DEG
40 PRINT "3,4 ELLER 6 SIDET MOENSTER ? ",
50 LINE INPUT P$
60 IF P$ = "3" THEN SA = 120:GOTO 100
70 IF P$ = "4" THEN SA = 135:GOTO 100
80 IF P$ = "6" THEN SA = 150:GOTO 100
90 GOTO 50
100 PRINT "JEG REGNER";
105 IF P$ = "3" THEN ORIGIN 0,-50,0,640,0,400
ELSE ORIGIN 0,0,0,640,0,400
110 DIM CX(5),CY(5),R(5),LC(5)
120 DIM NP(5)
130 DIM DIM P%(5,81),PY%(5,81)
140 ST = 1
150 CX(1) = 320:CY(1) = 200:R(1) = 80
160 FOR ST = 1 TO 4
170 R(ST + 1) = R(ST)/2
180 NEXT ST
190 FOR ST = 1 TO 5
200 LC(ST) = 0:NP(ST) = 0
210 NP(ST) = NP(ST) + 1
220 PX%(ST,NP(ST)) = R(ST) * SIN(LC(ST))
230 PY%(ST,NP(ST)) = R(ST) * COS(LC(ST))
240 LC(ST) = LC(ST) + 360/R(ST)
250 IF LC(ST) « 360 THEN 210
252 PX%(ST,NP(ST) + 1) = PX%(ST,1)
254 PY%(ST,NP(ST) + 1) = PY%(ST,1)
260 NEXT ST
265 CLS:INK 1,2
270 ST = 1
280 GOSUB 340
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 GOTO 330
340 REM tegn cirkel plus 3,4 eller 6 andre
rundt om den
350 CX% = CX(ST):CY% = CY(ST):LC(ST) = 0
360 FOR X% = 1 TO NP(ST)
370 MOVE CX%,CY%
```

```
380 DRAW CX% + PX%(ST,X%),CY% + PY%(ST,X%),
1 + (ST MOD 3)
390 DRAW CX% + PX%(ST,X% + 1),CY% + PY%(ST,-
X% + 1),1 + (ST MOD 3)
400 NEXT X%
410 IF ST = 5 THEN RETURN
420 LC(ST) = 0
430 CX(ST + 1) = CX(ST) + 1.5 * R(ST) * SIN(SA + LC(ST))
440 CY%(ST + 1) = CY(ST) + 1.5 * R(ST) * COS(SA + LC(ST))
450 ST = ST + 1
460 GOSUB 340
470 ST = ST-1
480 LC(ST) = LC(ST) + 2 * SA
490 IF (LF(ST) MOD 360)<>0 THEN 430
500 RETURN
510 IK(1) = 1 + RND * 25
520 IF IK(1) = IK(2) OR IK(1) = IK(3) THEN 510
530 INK 1,IK(1)
540 RETURN
550 IK(2) = 1 + RND * 25
560 IF IK(2) = IK(1) OR IK(2) = IK(3) THEN 550
570 INK 2,IK(2)
580 RETURN
590 IK(3) = 1 + RND * 25
600 IF IK(3) = IK(1) OR IK(3) = IK(2) THEN 590
610 INK 3,IK(3)
620 RETURN
```

Når du kører dette program, vil det stille dig et spørgsmål (linie 40)- svar 3 for de hurtigste resultater. Programmet vil da skrive ''JEG REGNER'', og tegne en prik for hver fem sekunder (linie 245) for at vise at det stadig er igang og tænker.

Subrutinerne, der kaldes i linie 300-320 får de forskellige farver til at skifte med den fart, som bestemmes af EVERY kommandoen. Hvis du vil sætte hastigheden ned, så lav linierne 300-320 om til:

```
300 EVERY 250,1 gosub 510
310 EVERY 150,2 GOSUB 550
320 EVERY 50,3 GOSUB 590
```

For at se hvad du har gjort, skal du kigge under ''EVERY'' i kapitel 8. Det er en af de smarteste ting i AMSTRAD BASIC. En interessant mulighed i EVERY kommandoen er måden den stabler kald op til at gøre noget, hvis programmet afbrydes med ET tryk på |**ESC**|.

Tryk |**ESC**| en gang, og start da ved at trykke på en tilfældig tast. Skærmen vil blinke hysterisk, da alle farverne som har stået i kø, ræser gennem datamaten for at indhente det tabte. Der er selvfølgelig kun begrænset plads i køen, så efter et stykke tid vil hver ny EVERY kommando blive droppet indtil køen bliver tømt på en eller anden måde.

5.3 Vinduer

Brugeren kan vælge op til 8 tekst vinduer, i hvilke karakterer kan skrives, og et grafik vindue, hvori der kan plottes. Vinduerne sættes tilbage til opstarttilstanden, hvis du ændrer MODE på skærmen. Se beskrivelsen af nøgleord i kapitel 8.

NB: Hvis tekst vinduet er hele skærmen (standard), så kan skærmen rulles meget hurtigt, da det styres af hardware. Hvis tekstvinduet er mindre end hele skærmen, så udføres rulningen af software, hvilket er noget langsommere.

Kommandoen WINDOW giver venstre/højre/top/bund karaktererne i den specificerede skærmstrøm. Vinduerne kan overlappe hinanden, og kan derved give en hurtig måde at tegne fyldte kasser. Før du starter med at udforske dette, så prøv:

```
KEY 139, "mode 2:paper 0:ink 1,0:ink 0,9:
list" + chr$(13)
```

Dette sætter den lille [**ENTER**] tast til at slette tekstvinduerne og sætte farverne på PEN og PAPER til en synlig kombination, hvis du skulle komme ud for en uheldig kombination af farver. Følgende program tegner en serie vinduer på skærmen, og illustrerer to hovedpointer:

```
5 MODE 0
10 FOR N=0 TO 7
20 WINDOW #N,N+1,N+6,N+1,N+6
30 PAPER # N,N+4
40 CLS # N
50 FOR C=1 TO 200:NEXT
60 NEXT
```

Den første pointe er at en ny skærm skrives oveni den fra før, og den anden pointe er at understrege, at beskederne fra datamaten kommer i strøm # til enhver tid (medmindre de har fået ordre til andet). Skriv nu:

```
LIST
```

Programmet bliver klemt gennem strøm 0. Prøv:

```
LIST # 5
```

og derefter:

```
CLS #6
```

Dette viser at den senest adresserede strøm vil blive skrevet oveni alle de andre, og at systembeskeden Ready kommer i strøm 0, selvom listningen blev sendt til strøm 5.

Prøv kommandoen WINDOW SWAP ved at indtaste dette i linie 55:

```
55 IF N=3 THEN WINDOW SWAP 7,0
```

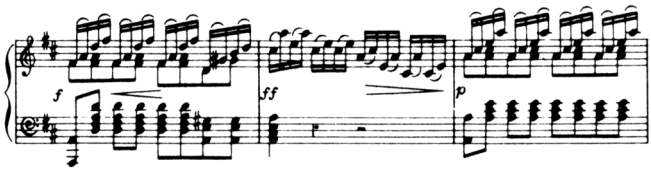
Du kan forestille dig at dette vil sende beskeden Ready til strøm 7. Kør det og se. Ved at udvikle dette simple program, vil du få en forståelse for hvordan vinduer virker og samarbejder.

6 LYDKURSUS

Lydeffekter dannes med en højtaler, som er indbygget i selve datamaten. Hvis du bruger MP1 strømforsyningen og et almindeligt TV, skal du skrue TVets volumen helt ned.

Lydstyrken kan indstilles med volumenknappen i højre ende af datamaten. Lyden kan også føres til AUX-indgangen på dit stereoanlæg, v.h.a. (I/O) soklen i venstre ende af datamatens bagside. Dette vil gøre det muligt for dig, at lytte til datamatens lyd i stereo, gennem dine HI-FI højtalere eller hovedtelefoner.

CPC 464 lyder lige så godt, som den ser ud. For at udnytte lydsoftwaren bedst muligt, behøver du at forstå tankegangen bag tidsstrukturen.



I dette afsnit behandles:
- ★ *Toneperiode*
- ★ *Sound kommandoen*
- ★ *Envelopes*
- ★ *Kødannelser og synkronisering.*

Hvis alt hvad du ønsker at gøre, er at få datamaten til at "dytte", så indtast:

PRINT CHR$(7)

Og lad det blive ved det.... men du vil mangle nogle af de mest spændende og udbytterige muligheder i CPC464. Dette er en indledning der rækker fra et overblik, og til en detaljeret gennemgang af nøgleordene og hvordan man bruger dem. Det vil give dig forklaring på hvordan man opbygger en skala af toner, opfinder forskellige musikinstrumenter, og opbygger melodier ved hjælp af dette.

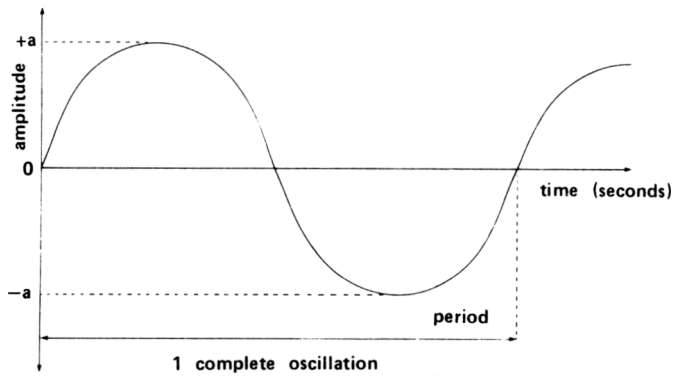
6.1 Grundlæggende lydteori

Når du hører lyden af en standardnode fra en melodi, er der flere egenskaber at lægge mærke til:

1) Frekvens, og variationer af frekvens i løbet af den tid tonen varer.
2) Styrke, og variationer i styrke i løbet af den tid tonen varer.
3) Tonen samlede varighed.

6.2 Frekvens.

I en tone er frekvensen den vigtigste egenskab. En tone kan beskrives som en jævn svingning, alle svingninger har frekvenser, periode og amplitude (størrelse). Se fig. 1.



Figur 1. Egenskaber ved en tone

Frekvensen er antallet af svingninger pr sekund, og perioden er det tidsrum en svingning varer. Amplitude er et udtryk for volumen, og har ikke noget med tonens frekvens at gøre.

Frekvens og periode hænger sammen ved følgende simple formel:

Frekvens (udtryk i Hertz) = 1/perioden (udtryk i sekunder)

Sammenhængen mellem frekvensen og toneperioden i SOUND kommandoen er:

Frekvens (udtrykt i Hertz) = 125000/(toneperiode)

Dermed svarer en toneperiode på 1000 til en frekvens på 125 Hz, og en toneperiode på 125 svarer til en frekvens på 1000 Hz.

Begge disse kunne bruges til at definere en tone, men Amstrad BASIC har valgt at bruge toneperioden. Lad dig ikke snyde af, at når toneperiodens talværdi stiger, går tonehøjden ned. Tonehøjden ses i beskrivelsen af SOUND, som toneperiode. Toneperioden er et tal mellem 0 og 4095, og skal udtrykkes som et heltal. Dette kan føre til små fejl, når man konstruerer en skala, men dette vil kun genere de allermest kritiske ører. Se appendiks VII.

Når en tone spilles på et rigtigt musikinstrument, kan tonehøjden variere, sommetider med vilje, som vibrato. hvis man bruger ENT kommandoen (ENT er forkortet af ENvelope Tone), kan vi designe et specifikt "udseende" for ændringen af tonehøjden, i den tid tonen varer. Dette kan indbygges i SOUND kommandoen. Denne mulighed kaldes for tonens envelope.

6.3 Volumen

Volumen er simpelthen et mål for hvor høj tonens styrke er. Til at sætte tonens begyndelsesstyrke på CPC 464, findes der en ligefrem skala:

forøget værdi = forøget styrke, angivet ved et heltal mellem 0 og 1 5.

Denne værdi kan også sættes i SOUND kommandoen, og kaldes for volumen.

Hvis du allerede har prøvet nogle simple toner, vil du have lagt mærke til den relativt kedelige natur, som en tone uden envelope har. Dette skyldes at rigtige musikinstrumenter har en stigning i styrke, når tonen starter, som kaldes attack, og et fald i volumen til sidst, som kaldes decay.

De forskellige former for attack og decay, som forskellige instrumenter har, kan simuleres på datamaten, ved at justere volumenforløbet, som kaldes for styrkeenvelopen, og som opbygges ved hjælp af ordren ENV.

6.4 Tonens længde

Tonens længde er en grundlæggende egenskab ved enhver musikalsk konstruktion, enheden for musisk tonelængde er fjerdedelsnoden, og der er halvnoder, helnoder etc. som er simple brøkdele eller multipla af den grundlæggende enhed. Men melodier kan spilles med forskellig tempo, og den grundlæggende varighed for fjerdedelsnoden må fastlægges.

For at gøre dette, findes der et udtryk i SOUND kommandoen som kaldes (varighed), dette er et heltal, hvor en enhed er en hundrededel sekund. (D.v.s. at 100 = 1 sek.). Bemærk at dette også kan styres af styrkeenvelopen, så du undgår misforståelser, når du også bruger en styrkeenvelope.

Stævnemøde med lydkanaler.

6.5 Andre lyde

Tilfældig støj (hvid støj) er en lyd, som kan dannes i datamaten, men det er ikke en musik node. Det kan føjes til baggrunden for en melodi, for at lave spændende variationer, eller det kan bruges for sig selv, for at lave specielle effekter. Fx er en eksplosion grundlæggende set hvid støj med en styrkeenvelope til at styre volumen. En vigtig egenskab er, at selvom der er tre separate kanaler til rådighed, med mulighed for at sætte tre toneperioder, er der kun mulighed for at lave hvid støj på en kombination af kanaler ad gangen.

6.6 Flere lyde og kanaler

De fleste stykker musik er skrevet i mindst to nøgler, bas og diskant. For at muliggøre denne fremgangsmåde på en CPC 464 er der indbygget tre lydkanaler. Disse kan alle bruges uafhængigt, eller de kan styres til overlap, hvis det er nødvendigt. Kanalvalg vælges i SOUND kommandoen med parametren (kanal status).

6.7 Kanal køer

Hver lydkanal har en kø af lyde, som den skal spille. I hver kø er der plads til fem lyde, en der spilles og fire der venter. Operativsystemet i CPC464 kan tage sig af andre ting mens lydene spilles.

6.8 Kanal status

Kommandoen SQ bruges til at læse tilstanden for den kanal du vil bruge. SQ returnerer information om ledige pladser i køen, og om "møde" og "hold". Kommandoen ON SQ GOSUB er en interupt kommando, som bruges til at bringe datamatens opmærksomhed tilbage til kommandoen SOUND.

6.9 Møde og hold

For at tvinge kanalerne til at følges ad, er der en "møde" facilitet. Dette er en markør sat ved to eller flere kanaler, der tvinger dem til at arbejde samtidig. Dette er meget nyttigt til at få to stemmer til at følges ad, hvor den ene stemme har varet lidt længere end den anden.

HOLD (se SOUND (kanal status)) er nyttig, når alle kanaler skal startes samtidig (fx ved starten på en melodi), måske efter en forsinkelse. Det er nødvendigt at have fyldt køerne allerede. I dette tilfælde bruges rækkefølgen HOLD og RELEASE.

6.10 Kommandorækkefølge ved lyddannelse

Kommandoform: SOUND G,H,I,J,K,L,M

Hvor ...
G: Kanal status
H: Toneperiode
I: Varighed
J: Volumen
K: Styrke envelope
L: Tone envelope
M: Støj periode

SOUND er en kommando, hvor alle parametrene G til M er heltal, og kun de første to er nødvendige. De øvrige er valgfrie, men har "afbudsværdier", som vil blive behandlet under beskrivelsen af de enkelte parametre.

6.10.1 Parameter beskrivelse:

G: Kanalstatus

Værdiområde 0-255

På CPC 464 er det muligt at spille tre forskellige lyde på en gang. Dette opnås ved at have tre forskellige lyd køer, som kaldes A,B og C. Heltallet, som indsættes af brugeren, gives i titalsform, men når det omsættes til 8-bit binær kode, viser bitene følgende information:

Decimal	Bit	Kommando
1	0 MB	Send lyd til kanal A
2	1	Send lyd til kanal B
4	2	Send lyd til kanal C
8	3	Mød kanal A
16	4	Mød kanal B
32	5	Mød kanal C
64	6	Hold
128	7 SB	"Skyl ud"

MB betyder Mindste Bit, SB betyder Største Bit i tabellen ovenfor.

Eksempler på decimaltal:

```
 2 = send følgende lyd til kanal B
 5 = 1+4
   = send følgende lyd til kanal A og C
98 = 64 + 32 + 2
   = send følgende lyd til kanal B, mød kanal C, og vent.
```

Det er vigtig at huske at hvis et møde skal arrangeres mellem to eller flere kanaler, så er det nødvendigt at fortælle hver kanal dette.

HOLD, på enhver kombination af kanaler, stopper behandlingen af kommandoen, og fastfryser køen indtil denne frigøres af en RELEASE kommando (eller køen "skylles ud" med en senere SOUND kommando).

H: Toneperiode

Værdiområde 0-4095

Giver perioden, der bestemmer frekvensen (tonehøjden) af den lyd, som skal spilles. Frekvensen kan findes af formlen:

frekvens = 125000/periode

Ved at bruge 0, sættes ingen frekvens. Dette bruges mest ved dannelse af støj.

I: Varighed

Værdiområde -32768 til + 32767
Afbudsværdi, hvis tallet udelades: 20

For tal større end 0 giver tallet lydens varighed i 1/100 sekunder. Hvis tallet er 0, bestemmes lydens længde af den specificerede styrke envelope.

Ved en varighed under 0, giver den numeriske værdi af tallet det antal gange, som styrke envelopen skal gentages.

J:Styrke

Værdiområde 0..15 (0..7 hvis der ikke bruges styrkeenvelope) Afbudsværdi, hvis tallet udelades:12 (4 hvis der ikke bruges styrkeenvelope).

Dette er startstyrken. Styrken kan ændres af en styrkeenvelope hvis en sådan bruges. Nul er ingen styrke.

K: Styrke envelope

Værdiområde 0..15 Afbudsværdi:0

Dette tal er nummeret på en foruddefineret styrkeenvelope. For at definere en envelope, skal man bruge kommandoen ENV. En permanent definition er envelope nummer 0. Denne kan ikke ændres med ENV kommandoen, og er sat til 2 sek. konstant lyd med styrken J.

L: Tone envelope

Værdiområde 0..15 Afbudsværdi:0

Værdien angiver en forudefineret toneenvelope. For at definere en envelope, skal du bruge kommandoen ENT. En permanent definition er toneenvelope 0, som ikke kan ændres af ENT kommandoen. Denne er sat til konstant toneperiode.

M: Støj periode

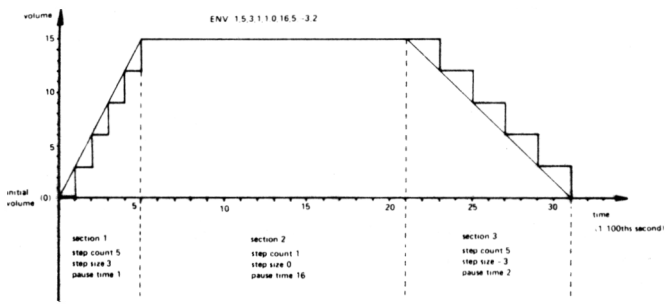
Værdiområde 0..15

Afbudsværdi:0

Dette angiver den støj, der skal lægges til lyden. Hvis afbud eller 0 bruges, kommer der ingen støj. Husk at kun en støj periode kan angives ad gangen. Dette betyder, at alle kanaler, der gør brug af støj, vil modtage samme støj.

6.11 ENV kommandoen og volumen envelopes

Anslag og afdæmpning er nødvendigt for at give en tone liv. Denne kommando (ENvelope Volume) giver mulighed for at styre en tones styrke. Det er en god ide at tegne styrkeforløbet for ens tone op på papir, før man forsøger at omsætte dette til tal. Se eksemplet herunder:



Figur 3: Styrke envelope eksempel

Formen skal optegnes med tal på, så den kan indtastes i en kommando. For at gøre dette, skal man dele faconen ind i sektioner, som du må have op til 5 af. Men i hver sektion skal faconen være en ret linie. Nu skal du dele hver sektion ind i trin, idet du vælger antallet af trin som (trintal). Hvert af disse trin vil tage en bestemt tid, som kaldes pausetiden. Pausetid $f \times 1/100$ sek.

Disse trin kan også have en stigning, eller et fald i styrke. Dette kaldes trinstørrelse. Hvis du ikke bruger nogen trin i en sektion, bliver styrken fra forrige sektion bare genbrugt.

N.B. Når du prøver på at efterligne et musikinstrument, er det ofte tilfældet at begyndelses og slutstyrken er nul. Derfor skal styrken i SOUND kommandoen være nul, og den nødvendige styrke skal komme alene fra styrkeenvelopen.

Kommandoens form:

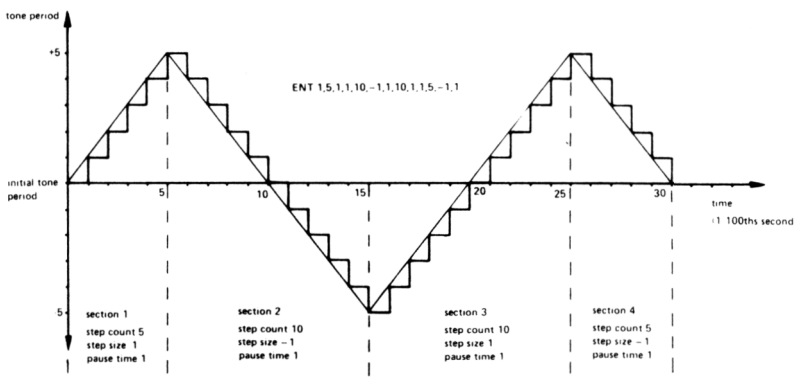
ENV N,P1,Q1,R1,P2,Q2,R2,P3,Q3,R3,P4,Q4,R4,P5,Q5,R5

N:Envelopenummer	Værdiområde fra 0..15
P1..5:Trin antal for sektion 1..5	Værdiområde fra 0..127
Q1..5: Trinstørrelse for sektion 1..5	Værdiområde fra -128.. + 127
R1..5: Pausetid for sektion 1..5	Værdiområde fra 0..255

Envelopenummeret er et navn. En komplet sektion er påbudt, men antallet af sektioner er frit op til 5. Fx sker der ikke noget ved at udelade sektion 4 og 5. Der antages så kun at være tre sektioner.

6.12 ENT kommandoen og tone envelopes

Disse har samme opbygning som styrkeenvelopes, forskellen er at en tone envelope giver små variationer i tonens frekvens. Dette virker som en slags vibrato. Når man har besluttet sig til udseendet af en tones frekvens, skal man tegne den som før, og dele den ind i sektioner og trin. Se eksemplet herunder:



Figur 4: En tone envelopes udseende.

Den største forskel på en styrkeenvelope og en tone envelope er at en tone envelope ikke har nogen indflydelse på tonens varighed. Hvis tone envelopen løber ud før tonen, vil tonen bare fortsætte med konstant frekvens, til den er færdig. Hvis den derimod er for lang, vil den overskydende del bare ikke blive brugt.

For at gentage en tone envelope, mens en tone klinger, skal man bruge et negativt envelopenummer. (Man skal ikke kalde den op med et negativt tal i SOUND kommandoen).

Kommandoens form:

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

S: Envelopenummeret	Værdiområde fra 1..15 (minus foran, hvis der gentages)
T1..5: Trinantal for sektion 1..5	Værdiområde fra 0..239
V1..5: Trin størrelse for sektion 1..5	Værdiområde fra -128.. + 127
W1..5: Pausetiden for sektion 1..5	Værdiområde fra 0..255 (i 1/100 sekund)

S er obligatorisk, komplet definition på hvert afsnit er påbudt.

Når et envelopenummer defineres, bliver alle tidligere indstillinger slettet. Hvis man definerer en envelope af en af typerne, uden nogen sektioner, vil man slette indstillingerne for denne envelope.

6.13 Andre forbundne funktioner og kommandoer

SQ(x)

x er kanalnummeret, som skal skrives som 1,2 eller 4.

Disse værdier repræsenterer kanalerne A,B eller C respektivt. Dette er en forespørgsel til tilstanden for den pågældende kanal.
Som vist i tabellen nedenfor kan man trække en del information ud om en kanals tilstand:

Bit 0..2	Antal frie pladser i den undersøgte kø.
Bit 3	Møde med kanal A på vej.
Bit 4	Møde med kanal B på vej.
Bit 5	Møde med kanal C på vej.
Bit 6	Hold på vej i køen.
Bit 7	Kanalen spiller nu.

Bit 3 til 6 (møde og hold) blev beskrevet i afsnittet om kanalstatus. Den eneste anden facilitet i denne test, er at desarmere ON SQ GOSUB kommandoen, som beskrives nu.

ON SQ GOSUB

ON SQ(y) GOSUB linienummer

y er kanalnummeret, 1,2 eller 4

Dette er en interrupt kommando, som afsikres, og da føler om der er plads i den pågældende kanals lydkø. Denne hopper ind i en subrutine, som skal definere lyden med SOUND. Kommandoerne SQ og SOUND desarmerer begge interruptet. Subrutinen skal slutte med RETURN, som sædvanligt. Alle kanaler har samme prioritet, og når der er plads i køen, vil subrutinen køre, og derved desarmere sig selv samtidig. Derfor skal subrutinen armere interruptet, hvis det skal bruges igen.

RELEASE

RELEASE z

z angiver kanalnummre(t), og er et heltal fra 1..7 (summen af de brugte kanalers numre)

Som beskrevet under SOUND kommandoens parameter (kanal status), er det muligt at angive at køen skal holde. RELEASE bruges simpelthen for at frigive lydkøen igen. RELEASE på en kanal som ikke er igang ændrer intet.

Bit 0: Kanal A
Bit 1: Kanal B
Bit 2: Kanal C

7 Printere og styrepinde

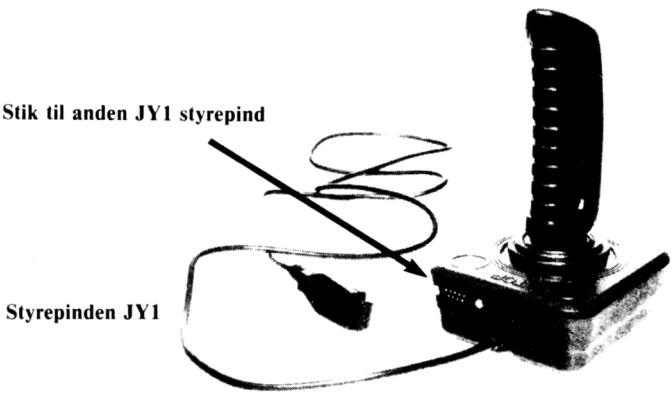
CPC464 behøver ikke ekstra forsatser (interfaces) for at arbejde med enten en eller to styrepinde og en Centronics kompatibel printer.

I dette kapitel behandles:

★ Styrepinde
★ Parallel printere
★ Tilslutning

AMSOFT styrepinden model JY1 er et stykke ekstraudstyr, som du måske vil købe, for at bruge CPC464 datamaten til spil. JY1 kan forbindes direkte til bagsiden af datamaten, v.h.a. det 9-benede stik, som er mærket USER PORTS (I/O). Amstrad CPC464 datamaten kan bruges med to styrepinde. Nummer to styrepind forbindes til siden af første styrepind.

Der er ikke nogen specielle problemer ved at forbinde en styrepind til CPC464. Forbindelserne kan ses i appendiks V i slutningen af denne bog, sammen med alle andre I/O funktioner og stik.



7.1 Styrepinde

Det indbyggede programmel i AMSTRAD CPC464 datamaten kan bruge op til 2 styrepinde. Disse opfattes som en del af tastaturet, og kan derfor aflæses med funktionerne INKEY$ og INKEY. Bemærk at hvis der kun er en affyringsknap på din styrepind, så er det sikkert knappen "fyr 2" i ARMSTRAD CPC 464-sprog.

Der findes en BASIC kommando til at læse en styrepinds tilstand. Dette er JOY(0) hhv. JOY(1) for de to styrepinde. Funktionen returnerer et tal, som skal læses binært. Tilstanden aflæses en gang hver 1/50 sek.

Fra styrepinden fås værdier som følger, hvor TAST er værdien, som skal bruges i en INKEY sætning, og SPEJL er den tilsvarende tast på tastaturet.

Styrepind 1	**JOY(0)**	**TAST**	**Styrepind 2**	**JOY(1)**	**TAST**	**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
Fyr 2	Bit 4	76	Fyr 2	Bit 4	52	G
Fyr 1	Bit 5	77	Fyr 1	Bit 5	53	F

Bemærk, at når styrepind 2 bliver aflæst, kan CPC464 ikke se forskel på denne og tasterne på tastaturet. I praksis er det usandsynligt at man vil bruge tastaturet og styrepinden samtidig, i det samme program. Derfor vil dette ikke være noget problem. Tastaturet kan bruges i stedet for styrepind 2.

Når du bruger AMSOFT styrepinden JY1, er styrepind 2 magen til den første. Tilslutningen foregår til siden af styrepind 1. Der skal ikke trækkes ekstra ledninger for at bruge to styrepinde samtidig.

Det 9-benede stik mærket **USER PORTS (I/O)** kan bruges til standard styrepinde, som også kan bruges til andre datamater. Disse vil ikke kunne bruges sammen med en styrepind mere, medmindre man bruger et specielt mellemled.

Programmører kan overveje at bygge en mulighed for at vælge mellem at bruge styrepind eller markørtasterne på tastaturet i starten af programmerne. Tasten |**COPY**| kunne så fx bruges til affyringsknap.

7.2 Printer forbindelser

AMSTRAD CPC464 tillader forbindelse af en standard printer af CENTRONICS type.

Printerkablet er blot en ledning mellem **PRINTER** porten og parallelprinterens stik. Læg mærke til at der er to ben mindre på printet, end på printerens stik. Dette giver mulighed for at bruge et standardstik til forbindelsen til printet.

De tekniske detaljer findes i Appendiks V.

Kablet skal opbygges således at ben 1 på datamaten forbindes til ben 1 på printeren, ben 19 på datamaten forbindes til ben 19 på printeren o.s.v. Ben 18 og 36 på printeren skal ikke forbindes til datamaten.

Læg mærke til at de nederste ben på datamaten er nummereret fra 19 og opad (i stedet for 18 og opad, som man skulle forvente, da der er 17 ben i den øverste række). Dette er gjort således for at de ledninger, der skal bruges har samme nummer på datamaten og printeren.

Datamaten bruger signalet *BUSY* (ben 11) til at synkronisere sig med printeren. Datamaten vil vente, hvis printeren ikke er klar - *OFF LINE*.

Brugeren skal ikke bruge nogen ekstra kommandoer for at sætte printeren op. Det, der skal skrives på printeren sendes blot til strøm nr. 8:

LIST #8

Dette vil liste BASIC programmet i hukommelsen, så længe programmet ikke er beskyttet.

Inden i et program kan der skrives til printeren på denne måde:

PRINT # 8,"Dette skrives på printeren"

Mange printere vil automatisk fortsætte på næste linie, hvis den linie, som man sender til den, er for lang. Se dette i printerens håndbog. AMSTRAD BASIC kan gøre det samme, dette styrer man med ordren WIDTH. Afbudsværdien er 132. En ny værdi kan indsættes med fx WIDTH 80.

Hvis du sætter værdien til 255, vil AMSTRAD BASIC ikke sende udskrifterne ud linievis, men vil helt overlade det til printeren at checke linielængden. BASICen har indbygget en tæller, som fortæller hvor langt man har skrevet på den sidste nye linie. Denne kan aflæses med funktionen POS.

IF POS(# 8) > 50 THEN GOTO 100

CPC 464 udsender karaktererne *nylinie* - CHR$(10) - *vognretur* - CHR$(13) - ved hvert linieslut. Printeren vil normalt indeholde en omskifter til at skifte mellem de forskellige former for inddata. Og det vil klart fremgå, hvilken standard din printer er indstillet til, i samme øjeblik som du sætter den til at skrive ud.

7.3 Udskrivning af grafik

Den håndbog, som du har fået sammen med din printer, vil fortælle dig hvilke kontrolkoder den har indbygget. Almindeligvis har de en form så man kan bruge:

PRINT CHR$(n)

Nogle printere har karakterer som ligner dem, AMSTRADen har indbygget. Det er usandsynligt at de vil have samme karakterkoder som AMSTRADen, så du bliver nød til at lave din egen konverteringstabel, for at tilpasse dette til din printer.

Selv om printerstikket er lavet til at arbejde sammen med billige matrixprintere, kan det selvfølgelig også bruges til typehjulsprintere, grafiske plottere, og flerfarvede ink-jet printere. Nøglen til dette er standard parallelstikket.

8 Kort redegørelse for AMSTRAD BASIC

En oversigt over BASIC kommandoerne, illustreret med eksempler på præcis brug af disse og de tilhørende nøgleord.

I dette kapitel behandles følgende emner:

★ Notation
★ Specielle karakterer og deres nødvendighed
★ Alle AMSTRAD BASIC kommandoer i alfabetisk orden

Dette kapitel behandler summarisk den BASIC dialekt, som leveres med AMSTRAD CPC464 i ROM. De fleste ordrer tilhører en standarddialekt af BASIC, med visse udvidelser for at kunne svare til CPC464s hardware.

8.1 Notation

Specialkarakterer

& eller &H	Står foran hexadecimale konstanter
&X	Står foran binære konstanter
:	Adskiller sætninger på samme linie
#	Står foran strømnumre

Datatyper

Strenge kan være fra 0 til 255 karakterer lange, og kaldes her «strengudtryk». Strenge kan sammensættes med + tegnet, hvis resultatet ikke overstiger 255 karakterer.

Numeriske data kan være heltal eller reelle tal. Heltal skal ligge i området fra -32768 og op til 32767. Reelle tal gemmes med en ca. 9 cifres nøjagtighed i området (+/-)1.7E+38 og det nærmeste tal til 0 er ca. 2.9E-39.

Variabler kendetegnes af:
% = heltal, ! = reelt tal og $ = streng.

Et «numerisk udtryk» er ethvert udtryk, som resulterer i et tal: Det kan simpelthen være tal, det kan være en variabel eller det kan være et regnestykke indeholdende variabler og tal. Næsten alt, der ikke er et «streng udtryk».

Et «strøm udtryk» henviser til et «numerisk udtryk», som identificerer skærmen, prjnteren, kassettebåndoptageren eller hvor teksten nu skal "strømme" hen.

Fejlen "Improper argument" betyder at et «numerisk udtryk» har resulteret i et tal, som er udenfor det område, som er tilladt i den pågældende situation. Eller at kommandoens parameter er forkert i den pågældende sammenhæng, og at BASICen ikke accepterer dette som inddata.

NØGLEORD

Bemærk venligst, at AMSTRAD BASIC nøgleordene beskrives på følgende måde:

NØGLEORD
Syntaks/funktion

Eksempel

Beskrivelse

Forbundne kommandoer

Vigtigt:

Nøgleord er enten:

KOMMANDOER: Operationer, der udføres direkte.
FUNKTIONER: Operationer, der bruges som parametre i et udtryk.

Parenteser

() skrives, når disse er påkrævede af en kommando eller funktion.

[]denne type parenteser anvendes omkring valgfrie punkter.
«»bruges omkring udtryk, der forekommer i den efterfølgende beskrivelse.

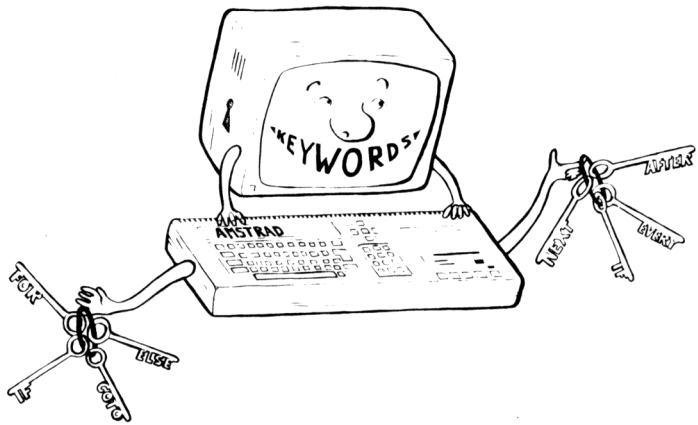
Apostroffer

Kun formen " " bruges i BASIC. Formen ' bruges for at understrege vigtige ting, og denne form kan optræde inde i strengudtryk.

Indtastning

BASIC omformer alle nøgleord indtastet med små bogstaver til store bogstaver, når et program udskrives. Eksemplerne brugt her anvender STORE bogstaver, eftersom det er sådan programmerne vil se ud, når de udskrives. Hvis du bruger små bogstaver, når du indskriver programmer, vil du lettere kunne finde indtastningsfejl, eftersom forkert stavede ord ikke omformes til STORE bogstaver.

Nøgleord afskilles fra resten af teksten i et program, med mellemrum. Dette gøres fordi AMSTRAD BASIC tillader brugeren at "begrave" nøgleord i variabelnavne: Fx er end2 og LISTCODE brugbare variabelnavne i AMSTRAD BASIC.



ABS

ABS («numerisk udtryk»)

PRINT ABS(-67.98)
67.98

FUNKTION: Returnerer den numeriske værdi af udtrykket givet i parentesen. Dette betyder at negative tal returneres som positive, og positive forbliver positive.

Forbundne nøgleord: SGN

AFTER

AFTER «heltals udtryk»|, «heltalsudtryk»| GOSUB «linienummer»

AFTER 200,2 GOSUB 320

KOMMANDO: Kalder en subrutine efter et givet stykke tid. Første heltalsudtryk angiver forsinkelsens varighed i antal 0.02 sek., og andet heltalsudtryk (i området 0...3) angiver hvilken af de fire timere, som skal bruges.

Forbundne nøgleord: EVERY, REMAIN

ASC

ASC («strengudtryk»)

PRINT ASC("X")
88

FUNKTION: Giver talværdien for den første karakter i en streng så længe ASCII karakterer bruges.

Forbundne nøgleord: CHR$

ATN

ATN («numerisk udtryk»)

PRINT ATN(1)
0.785398163

FUNKTION: Beregner arcustangens (idet numerisk udtryk antages at være et reelt tal i radianer fra 0 til pi/2) til tallet som angives.

Forbundne nøgleord: SIN,COS,TAN,DEG,RAD

AUTO

AUTO |«linienummer»||, «tillæg»|

AUTO 100,50

KOMMANDO: Giver automatisk linienumre. Linienummer angiver første linienummer, som skal laves, hvis du vil føje noget til et eksisterende program. Værdien af tillægget bliver afstanden mellem linienumrene. Begge får værdien 10, hvis de udelades.

Når en eksisterende linie er i fare for at blive overskrevet, indsætter BASIC en stjerne ★ efter linienummeret som advarsel.

BIN$

BIN$ («positivt heltalsudtryk»|, «heltalsudtryk»|)

PRINT BIN$(64,8)
01000000

FUNKTION: Fremstiller en streng af binære cifre, som repræsenterer værdien af det positive heltalsudtryk, med foranstillede nuller, så det totale antal cifre er som angivet af det andet heltalsudtryk.

Forbundne nøgleord:HEX$, STR$

BORDER

BORDER «farve»|, «farve»|

BORDER 3,2

KOMMANDO: Ændrer farven på skærmens kant. Hvis du angiver to farver, vil kanten skifte mellem disse to med den fart, som angives af kommandoen SPEED INK, hvis denne bruges. Farverne kan vælges mellem 0..26.

Forbundne nøgleord: SPEED INK

CALL

CALL «adresse udtryk»|,«liste af parametre»|

CALL &BD19

KOMMANDO: Tillader brugen af en maskinkodesubrutine fra BASIC. Bruges med forsigtighed, ikke en funktion for den uerfarne, da den kan føre til crash. Eksemplet ovenfor er ufarligt, da det kun venter på næste tilbageløb efter et halvbillede. Dette er brugbart til at få grafik til at bevæge sig jævnere, når man laver bevægelig grafik.

Forbundne nøgleord:UNT

CAT

CAT

CAT

KOMMANDO: Får datamaten til at læse en kassette, og udskrive navnene på alle filer der mødes. Dette har ingen indflydelse på det program, som er i hukommelsen, og kan derfor bruges til at checke at et program man lige har SAVEt er kommet ind på båndet. Kommandoen beder dig om at spille kassetten, og når den finder et program, skriver den:

FILENAME bloknummer flag OK

Flaget bruges til at fortælle hvilken filtype der er fundet:
$ En BASIC program fil
% En beskyttet BASIC fil
★ En ASCII tekstfil
& En binær fil

Andre karakterer kan optræde i denne kolonne, hvis filen ikke er lavet af BASIC.

Forbundne nøgleord: LOAD, RUN, SAVE

CHAIN
CHAIN MERGE

CHAIN «filnavn»|, «linienummer udtryk»|
CHAIN MERGE «filnavn»|, «linienummer udtryk»|
|,DELETE «linienummer område»|

CHAIN "TEST",350

KOMMANDO: CHAIN indlæser et program fra en kassette, idet det eksisterende program slettes. CHAIN MERGE sammenbygger det nye program med det gamle. Linienummeret angiver det sted, der skal startes i det nye program. Hvis der ikke angives et linienummer, vil BASIC vælge det laveste eksisterende.

Hvis der ikke angives noget filnavn, vil BASIC forsøge at bruge den første fil, som mødes. Hvis første karakter i filnavnet er et !, så bliver denne karakter fjernet fra filnavnet, og de sædvanlige beskeder udskrives ikke på skærmen.

CHAIN MERGE bibeholder alle variabler, mens brugerfunktioner og åbne filer afskaffes. ON ERROR GOTO slås fra, et RESTORE udføres, og DEFINT,DERREAL og DEFSTR indstillinger slettes. Alle aktive FOR WHILE og GOSUB kommandoer glemmes. Beskyttede filer kan ikke sammenbygges.

Forbundne nøgleord:LOAD, MERGE

CHR$

CHR$(«heltalsudtryk»)

PRINT CHR$(100)
d

FUNKTION: Konverterer et tal til dets ekvivalente karakter (idet AMSTRAD CPC464 karaktersættet, som beskrevet i appendiks III, bruges).

Forbundne nøgleord:ASC,LEFT$,RIGTH$,MIDT$,STR$

CINT

CINT («numerisk udtryk»)

10 n = 578.76543
20 PRINT CINT(n)
RUN
579

FUNKTION: Konverterer den givne værdi til et afrundet heltal i området -32768 til +32767.

Forbundne nøgleord:CREAL,INT,FIX,ROUND,UNT

CLEAR

CLEAR

CLEAR

KOMMANDO: Sletter alle variabler og filer.

CLG

CLG |«dækket blæk»|

KOMMANDO:Sletter grafik skærmen.

Forbundne nøgleord:CLS,ORIGIN

CLOSEIN

CLOSEIN

CLOSEIN

KOMMANDO: Lukker kassetteindlæsningsfilen. Kommandoer som fx NEW og CHAIN MERGE vil droppe åbne filer.

Forbundne nøgleord:OPENIN,CLOSEOUT

CLOSEOUT

CLOSEOUT

CLOSEOUT

KOMMANDO: Lukker kassetteudlæsnings filen.

Forbundne nøgleord:OPENOUT,CLOSEIN

CLS

CLS [#«strømudtryk»]

CLS

KOMMANDO: Sletter det givne skærmvindue til dets papirfarve.

CONT

CONT

CONT

KOMMANDO: Fortsæt programudførelsen efter ★Break★, STOP eller END, så længe programmet ikke er blevet ændret. Direkte kommandoer må bruges.

COS

COS («numerisk udtryk»)

?COS(34)
-0.848570274

og

deg:?cos(34)
0.829037573

FUNKTION: Beregner cosinus til et givet tal. Funktionen regner i radianer, medmindre denne er instrueret om at regne i grader, med kommandoen DEG. Læg mærke til brugen af ? som forkortelse af PRINT, og brugen af små bogstaver i nøgleordene, i det ovenstående eksempel.

Forbundne nøgleord:SIN,TAN,ATN,DEG,RAD

CREAL

CREAL(«numerisk udtryk»)

```
5 DEFINT n
10 n = 75.765
20 d = n/34.6
30 PRINT d
40 PRINT CREAL(n)
50 PRINT n/55.4
RUN
2.19653179
76
1.37184116
Ready
```

FUNKTION: Omformer et tal til et reelt tal (i modsætning til heltal).

Forbundne nøgleord:CINT

DATA

DATA «liste af «konstanter»

```
10 REM Brugernavneliste
20 DIM Bruger$(5)
30 DIM Brugernavn$(5)
40 FOR n = 1 to 5
50 READ Bruger$(n)
60 READ Brugernavn$(n)
65 PRINT Bruger$(n);'' ''Brugernavn$(n)
70 DATA Bob,Smith,Dicky,Jones,
Malcolm,Green,Alan,Brown,Ivor,Curry
90 NEXT
```

Kommando: Erklærer konstant data til brug i et program. En af de mest brugte muligheder i BASIC, som klumper konstante data sammen i DATAsætninger, så de kan bruges som nødvendigt. DATAene skal svare til den variabel de skal indlæses i. En DATAsætning kan stå hvorsomhelst i et program.

Forbundne nøgleord:READ,RESTORE

DEF FN

DEF·FN «navn»|(«formelle parametre»)| = «generelt udtryk»

```
10 DEF FNrente(kapital) = 1.14★kapital
20 INPUT "Hvad er kapitalen?";kapital
30 PRINT "Tilgodehavenet plus rente efter et år er",FNrente(kapital)
```

KOMMANDO: BASIC tillader programmet at definere sine egne simple funktioner. DEF FuNktion delen af denne mekanisme definerer funktionen, og laver en programspecifik funktion, som virker ligesom fx den i BASICen indbyggede COS funktion.

Denne kan kaldes igennem hele programmet. Varibeltyperne skal være rigtige og DEF FuNktions kommandoen skal stå udenfor programmets hovedløkke.

DEFINT
DEFSTR
DEFREAL

DEFtype «bogstavområde»

```
DEFINT I-N
DEFSTR A,W-Z
DEFREAL
```

KOMMANDO: Definerer variabeltyper, hvor typen er heltal, reelt tal eller streng. Variablen bliver defineret efter første bogstav i dens navn. Dette kan være et stort eller et lille bogstav.

Forbundne nøgleord:LOAD,RUN,CHAIN,NEW,CLEAR

DEG

DEG

DEG

KOMMANDO:Sætter datamaten til at regne med grader. Hvis dette ikke bruges, vil SIN,COS og tilsvarende funktioner antage brugen af radianer. Kommandoen sætter datamaten til grader, indtil dette ændres af CLEAR eller RAD kommandoerne, eller et nyt program loades.

Forbundne nøgleord:RAD

DELETE

DELETE «linienummerområde»

DELETE 100-200

KOMMANDO: Kommandoen sletter en del af det eksisterende program, som defineret i linienummerområdet. Dette kan ikke gøres godt igen, hvis man gør det forkert, så brug den med forsigtighed, og se efter, om der er fejlskrivninger, inden du trykker |**ENTER**|.

Forbundne nøgleord:NEW

DI

DI

```
10 CLS
20 TAG
30 EVERY 10 GOSUB 100
40 X1 = RND★320:X2 = RND★320
50 Y = 200 + RND★200
60 FOR X = 320-X1 TO 320 + X2 STEP 2
70 DI:PLOT 320,0,1:MOVE X-2,
Y:PRINT " ";:MOVE X,Y:PRINT "#",:EI
80 NEXT
90 GOTO 40
100 MOVE 320,0
110 DRAW X + 8,Y-16,0
120 RETURN
```

Kommando: Slår interrupts fra (undtagen ★ Break ★) indtil de bliver slået til igen med EI, eller af RETURN efter en interrupt GOSUB rutine.

Bruges når programmet skal arbejde uden interrupts - fx når to rutiner i et program konkurrerer om ressourcerne. I eksemplet ovenfor konkurrer hovedprogrammet og interrupt subrutinen om brugen af grafik billedet.

Forbundne nøgleord:EI

DIM

DIM «liste af: «indicerede variabler»

```
10 CLS:PRINT "Indtast 5 navne.....":PRINT
20 DIM B$(5)
30 FOR N=1 TO 5
40 PRINT "Navn";N;"tak";
50 INPUT B$(N)
60 NEXT
70 FOR N=1 to 5
80 PRINT "Hvor klogt af dig ";B$(N);
" at koebe en CPC464"
90 NEXT
```

KOMMANDO: Lav plads til en indiceret variabel og giv det højeste indeks. BASIC skal vide hvor meget plads, der skal reserveres, ellers vil det vælge 10 pladser automatisk. Når størrelsen en gang er fastsat, må størrelsen ikke ændres, så vil du få en fejludskrift.

En indiceret variabel er en, hvor det samme variabelnavn kan antage en hel stribe værdier, idet antallet er angivet i DIM sætningen. Første tal i DIM sætningen kan ses som etagerne i et parkeringshus, de efterfølgende tal er da antallet af parkeringspladser på hver etage. En dyb forståelse for indicerede variabler er nødvendig for at kunne bruge BASIC godt. Størrelsen af en indiceret variabel begrænses kun af den tilgængelige hukommelse, og programmørens evne til at huske hvad han har læst ind i variablen og hvor.

Forbundne nøgleord:ERASE

DRAW

DRAW «x-koordinat»,«y-koordinat»|,«dækket blæk»|

DRAW 200,200,13

KOMMANDO: Tegner en linie på skærmen fra de nuværende koordinater for grafik markøren, til en absolut position. Koordinaterne er de samme i alle tre skærmindstillinger. Yderligere eksempler findes i kapitel 5.

Forbundne nøgleord:DRAWR,PLOT,PLOTR,MOVE,MOVER,TEST,TESTR,XPOS, YPOS,ORIGIN

DRAWR

DRAWR «x afstand», «y afstand»|,«dækket blæk»|

DRAWR 200,200,13

KOMMANDO: Tegn en linie på skærmen fra den nuværende position for grafik markøren, til en position relativt til denne.

Forbundne nøgleord: DRAW,PLOT,PLOTR,MOVE,MOVER,TEST,TESTR,XPOS, YPOS,ORIGIN

EDIT

EDIT «linienummer»

EDIT 110

KOMMANDO: Editerer en programlinie, ved at kalde dens nummer. Se kapitel 1.4.

Forbundne nøgleord:LIST

EI

EI

EI

KOMMANDO: Muliggør interrupts, som er slået fra med DI kommandoen.

Forbundne nøgleord:DI

END

END

END

KOMMANDO: Slutningen på programmet. END kommandoen er underforstået i AMSTRAD CPC464 BASIC, når programmet passerer sidste instruktion i programmet. END lukker alle kassette filer og returnerer til direkte kommandoer. Lydkøer vil fortsætte til de er tomme.

Forbundne nøgleord:STOP

ENT

ENT «envelopenummer»|,«envelope sektioner»|

KOMMANDO: Mens en lyd laves er det muligt at ændre dens tone. En tone envelope definerer hvordan tonen skal ændres. Se kapitel 6 og appendiks VII for en mere fuldstændig beskrivelse af lyd og brugen af denne.

Envelopenummeret er et heltalsudtryk, hvis absolutte værdi skal være i området fra 1..15, som angiver hvilken af 15 envelopes, der skal ændres. Hvis tallet er negativt, betyder det at envelopen skal gentages.

Op til 5 envelope sektioner er tilladt, og hver af dem antager en af formerne:

Trinantal, trinstørrelse, pausetid
eller: Toneperiode, pausetid

Første form giver en stigning (eller et fald) relativt til den nuværende toneperiode.

Trinantallet angiver antallet af trin i sektionen, et heltals udtryk i området 0..239.

Trinstørrelsen angiver hvor meget tonen skal ændres pr. trin i sektionen, et heltalsudtryk i området -128.. + 127.

Pausetiden angiver den tid, der skal ventes mellem hvert trin, et heltalsudtryk som giver tiden i 0.01 sek. Udtrykket skal give en værdi i området 0..255, idet 0 læses som 256.

Anden form giver en indstilling af en ny toneperiode.

Toneperioden er et tal mellem 0..4095.

SOUND kommandoen sætter begyndelsestoneperioden, og kan angive at en af de 15 tone envelopes skal bruges. Hvis der ikke angives nogen envelope, eller en envelope, som endnu ikke er sat op, bruges, vil tonen være konstant igennem hele den tid den lyder.

En tone envelope har ingen indflydelse på en tones varighed. Hvis der er ubrugte trin i envelopen, efter at tonen er færdig, vil de bare blive sprunget over.

En tone envelope med et negativt nummer, vil blive gentaget forfra hvergang den er færdig, indtil lyden er færdig.

Udtrykkene i toneenvelope kommandoen bliver brugt, når kommandoen mødes, og de bliver derefter gemt til fremtidig brug. Når toneenvelopen bruges, vil selve kommandoen ikke blive kørt igen.

Hver gang en toneenvelope bliver indstillet, vil dens forrige indstillinger blive glemt. Hvis man ændrer en toneenvelope, mens den bliver brugt, vil man få ubestemmelige, men måske interessante virkninger frem.

Hvis man specificerer en envelope, uden at angive nogen sektioner, vil det medføre at alle tidligere indstillinger glemmes. Hvis man forsøger at bruge en sådan tom envelope, vil den ingen virkning have idet dens oprindelige (manglende) indstilling vil blive brugt i stedet.

Forbundne nøgleord:ENV, SOUND

ENV

ENV «envelopenummer»|,«envelope sektioner»|

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

KOMMANDO: Mens en lyd laves, er det muligt at variere dens volumen v.h.a. denne kommando. En styrkeenvelope består af:

trin antal,trinstørrelse,pausetid

Dette definerer hvordan styrken skal ændres, ved at angive et antal trin fra 0..127, trinstørrelsen fra -128.. + 127, og pausetiden i 0.01 sek. intervaller i området 1..256.

Envelopenummeret er et heltalsudtryk, som giver et tal i området 1..15, dette angiver hvilken af 15 envelopes, du vil indstille.

Op til 5 envelope sektioner kan bruges. Hver af dem har en af formerne:

trinantal,trinstørrelse,pausetid
eller: hardware envelope, envelope periode

Den første form angiver en envelope under softwarekontrol, hvor parametrene er:

trinantal, som angiver antallet af trin i en sektion. Værdiområde 0..127.

trinstørrelse: giver størrelsen med hvilken styrken skal ændres i hvert trin i sektionen. Værdiområde -128.. + 127.

Eller hvis trin antallet er nul, så bliver tallet brugt som absolut størrelse af volumen.

pausetiden, angiver tiden, der skal ventes i hvert trin. Et heltalsudtryk, som angiver tiden i 1/100 sekund, i området 0...255. 0 opfattes som 256.

Anden form angiver en envelope som skal bruges direkte af lydhardwaren, hvor:

hardware envelope er værdien som skal indsættes i enveloperegistret i lydchippen (register 15 oktal).

envelope periode er tallet, som skal indsættes i envelopeperiode registret i lydchippen (register 13,14 oktal).

Hardware envelopes har ikke nogen forbundet pausetid, så næste sektion af envelopen bliver brugt med det samme. Det er derfor anbefalelsværdigt at næste trin skal have en pause af en passende længde. Hvis der ikke er noget næste trin, vil en pause på to sekunder blive antaget.

SOUND kommandoens indstiller begyndelsesstyrken og kan benytte sig af en af de 15 styrkeenvelopes. Hvis ingen envelope, eller en envelope, som ikke er indstillet, bliver brugt, vil styrken være konstant i den tid lyden varer.

Indstiller du en trinstørrelse på nul, med et trinantal forskelligt fra nul, vil medføre at styrken bliver konstant i den pågældende sektion.

Udtrykkene i styrkeenvelopen bliver beregnet, når kommandoen mødes, og resultatet gemmes til senere brug. Hvis du bruger styrkeenvelopen, vil selve kommandoen ikke blive gentaget.

Hver gang en given styrkeenvelope bliver ændret, vil dens forrige indstillinger blive slettet. Hvis du ændrer envelopen mens lyden er igang, vil du lave ubestemmelige, men måske interessante virkninger.

Hvis du angiver en envelope med nul sektioner, vil du slette de tidligere envelopeindstillinger. Envelope begyndelsesindstillingen vil blive brugt i stedet.

Forbundne nøgleord:ENT, SOUND

EOF

EOF

```
PRINT EOF
-1
```

FUNKTION: Tester om kassetteindgangen er ved slutningen af en fil. Returnerer -1 (sand) ved slutningen, ellers 0 (falsk).

Forbundne nøgleord:OPENIN

ERASE

ERASE «liste af; «variabelnavne»

```
ERASE A,B$
```

KOMMANDO: Når en indiceret variabel ikke længere skal bruges, kan den slettes og hukommelsen bruges til andre formål.

Forbundne nøgleord:DIM

ERR
ERL

```
ERR
ERL
10 CLS
20 ON ERROR GOTO 1000
30 DATA AGNETE,EMMA,JOHANNE,HELENE,OLGA
40 READ A$
50 PRINT A$
60 GOTO 40
70 REM her starter fejlsporingen
1000 IF ERR=4 THEN PRINT ''jeg har fundet en fejl,
mangel på DATA!''
1010 IF ERL<70 AND ERL>20 THEN PRINT
''i linierne 30-60''
1020 END
```

VARIABLER: Disse variabler bruges til at håndtere fejl, således at en subrutine kan finde typen af fejl, og linien hvori fejlen opstod, og derved behandle fejlen. Se listen over fejlmeddelser i APPENDIKS VII.

Forbundne nøgleord: ON ERROR, ERROR

ERROR

ERROR «heltalsudtryk»

ERROR 17

KOMMANDO: Reagerer på et givet fejlnummer. Fejlen kan være en, som allerede er brugt af BASIC (Appendiks VIII), i så fald vil fejlen blive behandlet, som hvis den var opdaget af BASICen. Andre fejlnumre end de eksisterende, kan bruges af programmet til at signalere egne fejl.

Forbundne nøgleord:ON ERROR,ERR,ERL

EVERY

EVERY «heltalsudtryk»|,«heltalsudtryk»|GOSUB «linienummer»

EVERY 500,2 GOSUB 50

KOMMANDO: CPC464 har indbygget et sandtidsur. Kommandoen EVERY tillader et BASIC program at lade subrutiner kalde med bestemte intervaller. Der er fire timere til rådighed, som angives af det andet heltalsudtryk i området 0...3. Hver af disse kan have en subrutine forbundet med sig.

Forbundne nøgleord: AFTER,REMAIN

EXP

EXP («numerisk udtryk»)

PRINT EXP (6.876)

968.743625

FUNKTION: Beregner *e* opløftet til tallet i numerisk udtryk, hvor *e* er ca. 2.7182818. Tallet, hvis naturlige logaritme er 1.

Forbundne nøgleord:LOG

FIX

FIX («numerisk udtryk»)

PRINT FIX(9.99999)
9

FUNKTION: I modsætning til CINT vil FIX simpelthen fjerne decimalerne fra et tal og efterlade et heltal som resultat, idet tallet afrundes mod nul.

Forbundne nøgleord:CINT,INT,ROUND

FOR

FOR «simpel variabel» = «start» TO «slut» |«STEPstørrelse»|

FOR DAG = 1 TO 5 STEP 2

KOMMANDO:Udfører en programblok et givet antal gange, idet en kontrolvariabel løber i skridt mellem en start og en slutværdi. Hvis der ikke angives nogen værdi for STEP (skridt), bliver den sat til 1.

Forbundne nøgleord:NEXT, WHILE

FRE

FRE («numerisk udtryk»)

FRE («streng udtryk»)

PRINT FRE(0)

PRINT FRE('''')

FUNKTION: Fastslår, hvor megen lagerplads BASIC har ladet stå ubrugt i RAMen. Formen FRE('' '') fremtvinger en ''skraldindsamling'', før værdien af den ubrugte RAM-plads meddeles.

GOSUB

GOSUB «linienummer»

GOSUB 210

KOMMANDO: Kalder en BASIC subrutine ved at hoppe til det angivne linienummer. Se RETURN.

Forbundne nøgleord: RETURN

GOTO

GOTO «linienummer»

GOTO 90

KOMMANDO: Hop til det angivne linienummer.

HEX$

HEX$(«positivt heltal»)|,«heltals udtryk»|)

PRINT HEX$(65534)
FFFE

FUNKTION: Omformer det givne tal til hexadecimal form. (Se appendiks II).

Forbundne nøgleord: BIN$,STR$

HIMEM

HIMEM

?HIMEM
43903

VARIABEL: Giver adressen på den højeste byte i hukommelsen, som bruges af BASIC. Kan bruges i numeriske udtryk på sædvanlig måde.

Inden du flytter den højeste byte i hukommelsen med MEMORY kommandoen, er det tilrådeligt at skrive mm = HIMEM. Du vil da altid kunne få hukommelsen tilbage ved at skrive MEMORY mm.

Forbundne nøgleord: FRE, MEMORY

IF

IF

IF «logisk udtryk» THEN «valgfri del» |ELSE «valgfri del»|

IF «logisk udtryk» GOTO «lininummer» |ELSE «valgfri del»|

IF A<B THEN A = C ELSE A = D

IF A<B GOTO 1000 ELSE 300

KOMMANDO: En meget brugt kommando, som anvendes til betinget at bestemme delingspunkter i et program. Den logiske del bliver undersøgt, og hvis den er sand, så bliver THEN eller GOTO delen udført. Hvis den er falsk, bliver ELSE delen udført, eller bare næste linie i programmet. Sætninger kan bygges sammen til enhver dybde, kun begrænset af linielængden. IF slutter linien, det er ikke muligt at have flere sætninger, som er uafhængige af IF sætningen, på samme linie.

Forbundne nøgleord: THEN,ELSE,GOTO,OR,AND,WHILE

INK

INK «blæk», «farvenummer»|,«farve»|

INK 0,1

KOMMANDO: Afhængigt af den nuværende skærm MODE (kapitel 5), er et antal blæk til rådighed. Farven, eller farverne, som bruges til en INK sætning kan ændres af en INK kommando, ved at sammenligne med tabellen over farver i appendiks VI.

Forbundne nøgleord:PEN,PAPER

INKEY

INKEY («heltals udtryk»)

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

FUNKTION: Denne funktion aflæser tastaturet, for at fortælle hvilke taster, der er nedtrykket. Tastaturet aflæses hver 0.02 sek. Funktionen er særlig nyttig til at aflæse J/N svar fra brugeren, eftersom det ikke er nødvendigt at undersøge om |**SHIFT**| er trykket ned for at bruge svaret. Ovenstående eksempel undersøger om |**SHIFT**| og V er trykket ned sammen. |**SHIFT**| og |**CTRL**| kan findes ved følgende taster:

Værdi returneret	|**SHIFT**|	|**CTRL**|	**TAST**
-1	?	?	OP
0	OPPE	OPPE	NEDE
32	NEDE	OPPE	NEDE
128	OPPE	NEDE	NEDE
160	NEDE	NEDE	NEDE

Forbundne nøgleord:INPUT, INKEY$

INKEY$

INKEY$

```
10 CLS
20 PRINT "Er du klog (j eller n) ?"
30 A$ = INKEY$: IF A$ = "" GOTO 30
40 IF A$ = "N" OR A$ = "n" THEN
PRINT "Det maa du have vaeret for at koebe mig!":END
50 IF A$ = "J" OR A$ = "j" THEN
PRINT "Du er for beskeden!!!!":END
60 GOTO 20
```

FUNKTION: Læser tastaturet for at du kan lave dataforespørgsler uden at skulle trykke |**ENTER**| efter hvert svar. Hvis en tast er nedtrykket, så giver funktionen et svar, hvis ingen tast er nedtrykket, så giver funktionen en tom streng som svar. Denne kan bruges til at køre programmet i løkke indtil en rigtig tast nedtrykkes.

Forbundne nøgleord: INPUT, INKEY

INP

INP («port nummer»)

FUNKTION: Giver værdien af I/O porten med det angivne nummer.

Forbundne nøgleord: OUT, WAIT

INPUT

INPUT |#«strømudtryk»,||;||«streng med citationstegn»;| «liste af:|variabler|»

eller

INPUT |#strømudtryk»,||;||«streng med citationstegn»;| «liste af:|variabler|»

```
10 CLS
20 INPUT "GIV MIG TO TAL, ADSKILT MED KOMMA";A,B
30 IF A=B THEN PRINT "TALLENE ER ENS"
40 IF A>B THEN PRINT A;" ER MERE END ";B
50 IF A<B THEN PRINT A;" ER MINDRE END ";B
60 CLEAR: GOTO 20
```

KOMMANDO: Læser data fra den angivne strøm. Et semikolon efter INPUT fjerner det vognretur, som bliver indtastet efter den linie, som bliver indskrevet. Et semikolon efter strengen med citationstegn om udskriver et spørgsmålstegn. Et komma fjerner spørgsmålstegnet. Hvis der bliver indskrevet noget forkert (fx bogstavet O i et tal, i stedet for 0), så vil BASIC skrive:

?Redo from start (Skriv om forfra igen)

...og den tekst som du oprindelig indskrev.

Alle svar skal afsluttes af |**ENTER**|. Semikolonnet umiddelbart efter «strøm» undertrykker vognreturen ved afslutningen af linien, der indskrives, og derfor bliver markøren ved afslutningen af den indskrevne tekst. Når du bruger en kassettestrøm, bliver der ikke skrevet noget spørgsmål. Hvis du har skrevet et, i INPUT sætningen, vil den blive ignoreret af kassettesoftwaren, så samme program kan læse fra enhver strøm.

Der vil blive læst et element fra filen, for hver variabel i listen. Det, der læses, skal passe til den variabel, der skal læses ind i. Det kan være: En numerisk variabel, afsluttet enten af komma eller af vognretur, ingenting (mellemrum) eller filslut EOF. Kommaer eller |**ENTER**| sendt efter mellemrum, vil blive ignoreret. Strenge med citationstegn om, vil blive læst til de afsluttes af ", efterfølgende karakterer vil blive ignoreret, som for numeriske variabler. Strenge uden citationstegn om afsluttes som omtalt under tal.

Forbundne nøgleord: LINE INPUT, READ, INKEY$

INSTR

INSTR(|«heltalsudtryk»,|«strengudtryk»,«streng udtryk»)

PRINT INSTR(2,"BANAN","AN")

FUNKTION: Søger i det første strengudtryk efter den første optræden af det andet strengudtryk, hvor et heltalsudtryk kan angive stedet, hvor søgningen skal starte - ellers begynder søgningen med første karakter i første strengudtryk.

Forbundne nøgleord: MID$,LEFT$, RIGHT$

INT

INT («numerisk udtryk»)

PRINT INT(-1.995)
-2

FUNKTION: Afrunder tallet til det nærmeste mindre heltal, idet decimalerne smides væk. Virker som FIX på positive tal, men returnere en mindre end FIX ved negative tal, som ikke allerede er heltal.

Forbundne nøgleord: CINT,FIX,ROUND

JOY

JOY(«heltalsudtryk»)

10 IF JOY(0) = 8 THEN GOTO 100

FUNKTION: JOY læser tilstanden af styrepind nummer «heltalsudtryk», og giver et resultat, som skal betragtes som bits.

Bit	Decimal
0:OP	1
1:NED	2
2:VENSTRE	4
3:HØJRE	8
4:FYR 2	16
5:FYR 1	32

Forbundne nøgleord:INKEY

KEY

KEY «heltalsudtryk»,|CHR$(n) + |«strengudtryk»| + CHR$(n)|

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

KOMMANDO: Definerer en ny funktion for en tast. Der er 32 udvidelseskarakter i området 128-159 skrevet op i Appendiks III. Når en af disse karakterer læses, bliver den udvidet til strengen (op til 32 karakterer), som forbindes med tasten. Der kan ikke være mere end 120 udvidelseskarakterer ialt, for alle taster. KEY kommandoen forbinder en streng med en given karakter.

Forbundne nøgleord: KEY DEF

KEY DEF

KEY DEF«tastnummer»,«gentagelse»|,«normal»|,«skiftet» |,«kontrol»|||

KEY DEF 46,1,63

KOMMANDO: Konverterer værdien, som laves af en tast på tasturet som defineret i Appendiks III. Ovenstående eksempel konverterer N tasten til at skrive spørgsmålstegns karakteren ?. (Karakter nummer 63).

For at få tasten tilbage til normal tilstand:

KEY DEF 46,1,110

hvor karakter nummer 110 er et lille n.

Forbundne nøgleord: KEY

LEFT$

LEFT$(«strengudtryk»,«heltalsudtryk»)

```
10 CLS
20 A$ = " AMSTRAD"
30 B$ = LEFT$(A$,3)
40 PRINT B$
RUN
|SKÆRMEN SLETTES|
AMS
Ready
```

FUNKTION:Trækker karakterer til venstre for, og incl. positionen heltalsudtryk, fra det givne strengudtryk. Hvis strengudtrykket er for kort, bliver hele strengudtrykket returneret.

Forbundne nøgleord: MID$,RIGTH$

LEN

LEN(«strengudtryk»)

A$ = ''AMSTRAD'':PRINT LEN(A$)

FUNKTION: Returnerer det samlede antal karakterer i en streng.

LET

LET «logisk udtryk»

LET X = 100

KOMMANDO: En rest fra dengang BASIC var ung, og tildeling af variabler skulle forudses. Der er ingen grund til at bruge LET, undtagen for at give sammenlignelighed med programmer fra gamle BASIC-manualer. Du behøver kun at skrive ovenstående således:

X = 100

i AMSTRAD BASIC.

LINE INPUT

LINE INPUT |«#«strømudtryk»,||#||[streng med citationstegn om;|«strengvariabel»

LINE INPUT A$
LINE INPUT ''NAVN'';N$

KOMMANDO: Læser en hel linie fra den angivne strøm. Et semikolon efter LINE INPUT undertrykker ekkoet af vognretur/nylinie. Hvis du ikke skriver noget strømnummer, vil datamaten antage strøm #0, = selve datamaten.

Forbundne nøgleord: READ,INPUT,INKEY$,INPUT$

LIST

LIST |«linienummerområde»||;||#strengudtryk|

LIST 100-1000,#1

KOMMANDO: Udskriver programmet til den angivne strøm. Strøm 0 bruges, hvis du ikke angiver nogen strøm, 8 er printeren. LISTning kan afbrydes ved at trykke |**ESC**| en gang, og genoptages ved at trykke på mellemrumstangenten. To gange |**ESC**| vil stoppe udlistningen helt. Programmer kan listes ud til foruddefinerede vinduer, for at hjælpe med fejlfinding i programmerne uden at skrive over hele skærmen. Listning kan udføres fra starten af programmet, til et givet punkt, eller fra et givet punkt til slutningen af programmet ved at udelade første eller sidste tal i linienummerområdet fx:

LIST -200 eller LIST 30-

LOAD

LOAD «filnavn»|,«adresseudtryk»|

LOAD "REGISTER"

KOMMANDO: At læse et BASIC program fra kassette ind i hukommelsen, idet et eksisterende program slettes, eller hvis det valgfrie adresseudtryk bruges, at LOADe en binær fil ind i hukommelsen. Se kapitel 2.

LOCATE

LOCATE |#«strømudtryk»,|«xkoordinat»,«ykoordinat»

```
10 MODE 1
20 LOCATE 20,12
30 PRINT CHR$(249)
```

KOMMANDO: Flytter tekstmarkøren til en ny position, som er relativ til det nuværende vindues nulpunkt. Øverste venstre hjørne af vinduet er 1,1, og hvis du ikke skriver noget, er det der hvor hele skærmen er vinduet.

Forbundne nøgleord: WINDOW

LOG

LOG («numerisk udtryk»)

```
?LOG(9999)
9.21024037
```

FUNKTION: Beregner den naturlige logaritme til «numerisk udtryk» og returnerer et reelt resultat, selvom numeriskudtryk kan være et heltal.

Forbundne nøgleord: EXP, LOG10

LOG10

LOG10(«numerisk udtryk»)

?LOG10(9999)
3.99995657

FUNKTION: Beregner titalslogaritmen til «numerisk udtryk» og returnerer denne som et reelt resultat, selvom argumentet var et heltal.

Forbundne nøgleord: EXP,LOG

LOWER$

LOWER$(«strengudtryk»)

A$ = "AMSTRAD":PRINT LOWER$(A$)
amstrad

FUNKTION: Laver streng udtrykket om til små bogstaver, hvor der før var store bogstaver. Nyttig til indlæsninger hvor der kan komme både store og små bogstaver med.

Forbundne nøgleord: UPPER$

MAX

MAX («liste af:«numerisk udtryk»)

10 n = 66
20 PRINT MAX(1,n,3,6,4,3)

FUNKTION: Returnerer den største værdi fra en række af numeriske udtryk.

Forbundne nøgleord: MIN

MEMORY

MEMORY «adresseudtryk»

MEMORY &20AA

KOMMANDO: Ændrer den højeste byte i hukommelsen som BASIC kan bruge, til en ny adresse. Se beskrivelsen af nøgleordet HIMEM. For at checke hukommelsen kan du bruge FRE kommandoen.

Forbundne nøgleord: HIMEM,FRE

MERGE

MERGE |«filnavn»|

MERGE "PLAN"

KOMMANDO: Læs et program ind fra kassette, og læg det til det eksisterende program. Hvis du ikke angiver et filnavn, så tager BASIC den første brugbare fil på båndet. Hvis første karakter i filnavnet er et !, så vil dette blive fjernet, og de sædvanlige beskeder vil blive udeladt.

Hvis du vil læse et program ind i hukommelsen, uden at slette det eksisterende program, så skal det eksisterende program RENUMereres til et linienummerområde over det indkommende program.

Alle variabler, brugerfunktioner og åbne filer fjernes. ON ERROR GOTO slås fra, en RESTORE bliver udført og DEFINT,DEFREAL og DEFSTR indstillingerne slås fra. Beskyttede filer kan ikke MERGEs.

Forbundne nøgleord: LOAD, CHAIN,CHAIN MERGE

MID$

MID$(«streng»,«heltalsudtryk»|,«heltalsudtryk»|)

A$ = "AMSTRAD":PRINT MID$(A$,2,4)
MSTR

A$ = "AMSTRAD":B$ = MID(A$,2,4):PRINT B$
MSTR

KOMMANDO og FUNKTION:MID$ finder en del af en streng, som kan bruges enten til modtager af en tildeling, eller som argument i et strengudtryk (MID$ som funktion). Første heltalsudtryk angiver positionen af første karakter. Andet heltalsudtryk angiver længden af den nye streng. Hvis længden udelades, vil det blive hele resten af den originale streng.

Forbundne nøgleord: LEFT$,RIGHT$

MIN

MIN («liste af:«numerisk udtryk»)

PRINT MIN(3,6,2.999,8,9)
2.999

FUNKTION: Det modsatte af MAX.

Forbundne nøgleord: MAX

MODE

MODE «heltalsudtryk»

MODE 1

KOMMANDO: Skift skærm udseende (0,1 eller 2), og slet skærmen til INK 0, som ikke behøver at være den anvendte PAPER farve. Alle tekst og grafik vinduer er set i forhold til hele skærmen, og deres markører flyttes hjem til deres nulpunkter.

Forbundne nøgleord: WINDOW,ORIGIN

MOVE

MOVE «xkoordinat»,«ykoordinat»

MOVE 34,34

KOMMANDO: Flyt grafikmarkøren til et sted angivet i absolutte koordinater. YPOS og XPOS er de tilsvarende funktioner til at fastslå placeringen af markøren.

Forbundne nøgleord:
MOVER,PLOT,PLOTR,DRAW,DRAWR,ORIGIN,TEST,TESTR, XPOS,YPOS

MOVER

MOVER «xafstand»,«yafstand»

MOVER 34,34

KOMMANDO: Flytter grafikmarkøren til en position relativt til de nuværende koordinater. XPOS og YPOS kan bruges til at finde markørens position.

Forbundne nøgleord:
MOVE,PLOT,PLOTR,DRAW,DRAWR,TEST,TESTR,XPOS,YPOS

NEW

NEW

NEW

KOMMANDO: Slet det nuværende program og variabler. KEY definitioner bibeholdes, og skærmen slettes ikke. Det mest uhelbredelige stadium, næstefter en total reset eller slukning. Brug med forsigtighed.

NEXT

NEXT |«liste af:«variable»|

FOR N = 1 TO 1000: NEXT

KOMMANDO: Angiver slutningen på en FOR-løkke. NEXT kommandoen kan være anonym, eller den kan referere til dens tilsvarende FOR. I ovenstående eksempel ville det være:

NEXT N

Forbundne nøgleord:FOR

ON GOSUB
ON GOTO

ON «heltalsudtryk» GOSUB «liste af:«linienummer»
ON «heltalsudtryk» GOTO «liste af:«linienummer»

10 ON DAG GOSUB 100,200,300,400,500
20 ON GRAD GOTO 1000,2000,3000,4000

KOMMANDO: GOSUB til subrutinen, eller GOTO sætningen, efter hvad tallet i heltalsudtrykket er. Når tallet er 1 gås til første linienummer, 2 til andet linienummer etc. I ovenstående linie 10 vil subrutinen på linie 100 blive besøgt, når DAG = 1. DAG = 2 vil kalde linie 200 etc.

Forbundne nøgleord: GOTO,GOSUB

ON BREAK GOSUB

ON BREAK GOSUB «linienummer»

10 ON BREAK GOSUB 40
20 PRINT "PROGRAMMET ER I GANG"
30 GOTO 20
40 CLS
50 PRINT "VED DOBBELT ESC-TRYK KALDES DENNE RUTINE"
60 FOR T = 1 TO 2000:NEXT
70 RETURN

KOMMANDO:Kalder en subrutine, når programmet forsøges stoppet ved at trykke |**ESC**| to gange.

Forbundne nøgleord: ON BREAK STOP, RETURN

ON BREAK STOP

ON BREAK STOP

```
10 ON BREAK GOSUB 40
20 PRINT "PROGRAMMET ER I GANG"
30 GOTO 20
40 CLS
50 PRINT "HVIS DU TRYKKER ESC TO GANGE
KALDES GOSUB RUTINEN"
60 FOR T = 1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
```

KOMMANDO: Når kommandoen gives i en ON BREAK subrutine, vil ON BREAK slå fælden fra. I ovenstående program, som indeholder ON BREAK STOP, vil ON BREAK GOSUB fælden kun virke en gang.

Forbundne nøgleord: ON BREAK GOSUB

ON ERROR GOTO

ON ERROR GOTO «linienummer»

```
10 ON ERROR GOTO 80
20 CLS
30 PRINT"Hvis der er en fejl, vil jeg"
40 PRINT"godt have programmet listet,"
50 PRINT"saa jeg kan se hvad der gik galt"
60 FOR T = 1 to 4000:next
70 GOTO 200
80 CLS:PRINT"DER ER EN FEJL I LINIE";ERL:PRINT
90 LIST
```

KOMMANDO: Går til en specifik linie i programmet, når der opstår en fejl. I dette eksempel opstår der en fejl i linie 70.

Forbundne nøgleord:ERR,ERL,RESUME

ON SQ GOSUB

ON SQ («kanal») GOSUB «linienummer»

ON SQ (2) GOSUB 2000

KOMMANDO: Tillad et interrupt når der er plads i en given lydkø. «kanal» er et heltalsudtryk med en af værdierne:

1: for kanal A
2: for kanal B
4: for kanal C

Forbundne nøgleord:SOUND, SQ

OPENIN

OPENIN «filnavn»

100 OPENIN "!INFORMATION"

KOMMANDO: Åbner en indlæsningsfil fra kassette, som indeholder information til brug for det nuværende program i datamatens hukommelse.

Hvis første karakter i filnavnet er et !, så vil der ikke komme beskeder på skærmen fra kassette processen, og programmet læser første blok fra kassetten ind, klar til behandling.

Forbundne nøgleord:CLOSEIN,OPENOUT

OPENOUT

OPENOUT «filnavn»

OPENOUT "!FACTS"

KOMMANDO: Åbner en udskrivningfil til kassette, til brug for det nuværende program. Hvis første karakter i filnavnet er !, så vil beskeder fra kassette-processen ikke blive skrevet på skærmen, og programmet skaber den første blok af data, idet der oprettes en fil med det givne navn. Hver blok består af 2K af data, og der bliver ikke skrevet noget ud for der er samlet 2K sammen, eller filen lukkes med CLOSEOUT etc.

NB en NEW kommando vil slette en åben fil, og dataene vil være tabt.

Forbundne nøgleord:CLOSEOUT,OPENIN

ORIGIN

ORIGIN «x»,«y»|,«venstre»,«højre»,«top»,«bund»|

```
10 CLS:BORDER 13
20 ORIGIN 0,0,50,590,350,50
30 DRAW 540,350
40 GOTO 20
```

KOMMANDO: Bestemmer startpunktet for grafik markøren. Den |valgfrie del| af kommandoen indeholder instruktioner til at sætte et nyt grafikvindue, som vil være virksomt i alle skærm MODEs på grund af adresseringsteknikken.

ORIGIN er punktet med koordinaterne 0,0 i venstre sides bund af vinduet. Koordinaterne vokser opad og til højre. Se Appendiks VI for grafikgitter oversigt.

Hvis et vindues hjørner angives til en position uden for skærmen, så antages de at være den yderste synlige position i den givne retning.

Forbundne nøgleord:WINDOW

OUT

OUT «port nummer»,«heltalsudtryk»

OUT &F8F4,10

Kommando: Sender værdien i heltalsudtrykket (som skal være mellem 0 og 255) ud på det angivne portnummer.

Forbundne nøgleord:INP,WAIT

PAPER

PAPER [#«strømudtryk»,]«dækket blæk»

```
10 MODE 0
20 FOR P=0 TO 15
30 PAPER P:CLS
40 PEN 15-P
50 LOCATE 6,12: PRINT"PAPIR";P
60 FOR T=1 TO 500:NEXT T
70 NEXT P
```

KOMMANDO: Sætter baggrunds blækket for karakterer. Når karakterer udskrives til tekst skærmen, bliver karaktercellen fyldt med PAPER blækket, før karakteren bliver indskrevet - medmindre transparent udskrift er valgt.

Antallet af farver, som kan vises samtidig afhænger af hvilket skærm MODE man er i. Hvis «dækket blæk» ikke er tilgængelig, som ønsket, vil skærmen bruge en tilgængelig farve.

Forbundne nøgleord: INK,WINDOW,PEN

PEEK

PEEK («adresseudtryk»)

```
10 MODE 2
20 INK 1,0:INK 0,12:BORDER 12
30 INPUT "Startadresse for undersoegelse";first
40 INPUT "Slutadresse for undersoegelse";last
50 FOR N=first TO last
60 VALUE$=HEX$(PEEK(N),2)
70 PRINT VALUE$
80 PRINT " paa";HEX$(N,4),
90 NEXT
```

FUNKTION: Læs indholdet af en byte. Programmet ovenfor lader dig se gennem RAMen på CPC464. Det læser RAM under den lavere (&0000-&3FFFH) og øvre (&C000-FFFF) ROM, ikke ROMen.

Forbundne nøgleord: POKE

PEN

PEN |«#«strømudtryk»,|«dækket blæk»

PEN 1,2

KOMMANDO: PEN sætter den farve, der skal bruges, når der bliver tegnet på den givne skærmstrøm, strøm #0 hvis ikke andet er angivet.

Forbundne nøgleord: 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 2000
80 ORIGIN 420,0
90 DRAW 0,200
100 REM TEGN VINKLER FRA FORSVINDINGSPUNKT
110 DRAW 30*N*SIN(N*PI/4),(SIN(PI/2))*N*SIN(N)
120 NEXT
130 MOVE 0,200
140 DRAWR 0,50
150 DRAWR 40,0
160 WINDOW 1,40,1,10
170 PRINT "Nu kan du goere
hangmanprogrammet faerdig"
```

FUNKTION: Værdien af forholdet mellem omkredsen og diametren af en cirkel. Den tættest mulige maskintilnærmelse (3.141592653468251). Bruges meget i grafikrutiner som den ovenfor.

Forbundne nøgleord: DEG,RAD

PLOT

PLOT «xkoordinat»,«ykoordinat»|,«dækket blæk»|

```
10 MODE 2:PRINT "INDTAST 4 TAL,
ADSKILT MED KOMMAER"
20 PRINT "GIV X NULPUNKT (0-639),
Y NULPUNKT (0-399), RADIUS OG
VINKELINDDELING":INPUT X,Y,R,S
30 ORIGIN X,Y
40 FOR VINKEL = 1 TO 360 STEP S
50 XPUNKT = R * COS(VINKEL)
60 YPUNKT = R * SIN(VINKEL)
70 PLOT XPUNKT,YPUNKT
74 REM MOVE 0,0
75 REM DRAW XPUNKT,YPUNKT
80 NEXT
```

KOMMANDO: Prøv 320,200,20,1 som din første reaktion. PLOT er det samme som DRAW, bortset fra at der kun tegnes et punkt. Hvis du fjerner REM i linie 75 ovenfor og sætter REM foran linie 70, for at fjerne dens virkning, kan du se forskellen. (Fjern REM i linie 74, hvis cirklen skal fyldes ud).

Læg mærke til at processen fylder cirklen ud udefra, ved hele tiden at løbe rundt langs periferien. Husk at dette program ikke har stillet væk fra radianer RAD, så vinklen i hvert trin er betydeligt mere end en grad. Sæt linien 25 DEG ind, og kør programmet.

Forbundne nøgleord:
DRAW,DRAWR,PLOT,PLOTR,MOVE,MOVER,ORIGIN,TEST,
TESTR,XPOS,YPOS

PLOTR

PLOTR «xkoordinat»,«ykoordinat»|«dækket blæk»|

```
10 MODE 2:PRINT "INDTAST 4 TAL, ADSKILT
MED KOMMAER":PRINT
20 PRINT "INDTAST X NULPUNKT (0-639),
Y NULPUNKT (0-399), RADIUS OG
VINKELSPRING":INPUT X,Y,R,S
30 ORIGIN X,Y
40 FOR VINKEL = 1 TO 360 STEP S
50 XPUNKT = R * COS(VINKEL)
60 YPUNKT = R * SIN(VINKEL)
70 PLOTR XPUNKT,YPUNKT
80 NEXT:GOTO 40
```

KOMMANDO: Prøv 320,0,20,1 i reaktion. PLOTR er det samme som DRAWR bortset fra at plot kun tegner et punkt.

Forbundne nøgleord:
DRAW,DRAWR,PLOT,PLOTR,MOVE,MOVER,ORIGIN,TEST,TESTR,XPOS,YPOS

POKE

POKE «adresseudtryk»,«heltalsudtryk»

POKE &00FF,10

KOMMANDO: Giver direkte adgang til maskinens hukommelse, og læser heltalsudtrykket i området 0..255 direkte ind i den angivne adresse. Det er nemt at få datamaten til at gå ned, hvis man ikke ved hvad man gør.

Forbundne nøgleord: PEEK

POS

POS («#«strømudtryk»)

PRINT POS(#0)
1

FUNKTION: Fastslår den nuværende position for en given strøm. I dette tilfælde er der intet tilbagefald til «strømudtryk» og udeladelse af det vil resultere i en Syntax error melding.

Skærmen: Returnerer markørens nuværende X-koordinat, relativt til det nuværende vindue, hvor dette er øverste venstre hjørne, repræsenteret som 1,1

Printer: Returnerer vognpositionen, hvor 1 er venstre margen. Alle tegn med ASCII-numre større end &1F(31) er inkluderet.

Kassette-strøm: Som ved printeren.
Forbundne nøgleord: VPOS

PRINT

PRINT |#«strømudtryk»,||«printliste»||«USING klausul»||«separator»|

PRINT #0,"ABC"

KOMMANDO: For en fuld forklaring skal du se i slutningen af dette afsnit.

Forbundne nøgleord:USING,TAB,SPC

RAD

RAD

RAD

KOMMANDO: Sætter datamaten til at regne i radianer.

Forbundne nøgleord: DEG,SIN,COS,TAN,ATN

RANDOMIZE

RANDOMIZE |«numerisk udtryk»|

```
10 RANDOMIZE 23
20 PRINT RND(6)
```

KOMMANDO: BASICs tilfældigtalsgenerator giver kun tilsyneladende tilfældige tal. Tallene afhænger altid af det forrige tal, hvis man starter fra et givet tal, vil rækkefølgen af tilfældige tal altid være den samme. RANDOMIZE sætter en ny begyndelsesværdi for tilfældigtals generatoren. RANDOMIZE TIME vil give en rækkefølge der er svær at gentage.

Forbundne nøgleord: RND

READ

READ list af:«variable»

```
10 FOR X=1 TO 4
20 READ N$
30 PRINT N$
40 DATA ADAM,DANNY,JAMES,RICHARD
50 NEXT
```

KOMMANDO: READ henter data fra listen, angivet efter de tilsvarende DATA sætninger, og tildeler dem til variabler, og automatisk hoppe videre til næste element. RESTORE vil returnere pilen til begyndelsen af DATA sætningerne igen. Se nøglordet DATA.

Forbundne nøgleord: DATA,RESTORE

RELEASE

RELEASE «lydkanaler»

RELEASE 4

KOMMANDO: Når en lyd placeres i lydkøen, kan den indeholde en hold ordre. Hvis en kanal er holdt, så bliver den frigivet af denne ordre. Udtrykket, der giver lydkanalnumrene er bitsignifikant: A = bit 0, B = bit 1,C = bit 2. Derved vil 4 (binært 0100) frigive kanal C.

Forbundne nøgleord: SOUND

REM

REM «resten af linien»

10 REM Intergalaktisk Hyperrums Monster
Invasion og Dommedagsjagt
20 REM Copyright AMSOFT 1984

KOMMANDO: Bruges til at placere bemærkninger rundt omkring i et program. De har ingen indflydelse på udførelsen af programmet. En enkelt apostrof ' virker som :REM undtagen i DATA sætninger, hvor det opfattes som en del af en streng uden citationtegn om.

REMAIN

REMAIN («heltalsudtryk»)

REMAIN (3)

PRINT #6,REMAIN(0);

FUNKTION: Slår en timer fra (i området 0..3). Læser den tilbageværende tid på en timer. Der kommer nul tilbage, hvis timeren ikke var sat til noget.

Forbundne nøgleord: AFTER,EVERY

RENUM

RENUM |«nylinienumre»||,|«gammelt linienummer»| |,«tillæg»||

RENUM
RENUM 100,,100

KOMMANDO: Giv nye numre til programlinierne fra «gammelt linienummer» med det angivne tillæg. «Nylinienummer» angiver det første linienummer for den nye nummerrække. «gammelt linienummer» angiver hvilken gammel linie, der skal have det første nye linienummer. Tillægget bliver sat til 10 hvis det ikke angives.

RENUM tager sig af alle GOTO, GOSUB o.s.v. Hvis man ikke skriver andet end RENUM, bliver alle linier renummereret fra 10 i spring på 10.

Linienumre op til 65535 kan bruges.

RESTORE

RESTORE «linienummer»

RESTORE 300

```
10 FOR N=1 TO 5
20 READ A$
30 PRINT A$ " ";
40 DATA restorede,data,kan,findes,igen
50 NEXT
60 PRINT
70 RESTORE
80 GOTO 10
```

KOMMANDO: Sætter læsepilens position tilbage til begyndelsen af den DATA sætning som linienumret angiver. Hvis man ikke giver et linienummer, bruges den første DATA sætning.

Forbundne nøgleord: READ,DATA

RESUME

RESUME |«linienummer»|
eller RESUME NEXT

RESUME 300

KOMMANDO: Når fejl i ens program er blevet fanget af en ON ERROR GOTO kommando, og er blevet behandlet, tillader RESUME at programmet genoptages, idet man kan angive fra hvilket lininummer programmet skal fortsætte.

Forbundne nøgleord: ON ERROR GOTO

RETURN

RETURN

RETURN

KOMMANDO: Angiver slutningen på en BASIC subrutine. BASIC returnerer fra denne, og genoptager programmet fra instruktionen lige efter den GOSUB-ordre som kaldte subrutinen.

Forbundne nøgleord: GOSUB, ON x GOSUB, ON SQ GOSUB, AFTERR n GOSUB, EVERY n GOSUB, ON BREAK GOSUB

RIGHT$

RIGHT$(«strengudtryk»,«heltalsudtryk»)

```
10 CLS
20 A$ = ''AMSTRAD''
30 B$ = RIGTH$(A$,3)
40 PRINT B$
RUN
```
skærmen slettes
```
RAD
Ready
```

FUNKTION: Trækker karakterer til højre for og incl. den angivne position ud af strengen. Hvis strengudtrykket er kortere end den nødvendige længde, bliver hele den oprindelige streng returneret.

Forbundne nøgleord: MID$,LEFT$

RND

RND|(«numerisk udtryk»)|

```
10 RANDOMIZE 23
20 PRINT RND(6)
```

FUNKTION: Finder et tilfældigt tal, som kan være det næste i rækken, en gentagelse af det sidste, eller det første i en ny rækkefølge. RANDOMIZE kommandoen i ovenstående program sikrer at RND(6) returnerer det samme hver gang (0.146940658).

RND(0) giver en kopi af det forrige tal. Hvor det numeriske udtryk er negativt, vil talfølgen være forudseelig.

Forbundne nøgleord: RANDOMIZE

ROUND

ROUND («numerisk udtryk»|,«heltals udtryk»|

```
10 X = 0.123456789
20 FOR R = 9 TO 0 STEP -1: PRINT R,ROUND(X,R):NEXT
25 X = 123456789
30 FOR R = 0 TO -9 STEP -1
40 PRINT R,ROUND( X,R)
50 NEXT
```

FUNKTION: Afrunder alle numeriske udtryk til et antal decimaler eller en tierpotens, angivet af heltalsudtrykket. Hvis heltalsudtrykket er mindre end nul, så afrundes værdien til at give et heltal, efterfulgt af et antal nuller, angivet af heltalsudtrykket, før decimalpunktummet.

Forbundne nøgleord: INT,FIX,CINT,ABS

RUN

RUN «strengudtryk»

RUN ''VELKOMMEN''

KOMMANDO: Læs et program fra kassetten og start udførelsen af det. Hvis strengen er tom '''', så vil første program, der mødes blive brugt. Hvis første karakter er et !, så vil der ikke blive skrevet beskeder på skærmen under indlæsningen.

NB: BASIC udfører et NEW umiddelbart efter at et program er fundet på båndet, således at hukommelsen er tom, inden indlæsningen.

Forbundne nøgleord: LOAD

RUN

RUN |«linienummer»|

RUN 100

KOMMANDO: Starter udførelsen af programmet fra den angivne linie, eller fra begyndelsen, hvis der ikke angives noget linienummer. Alle variabler og brugerfunktioner slettes. DEFINT,DEFREAL og DEFSTR bliver nulstillet. Alle kassettefiler smides væk, og eventuelle lagrede uddata tabes.

Forbundne nøgleord:LOAD

SAVE

SAVE «filnavn»|,«filtype»||,binære parametre»|

SAVE "PROG",P

KOMMANDO: Gem programmet med navnet «filnavn».

,A gemmer programmet som ASCII kode
,P beskytter programmet
,B gemmer programmet som binær fil, fx skærmen.

Forbundne nøgleord: LOAD,RUN «filnavn», MERGE,CHAIN, CHAIN MERGE

SGN

SGN («numerisk udtryk»)

```
10 INPUT "HVAD HAR DU PAA BANKBOGEN";KASSE
20 IF SGN(KASSE)<1 GOTO 30 ELSE 40
30 PRINT "AK,AK":END
40 PRINT "DU HAR FLERE PENGE END MIG"
```

FUNKTION: Bestemmer fortegnet af et numerisk udtryk. Returnerer -1, hvis tallet er negativt, 0 hvis tallet er 0 og returnerer 1, hvis tallet er større end 0.

Forbundne nøgleord: ABS

SIN

SIN («numerisk udtryk|

```
PRINT SIN(PI/2)
1
```

FUNKTION: Beregner den reelle værdi af sinus til det numeriske udtryk. Der bruges radianer, hvis ikke DEG kommandoen er blevet udført.

Forbundne nøgleord: COS,TAN,ATN,DEG,RAD

SOUND

SOUND «kanalstatus»,«toneperiode«
|,«varighed» |,«styrke»|,«styrkeenvelope» |,«toneenvelope» |,«støjperiode»|||||

SOUND 1,200,1000,7,0,0,1

KOMMANDO: SOUND er en af de mest komplekse dele af AMSTRAD BASIC, og beskrives i kapitel 6.

Forbundne nøgleord: ENV,ENT

SPACE$

SPACE$(«heltalsudtryk»)

SPACE$(190)

FUNKTION: Laver en streng af mellemrum med den givne længde.

Forbundne nøgleord:PRINT,SPC,TAB

SPEED INK

SPEED INK «heltalsudtryk»,«heltalsudtryk»

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

KOMMANDO: INK og BORDER kommandoerne tillader to farver pr. blæk. I sådanne tilfælde skifter INKen mellem de to farver. Det første heltal angiver den tiden for første farve, det andet angiver tiden for den anden farve. Tiderne imellem farveskiftene måles i 0.02 sek. Du skal vælge omhyggeligt, ellers kan du komme ud for at virkningen nærmest bliver frustrerende.

Forbundne nøgleord:INK,BORDER

SPEED KEY

SPEED KEY «start pause»,«gentagelseshastighed»

SPEED KEY 20,3

KOMMANDO: Hvis en tast holdes nede vedvarende, vil den gentage sig selv efter et stykke tid. Dette er «startpausen». Gentagelsen vil foregå med «gentagelseshastigheden». Tiderne angives i perioder på 0.02 sek i området 1...255. Normalt er tallene 10,10.

Meget små startpauser vil blande sig sammen med tastaturlæserutinerne. Hastigheden, som tastaturet aflæses med ændres ikke af denne kommando.

Ikke alle taster repeteres. KEY DEF kommandoen tillader brugeren at ændre en given tasts karakteristika.

Forbundne nøgleord:KEY DEF

SPEED WRITE

SPEED WRITE «heltalsudtryk»

SPEED WRITE 1

KOMMANDO: Kassetten kan arbejde med enten 1000 eller 2000 bauds overføringshastighed. Hvis heltalsudtrykket er 1 er hastigheden 2000 baud. Normalt bruges 1000 baud. Når en fil indlæses fra bånd, vil CPC464 automatisk vælge den rigtige hastighed, derfor behøver brugeren ikke at gøre dette.

Når du bruger bånd af lav kvalitet, bør du bruge 1000 baud, for at få størst pålidelighed. Se bemærkningerne i kapitel 2.

Forbundne nøgleord: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 funktionen bruges til at checke antallet af frie pladser i lydkøen til en given kanal, hvor kanal A er 1, B er 2 og C er 3. Funktionen undersøger om kanalen er aktiv - og hvis ikke - hvorfor det første element i køen ikke arbejder. Resultatet skal læses bitvis.

0,1,2 angiver antallet af frie pladser i køen.
3,4,5 angiver et møde på vej i kanalen (hvis der kommer et)
6 sættes hvis køen holdes
7 sættes hvis kanalen arbejder

Forbundne nøgleord: SOUND, ON SQ GOSUB

SQR

SQR («numerisk udtryk»)

```
PRINT SQR(9)
3
```

FUNKTION: Returnerer kvadratroden af numerisk udtryk.

Forbundne nøgleord:PRINT

STOP

STOP

```
300 IF N<0 THEN STOP
```

KOMMANDO: Stopper programmet, men sørger for at programmet kan startes igen efter STOP kommandoen. Dette kan bruges til at afbryde programmet på et givet punkt, når man laver fejlfinding.

Forbundne nøgleord: CONT,END

STR$

STR$(«numerisk udtryk»)

```
PRINT STR$(&766)
1894

PRINT STR$(&X1010100)
84
```

FUNKTION: Forvandler numerisk udtryk til en decimal streng representation i samme form som brugt i PRINT kommandoen.

Forbundne nøgleord:VAL,PRINT,HEX$,BIN$

STRING$

STRING$(«heltalsudtryk»,«karakternummer»)

```
PRINT STRING$(&16,"★")
★★★★★★★★★★★★★★★★
```

FUNKTION: Leverer et strengudtryk bestående af den specificerede karakter gentaget et antal gange.

Forbundne nøgleord:SPACE$

SYMBOL

SYMBOL «karakternummer»,liste af:«række»

```
5 MODE 2
10 SYMBOL AFTER 90
20 SYMBOL 93,&80,&40,&20,&10,&8,&4,&2,&1
30 FOR N=1 TO 2000
40 PRINT CHR$(93);
50 NEXT
60 GOTO 60
```

KOMMANDO: SYMBOL kommandoen ændrer udseendet på en karakter, som først er blevet givet i SYMBOL AFTER kommandoen. Karakternummeret vælges ud af det tilgængelige ASCII karaktersæt eller andre karakterer fra CPC464s karaktersæt, og de følgende tal angiver udseendet af den nye karakter i en 8★8 punkts matrice.

Et nul i rækken i et 8 bit binært tal indikerer, at PAPERfarven skal bruges, og et 1 indikerer at punktet skal have INK farve. Se også Appendiks II og III. Eksemplet ovenfor laver en skråstreg, der går diagonalt gennem karakteren, og som kan hentes under | tasten.

Forbundne nøgleord:SYMBOL AFTER

SYMBOL AFTER

SYMBOL AFTER «heltalsudtryk»

SYMBOL AFTER 90

KOMMANDO: Antallet af brugerdefinerbare karakterer sættes med SYMBOL AFTER kommandoen. Normalt er denne 240, som giver 16 brugerdefinerbare karakterer. Hvis heltals udtrykket er 32, så vil alle karakterer fra 32 til 255 være omdefinerbare.

Når SYMBOL AFTER kommandoen bruges, bliver alle brugedefinerbare karakterer sat til deres standardudseende.

Forbundne nøgleord: SYMBOL

TAG

TAG |#«strømudtryk»|

```
10 MODE 2
11 BORDER 9
14 INK 0,12
15 INK 1,0
20 FOR N=1 TO 100
30 MOVE 200+N,320+N
40 TAG
50 IF N<70 THEN GOTO 60 ELSE 70
60 PRINT "HALLO";:GOTO 80
70 PRINT "FARVEL";
80 NEXT
90 GOTO 20
```

KOMMANDO: Tekst sendt til en given strøm kan omstyres til at blive skrevet ud på grafikmarkørens plads. Dette tillader tekst at blive blandet sammen med grafik. Strømudtrykket sættes til, hvis det udelades. Det øverste venstre hjørne af karaktercellen bliver "hægtet fast" på grafik markøren, og ikke-udskrivbare kontrolkarakterer (fx. |**ENTER**|) vil blive skrevet, hvis printsætningen ikke efterfølges af et semikolon ; .

Forbundne nøgleord:TAGOFF

TAGOFF

TAGOFF|#«strømudtryk»|

TAGOFF #0

KOMMANDO: Aflyser TAG for en angivet strøm, og sender teksten til den foregående tekstmarkør position, på det sted hvor TAG blev startet.

Forbundne nøgleord:TAG

TAN

TAN («numerisk udtryk»)

PRINT TAN(45)

FUNKTION: Beregner tangens til vinklen angivet i «numerisk udtryk», som skal være i området -200,000 ... + 200,000. Der regnes normalt i radianer.

Forbundne nøgleord:COS, SIN,ATN,DEG,RAD

TEST

TEST («xkoordinat»,«ykoordinat»)

PRINT TEST(300,300)

FUNKTION:Giver nummeret på den INK, som er brugt på grafikpositionen angivet.

Forbundne nøgleord:TESTR,MOVE,MOVR,PLOT,PLOTR,DRAW,DRAWR

TESTR

TESTR («xafstand»,«yafstand»)

TEST (5,5)

FUNKTION: Giver farven på den INK, der brugt på den nuværende grafikmarkørposition, efter at afstandene er lagt til.

Forbundne nøgleord: TEST,MOVE,MOVER,PLOT,PLOTR,DRAW,DRAWR

TIME

TIME

```
10 DATUM = INT(TIME/300)
20 TIKTAK = ((TIME/300)-DATUM)
30 PRINT TIKTAK;
40 GOTO 20
```

FUNKTION: Giver den tid, der er gået siden datamaten blev tændt, minus de perioder hvor datamaten læste eller skrev til kassetten. Tidsenheden er 1/300 af et sekund.

TRON
TROFF

TRON
TROFF

TRON

KOMMANDO: BASIC har indbygget en facilitet, der gør det muligt at holde øje med udførelsen af et program, ved at angive nummeret på hver linie i firkantede parenteser | | lige før den udføres. TRON starter dette, TROFF stopper det igen.

Forbundne nøgleord:RUN

UNT

UNT(«adresseudtryk»)

PRINT UNT(&FF66)

FUNKTION:Omformer et positivt 16 bits heltal i området 0..65535. Returnerer et tal i området -32768... + 32767.

Forbundne nøgleord:INT,FIX,CINT,ROUND

UPPER$

UPPER$(«strengudtryk»)

PRINT UPPER$(''amstrad'')
AMSTRAD

FUNKTION: Omformer alle små bogstaver i en streng til store bogstaver.

Forbundne nøgleord:LOWER$

VAL

VAL(«strengudtryk»)

```
10 A$ = "7 ER MIT LYKKETAL"
20 PRINT VAL(A$)
```

FUNKTION: Trækker et numerisk udtryk ud fra begyndelsen af en streng. Det modsatte af STR$.

Forbundne nøgleord:STR$

VPOS

VPOS(#«strømudtryk»)

PRINT VPOS(#0)

FUNKTION:Returnerer den lodrette position af tekstmarkøren i den angivne strøm. Strømmen skal angives.

Forbundne nøgleord:POS

WAIT

WAIT «portnummer»,«maske»|,«inversion»|

WAIT &FF34,20,25

KOMMANDO: Stopper al operation, indtil en given I/O port giver et bestemt tal i området 0..255. BASIC kører i løkke mens der ventes. Værdien som indlæses bliver XOR'eret sammen med inversionen, og derefter ANDet sammen med masken indtil et resultat forskelligt fra 0 fremkommer. BASIC kan stoppe helt i en WAIT løkke, hvis den angivne kombination ikke kommer. Hvis du prøver ovenstående eksempel, må du resette datamaten for at komme ud igen.

Forbundne nøgleord:INP,OUT

WEND

WEND

```
10 MODE 1:REM BASIC UR RUTINE
20 INPUT "INDTAST KLOKKEN TIME,MINUT, SEKUND
(T,M,S)";TIME,MIN,SEK
30 CLS:DATUM = INT (TIME/300)
40 WHILE TIME<13
50 WHILE MIN<60
60 WHILE TICK<60
70 TICK = (INT(TIME/300)-DATUM) + SEK
80 LOCATE 70,4
90 PRINT #0,USING "## ##";TIME,MIN,TICK
100 WEND
110 TICK = 0
115 SEK = 0
120 MIN = MIN + 1
130 GOTO 30
140 WEND
150 MIN = 0
160 TIME = TIME + 1
170 WEND
180 TIME = 1
190 GOTO 40
```

KOMMANDO: WHILE/WEND løkken gentager en programblok indtil en given betingelse er sand. Eksemplet her bruges til at vise elegancen af et program, der er opbygget på denne måde. Skelettet af dette program kan videreudbygges med masser af andre ting. WEND afslutter en WHILE løkke.

Forbundne nøgleord:WHILE

WHILE

WHILE «logisk udtryk»

WHILE DAG<0

se eksemplet ovenfor

KOMMANDO: En WHILE løkke gentager en programblok indtil en betingelse er sand. WHILE kommandoen angiver starten af løkken og angiver betingelsen. WEND afslutter WHILE løkken.

Forbundne nøgleord:WEND

WIDTH

WIDTH «heltalsudtryk»

WIDTH 86

KOMMANDO: Fortæller BASIC hvor bred printeren er i karakterer. Denne information tillader BASIC at indsætte vognreturkarakterer som nødvendigt, når der udskrives.

Forbundne nøgleord:PRINT,POS

WINDOW

WINDOW |#«strømudtryk»,|«venstre»,«højre»,«top», «bund»

```
10 MODE 1
20 BORDER 6
30 WINDOW 10,30,7,18
40 PAPER 2:PEN 3
50 CLS
60 PRINT CHR$(143);CHR$(242);"DETTE ER POSITION"
70 PRINT "1,1 I TEKSTVINDUET"
80 GOTO 80
```

KOMMANDO: Sætter et tekst vindue for en given skærmstrøm.

Forbundne nøgleord: ORIGIN

WINDOW SWAP

WINDOW SWAP «strømudtryk»,«strømudtryk»

WINDOW SWAP 0,2

KOMMANDO: Bytter tekstvinduerne om. F.eks sendes BASIC beskeder til strøm 0. Disse kan sendes til et andet vindue for at være af vejen, når man udvikler et program.

Forbundne nøgleord:WINDOW,PEN,PAPER,TAG

WRITE

WRITE |#«strømudtryk»,||«udskrivningsliste»|

```
WRITE #2,"HALLO",4,5
"HALLO",4,5
```

KOMMANDO: Udskriver værdien af et antal udtryk til en given strøm, idet de bliver adskilt af kommaer, og strenge sendes i citationstegn.

Forbundne nøgleord: PRINT

XPOS

XPOS

PRINT XPOS

FUNKTION: Finder grafikmarkørens vandrette position.

Forbundne nøgleord:YPOS,MOVE,MOVER,ORIGIN

YPOS

YPOS

PRINT YPOS

FUNKTION: Finder grafikmarkørens lodrette position.

Forbundne nøgleord:XPOS,MOVE,MOVER,ORIGIN

ZONE

ZONE «heltalsudtryk»

```
10 PRINT 1,2,3
20 ZONE 19
30 PRINT 4,5,6
```

KOMMANDO: Ændrer bredden af den zone, som bruges af PRINT , fra standardværdien 13 til en ny værdi i området 0..255. Stilles automatisk af NEW, LOAD,CHAIN, RUN "«filnavn» kommandoerne.

Forbundne nøgleord:PRINT, WIDTH

PRINT

PRINT (#«strømudtryk»,||«print liste»|«USING klasul»||«separator»|

«print liste» er «udtryk»|«separator»«print del»|★

«print del» er «udtryk»

eller

SPC («heltalsudtryk»)
TAB («heltalsudtryk»)

Udskrift af data til en kassettefil:

```
10 OPENOUT "DATA"
20 PRINT #9, "Hello"
30 CLOSEOUT
```

Udskrift af data til en linieskriver:

```
10 BOBSSALARY =23*PI
20 PRINT #8, USING "####.##";BOBSSALARY
30 PRINT #0, BOBSSALARY
RUN
 72256.6311
Ready
```

(I mellemtiden på printeren...)

72256.63

KOMMANDO: Skriver data på «strømudtryk» med det angivne format. Se tabellen for formatkarakterer. AMSTRAD BASIC understøtter den fulde industristandard for formatteret udskrift med PRINT sætningen.

Når der ikke angives noget format, vil BASIC skrive i frit format, hvor et komma efter det udskrevne vil sende det næste udskrevne til næste print ZONE. Et semikolon adskiller udtrykkene med et enkelt mellemrum.

SPC («heltalsudtryk») udskriver det angivne antal mellemrum, nul hvis tallet er negativt. Hvis tallet er større end bredden af skærmen/printeren, så vil det blive afkortet. SPC behøver ikke at blive afsluttet med et komma eller et semikolon, et semikolon antages til alle tider.

TAB («heltalsudtryk») udskriver mellemrum for at flytte printpositionen, hvis negativ, antages 1. TAB reduceres til at passe til bredden på en strøm som ovenfor.

PRINT (fortsat)

PRINT USING "«format felt specifikation»"

NUMERISK UDSKRIFT

Specifikation	mulige cifre	Feltkarakter	Definition	eksempel
#	1	1	Numerisk felt	####
.	0	1	Decimalpunktum	#.#
+	0	1	Print fortegn foran eller bag Positive tal får + Negative tal kan ikke få et foranstillet -	+## ###+
-	0	1	Efterfølgende fortegn	##.##-
**	2	2	Foranstillet stjerne	**###.##
$$	1	1	Flydende dollar tegn. $ bliver skrevet foran første ciffer	$$##.##
**$	2	3	Udfyldning med stjerner og flydende dollartegn	**$#.##
,	1	1	Brug komma for hver 3 cifre til højre for punktum	##,###,#
↑↑↑↑	0	4	Exponentielt format. Tal anbragt så første ciffer <>0	#,##↑↑↑

STRENG

!	Kun første karakter	!
«mellemrum»	Antal mellemrum og et foran og et bagved	
&	Variabel længde på felt	&

9 Mere om programmering

I dette kapitel behandles:

★ Placering af tekst
★ Kontrolkarakterer
★ Operativsystemet
★ Interruptstrukturer

9.1 Markør placeringer og kontrolkode udvidelser

I en del anvendelsesprogrammer kan tekstmarkøren placeres udenfor det aktuelle vindue. Forskellige operationer tvinger markøren til en tilladt position før de udføres, disse er:

-når karakterer udskrives
-når markør "blokken" tegnes
-når kontrolkoderne adlydes, dersom disse er mærket med en stjerne i listen på de følgende sider.

Fremgangsmåden for at tvinge markøren til en tilladt position er som følger:

a. Hvis markøren er til højre for højre kant, så flyttes den til den yderste venstre kolonne i næste linie.

b. Hvis markøren er til venstre for venstre kant, så flyttes den til den yderste højre side af linien ovenfor.

c. Hvis markøren er ovenfor den øverste grænse, så rulles vinduet en linie ned og markøren anbringes i øverste linie af vinduet.

d. Hvis markøren er under nederste linie, så bliver vinduet rullet en linie op, og markøren anbringes på nederste linie i vinduet.

Undersøgelserne og operationerne udføres i den angivne rækkefølge. Den illegale markørposition kan være positiv eller negativ, hvilket er til venstre for eller ovenfor vinduet.

Karakterværdier (se Appendiks III) i området 0...31 afsendt til tekstskærmen giver ikke en karakter på skærmen (og kan få datamatens operativsystem til at gå i baglås, hvis brugt uklogt), men opfattes som kontrolkoder. Nogle af disse ændrer betydningen af en eller flere af de følgende karakterer, disse er da kodens parametre.

Kontrolkarakterer sendt til grafikskærmen bliver udskrevet, (se beskrivelsen af nøgleordet TAG i kapitel 8.), og karakterer som fx (&07 'BEL' (klokke)) vil blive vist med et symbol svarende til deres funktion, hvis de blev indtastet fra tastaturet (fx |**CTRL**| **G**), men de vil kun udføre deres kontrolkodefunktion, hvis de sendes med fx PRINT CHR$(&07).

Koder mærket med ★ tvinger markøren til en tilladt position i det aktuelle vindue før de adlydes - men de kan efterlade markøren på en ikke tilladt position. Koderne og deres betydning er beskrevet med deres hexadecimale værdi først, og derefter med deres tilsvarende decimale værdi.

Yderligere kontrolkarakter kommandoer: Ikke normalt tilgængelige med CTRL-tasten

Værdi	Navn	Parameter	Betydning
&00 0	NUL		Ingen virkning. Ignoreres.
&01 1	SOH	0..255	Skriv symbolet med parameterværdien. Dette tillader udskrift af symbolerne 0..31.
&02 2	STX		Slå tekstmarkøren fra.
&03 3	ETX		Slå tekstmarkøren til. Bemærk at BASIC ikke normalt tilsidesætter markøren, undtagen når der ventes på indtastninger fra tastaturet.
&04 4	EOT	0..2	Sætter MODE for skærmen. Parametren læses modulus 4. Svarer til kommandoen MODE.
&05 5	ENQ	0..255	Send parameterkarakteren til grafikmarkøren.
&06 6	ACK		Slå tekstskærmen til. (Se &15, NAKpå næste side).
&07 7	BEL		Udsender lyd. Starter lydkøerne.
&08 8 ★	BS		Flytter markøren en karakter tilbage.
&09 9 ★	TAB		Flytter markøren en karakter frem.
&0A 10 ★	LF		Flytter markøren en linie ned.
&0B 11 ★	VT		Flytter markøren en linie op.
&0C 12	FF		Sletter tekstvinduet og flytter markøren til øverste venstre hjørne. Svarer til CLS kommando.
&0D 13 ★	CR		Flytter markøren til venstre side af vinduet på aktuel linie.
&0E 14	SO	0..15	Sætter INK for papiret. Parameter læses modulus 16. Svarer til PAPER kommando.
&0F 15	SI	0..15	Sætter farven for pennen. Parametren læses modulus 16. Svarer til PEN kommandoen.
&10 16 ★	DLE		Sletter aktuel karakter. Fylder karaktercellen med den nuværende paper INK.
&11 17 ★	DC1		Slet fra venstre side af vinduet til og med aktuel karakterposition. Fylder de berørte celler med den aktuelle paper INK.

Værdi	Navn	Parameter	Betydning
&12 18 ★	DC2		Slet fra og med den nuværende karakterposition til højre side af vinduet. Fylder de berørte celler med paper INK.
&13 19 ★	DC3		Slet fra starten til og med den nuværende markørposition. Fylder de berørte celler ved paper INK.
&14 20 ★	DC4		Slet fra og med den nuværende markørposition til slutningen af vinduet.Fylder de berørte celler med den nuværende paper INK.
&15 21	NAK		Slår tekstskærmen fra. Skærmen vil ikke reagere på noget indtil ACK (&06 6) sendes.
&16 22	SYN	0..1	Parameter modulus to. 0 slår gennemsigtig udskrift fra, 1 slår til.
&17 23	ETB	0..3	Parameter modulus 4. 0 sætter normal grafik blæk, 1 XOR 2 AND 3 OR
&18 24	CAN		Byt om på PAPER og PEN INK.
&19 25	EM	0..255 0..255 0..255 0..255 0..255 0..255 0..255 0..255 0..255	Sætter matricen for en brugerdefinerbar karakter.Svarer til SYMBOL kommandoen. Bruger ni parametre. Den første angiver hvilken karakters matrice, der skal ændres. De næste otte angiver matricen: Den største bit i første byte angiver øverste venstre hjørne af karakteren. Den mindste bit i sidste byte angiver nederste højre hjørne af karakteren.
&1A 26	SUB	1..80 1..80 1..25 1..25	Sætter et vindue. Svarer til WINDOW kommando. De første to parametre angiver venstre og højre side af vinduet. De to andre angiver top og bund af vinduet.
&1B 27	ESC		Ingen virkning. Ignoreres

Værdi	Navn	Parameter	Betydning
&1C 28	FS	0..15 0..31 0..31	Sætter INK til to farver. Svarer til en INK kommando. Den første parameter (modulus 16) angiver INK nummeret, de næste to (modulus 32) angiver farverne.
&1D 29	GS	0..31 0..31	Sætter BORDER til to farver. Svarer til BORDER kommando. De to parametre (modulus 32) angiver de to farver.
&1E 30	RS		Flyt markøren til øverste venstre hjørne af vinduet.
&1F 31	US	1..80 1..25	Flyt markøren til den angivne position i det aktuelle vindue. Svarer til LOCATE kommandoen. Første parameter angiver kolonnen der skal flyttes til, anden parameter angiver linien.

9.2 Maskinens operativsystem

Husholdningen i CPC464 styres af et avanceret realtidsoperativsystem. Operativsystemet styrer "trafikken" gennem datamaten fra indgang til udgang.

Først og fremmest forbinder det maskinens hardware med dens BASIC fortolker. Fx den blinkende INK funktion, hvor BASIC giver parametren, og operativsystemet arbejder videre med sagen, idet een del af systemet beslutter, hvad der skal gøres, og en anden del, hvornår det skal gøres.

Mens det er næsten umuligt at låse CPC464 når du bruger BASIC, andet end med ON BREAK GOSUB, er det forholdsvis nemt at lave problemer, der kun kan løses af et fuldt reset, med deraf følgende tab af alle data program etc., hvis du dykker ned i operativsystemet.

Hvis du er fristet til at POKE rundt i maskinens hukommelse, eller at kalde subrutiner (med CALL), så gem dit program før du gør det. Ellers kan det være du fortryder det! Det ekstensive operativsystem indbygget i CPC464 beskrives i guiden for viderekommne brugere, og er udenfor denne introduktions rammer.

9.3 Interrupts

CPC464 bruger Z80 interrupts meget, for at kunne give et operativsystem der indeholder flere flerbruger muligheder, fx med AFTER og EVERY kommandoerne. Prioriteringen af timerne er som følger:

BREAK |**ESC**| |**ESC**|
Timer 3
Timer 2 (og de tre lydkanal køer)
Timer 1
Timer 0
Interrupt bør medtages efter overvejelse af konsekvenserne for alle mulige mellemstadier af variabler på forstyrrelsestidspunktet. Interrupt subrutinen bør undgå uønsket roderi med tilstanden af variablerne i hovedprogrammet.

Lydkøerne har uafhængige interrupts med lige prioritet. Når et lyd interrupt startes, bliver det ikke forstyrret af nogen andre. Dette gør det muligt for lydkommandoer at dele variabler, uden at de ovenfor nævnte muligheder kan forekomme.

Når en lydkøs interrupt bliver slået til, vil det omgående starte, medmindre køen til den pågældende kanal ikke er fuld. Ellers vil det starte når næste lyd starter, og der er plads til mere i køen. I det øjeblik et interrupt affyres, kan det ikke affyres mere. Derfor må subrutinen slå det til igen, hvis der er brug for flere. Hvis du forsøger at affyre en lyd, eller du læser tilstanden af køen vil også slå enhver begivenhed fra.

Prioriteten af |**ESC**| |**ESC**| over alle andre interrupts sikrer at BASIC programmer kan stoppes uden tab af programmet - under forudsætning af at der ikke er indbygget selvdestruktion i programmet som sikkerhed.

9.3 AMSOFT Assembler

For at programmere meget i maskinkode, er det nødvendigt at bruge en assembler. AMSOFT assembleren indeholder en relokerbar Z80 assembler, med editor, disassembler og monitor.

10 Interrupt-muligheder

I dette kapitel behandles:

★ AFTER
★ EVERY
★ REMAIN
★ Hoveduret

Hvis du ikke allerede har lagt mærke til det, så er en hovedide i CPC464s programmel, at den kan at styre interrupts fra BASIC. Dette betyder at AMSTRAD BASIC er i stand til at udføre et antal samtidige, men separate operationer indenfor et program. Denne facilitet kaldes multitasking, og denne er indbygget med brugen af de nye ord AFTER og EVERY.

Denne facilitet vises også klart af måden hvorpå lyd kan styres med muligheder som fx køer og møder.

Enhver brug af timing henfører til hoveduret, som er et kvartsstyret ur indbygget i hoveddatamaten, som sørger for timing og synkronisering af begivenheder som sker i datamaten. Dette kan fx være aflæsning af tastatur, scanning af skærmen og clockpulser til CPU'en. Når en funktion i hardwaren hænger sammen med tiden, kan tiden føres tilbage til hoveduret.

Indbygget i softwaren er AFTER og EVERY, som for at blive i den brugervenlige stil i AMSTRAD BASIC, gør præcis hvad de siger: Fx AFTER (efter) den tid du har angivet i kommandoen, vil programmet gå til den subrutine der er angivet, og udføre den opgave der er her.

10.1 AFTER

CPC464 har indbygget et sandtidsur. AFTER kommandoen tillader BASIC programmer at kalde subrutiner på et bestemt tidspunkt i fremtiden. Fire timere er til rådighed til dette, og hver af disse kan have sin egen subrutine at styre.

AFTER «heltalsudtryk»|,«heltalsudtryk»| GOSUB «linienummer»

Den første heltalsudtryk angiver hvor længe der skal gå før subrutinen kaldes. Denne tid måles i 1/50 sek.

Det andet heltalsudtryk angiver hvilken af de fire timere, der skal bruges. Udtrykket skal give en værdi 0...3. Hvis udtrykket udelades, antages værdien 0.

Når den angivne tid er gået, bliver subrutinen automatisk kaldt, præcis som den ville være blevet det, hvis der havde stået en almindelig GOSUB sætning i programmet. Når subrutinen er færdig, bruges en normal RETURN kommando til at hoppe tilbage til hovedprogrammet, som så fortsætter hvor det slap.

Timerne har forskellig interrupt prioritet. Timer 3 har den højeste prioritet og timer 0 den laveste.

AFTER kommandoer kan sendes på ethvert tidspunkt, idet timeren da nulstilles. Timerne er de samme som bruges i EVERY kommandoen, så en AFTER kommando overskriver en evt. tidligere EVERY kommando for den enkelte timer og omvendt.

```
10 MODE 1:X=0
20 AFTER 45 GOSUB 100
30 AFTER 100,1 GOSUB 200
40 PRINT "AMSOFT"
50 WHILE X<100
60 LOCATE #1,30,1:PRINT #1,X:X=X+1
70 WEND
80 END
100 PRINT "Ekstraudstyr"
110 RETURN
200 PRINT "og programmer"
```

Læg mærke til brugen af to strømme (vinduer) for at muliggøre udskrift fra hovedprogrammet (linie 50-80) med uafhængig markørplacering af den, der bruges af subrutinerne.

AFTER (efter) den tid du har indstillet i kommando linien, hopper programmet til den angivne subrutine og udfører den opgave der står heri.

10.2 EVERY

Kommandoen EVERY lader et BASIC program kalde subrutiner med faste mellemrum. Fire timere er til rådighed, og hver af disse kan have en tilknyttet subrutine. Dens form er:

EVERY «heltalsudtryk»|,«heltalsudtryk»| GOSUB «linienummer»

Det første heltalsudtryk angiver hvor længe der skal ventes mellem hvert kald af subrutinen. Denne tid måles i 1/50 sek.

Det andet heltalsudtryk angiver hvilken af fire mulige timere, du vil bruge. Udtrykket skal give en værdi i området 0..3. Hvis udtrykket udelades, antages timer 0.

Når den angivne tid er gået, bliver subrutinen automatisk kaldt, præcis som hvis en GOSUB linie lige var blevet mødt på det aktuelle sted i programmet. Når subrutinen er færdig, bruges en normal RETURN kommando til at returnere til hovedprogrammet, som da forsætter hvor det slap.

Timerne har forskellig interrupt prioritet. Timer 3 har den højeste prioritet og timer 0 den laveste. I det øjeblik timeren udløber, bliver tællingen stoppet, og nedtællingen til næste kald begynder.

EVERY kommandoer kan indsættes hvorsomhelst, idet tiden og subrutinen forbundet med timeren naturligvis ændres. Timerne er de samme som bruges af AFTER kommandoen, så en EVERY kommando overskriver en tidligere AFTER kommando for en timer, og omvendt.

```
10 MODE 1:X = 0
20 P100 = 0:EVERY 10 GOSUB 100
30 P200 = 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 P100:LOCATE 1,2:PRINT" Ekstraudstyr":EI
105 IF P100 = 0 THEN P100 = 1 ELSE P100 = 0
110 RETURN
200 PEN P200:LOCATE 1,3:PRINT "og programmer"
205 IF P200 = 2 THEN P200 = 3 ELSE P200 = 2
210 RETURN
```

Bemærk brugen af DI og EI som hhv. slår timerinterrupts fra og til, mens kommandoerne mellem dem udføres. Dette gør at (den højere prioriterede) timer 1 forsinkes, så den aldrig dukker op under behandlingen af interruptet fra timer 0 (linie 100-110), og forstyrrer PEN og LOCATE indstillingerne.

10.3 REMAIN

Denne funktion giver den tilbageværende tid på en af de fire timere. Og derefter stoppes timeren. Tallet nul returneres, hvis timeren allerede er stoppet. Kommandoen bruges på formen:

REMAIN («heltalsudtryk»)

Appendiks I

Det muliges kunst

Nybegyndere i datamatik starter ofte med at etablere nogle referencepunkter, som vil hjælpe til at give en fornemmelse af en datamats muligheder og begrænsninger. Dette afsnit prøver at give en bred oversigt over hvad der sker hvorfor.

Skyd dimselimsen

Selv om den eneste grund, du havde til at købe CPC464, måske var at du ville spille de mange spil, der findes til den, så kan det stadig være at du undrer dig over nogle ting ved datamater, som går under betegnelsen "hardware".

Hardware er det du kan tage op og bære rundt med: Datamatens tastatur, monitoren, ledningerne o.s.v. Faktisk er det næsten alt som ikke lige er software - programmer, håndbøger, kassetter og den slags.

Nogle af de ting datamaten gør, laves med hardware. Fx farverne på skærmen. Det er så op til softwaren at bruge de muligheder der ligger i hardwaren.

Hardwaren styrer faktisk elektronstrålen inden i billedrøret, så det lyser op. Softwaren tilføjer orden og intelligens ved at fortælle hardwaren hvad der skal gøres hvornår. Den tilføjer timing og kontrol og rækkefølger for at give udseendet af rumskibe der letter, eller noget mere almindeligt som fx at der kommer en karakter på skærmen, når du trykker på tastaturet.

Spørgsmål: Hvad gør da en datamat bedre end andre datamater?

Hardware uden software er værdiløs. Software uden hardware er lige så værdiløs. Datamatens værdi opstår når de to dele forenes for at udføre forskellige opgaver. Der er nogle grundlæggende overvejelser, som kan bruges til at vurdere ydeevne af både hardware og software.

De generelt accepterede referencepunkter for mikrodatamater er nu:

1. Skærmens opløsning - den mindste del af et skærmbillede.

Dette er en kombination af faktorer, incl. antallet af farver tilgængeligt for programmøren, antallet af forskellige områder, der kan styres enkeltvis på skærmen, - kaldet pixels eller punkter. antallet af karakterer der kan udskrives på et enkelt område på skærmen.

Du kan se at CPC464 står sig godt i sammenligning med andre datamater i samme prisklasse.

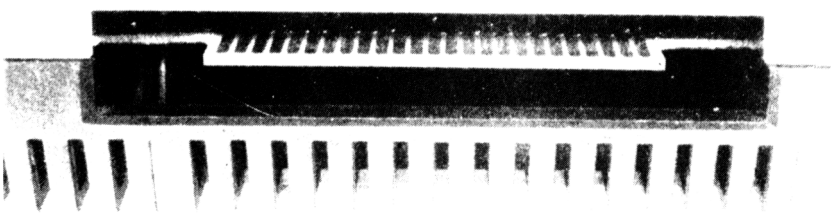
2. BASIC fortolkeren

Næsten alle hjemmedatamater har indbygget BASIC, som tillader brugeren at begynde at lave programmer som bruger hardwaren. Det sprog (BASIC) der er indbygget i din maskine er i sig selv et program, der er udviklet over en million mandeår siden BASIC blev opfundet i USA. Sproget, hvis navn er en forkortelse af Beginners All Symbolic Instruktion Code, er det mest brugte datamatsprog i hele verden, og som ethvert andet sprog har det dialekter.

Udgaven i CPC464 er en af de mest udbredte dialekter af BASIC, og mange fælles programmer kan køres direkte uden ændringer. Fx programmer skrevet til brug under CP/M operativsystemet. Det er en meget hurtig udgave af BASIC - med andre ord udføres beregningerne hurtigt. Mens du ikke behøver at bekymre dig om at en datamat bruger 0.05 sekund for at gange 3 med 5 og vise svaret og en anden vil bruge 0.075 sekund til det samme, vil du opdage at i et program hvor tusinder af beregninger skal udføres, bliver det meget afgørende at maskinen udfører de enkelte beregniger hurtigt.

Du vil ofte høre ordet "maskinkode" blive brugt. Maskinkode er det sprog datamatens processor bruger. Det tager kortere tid at udføre en opgave i maskinkode, end det tager at gøre det samme gennem BASIC. Men det tager 5 til 50 gange så meget tid at udvikle et program i maskinkode, som det vil tage at lave et program, der gør det samme i BASIC.

BASICen i din AMSTRAD datamat er blandt de hurtigste og bedst udbyggede, der kan findes på en hjemmedatamat, og den indeholder mange træk, der kan hjælpe den erfarne BASIC programmør til at kunne holde ud at arbejde i et højniveasprog, og alligevel få overraskende dynamiske visuelle og musikalske virkninger frem.



3. Udvidelsesmuligheder

De fleste datamater har indbygget mulighed for at hænge noget ekstraudstyr på: Printere, styrepinde, diskettestationer. Mærkeligt nok skal nogle af de mest succesfulde datamater have en såkaldt ekspansionsboks hængt på, for at kunne bruge noget så simpelt som en printer eller en styrepind.

Køberen tænker ikke altid frem på sine behov i fremtiden, for i så fald kan en datamat, der kan tilsluttes en printer og en styrepind direkte, vise sig at være billigere i det lange løb.

CPC464 datamaten har indbygget en Centronics kompatibel printerport, mulighed for op til to styrepinde, en stereoudgang for lyden og en ekpansionsbus, der kan bruges til at forbinde diskettestioner, ekstra ROMer, serielt RS232 interface etc.

En ROM (Read Only Memory) er et integreret kredsløb, der indeholder fast hukommelse fx programmer. Den BASIC du har fået med din datamat er leveret i ROM, og det er muligt for andre ROMer at indeholde andre sprog eller udvidelser af det eksisterende sprog. En ROM har da samme funktion som en kassette, men den giver datamaten informationerne langt hurtigere end kassetten, og er derfor langt bekvemmere at arbejde med.

En ROM kan ikke bruges til at gemme informationer i og derefter transportere dem til en anden datamat på samme måde som en kassette kan.

Udvidelse er måden at sikre at datamaten kan bruges til dine fremtidige ideer indenfor programmer og ekstraudstyr. CPC464 systemet har en meget komplet og fuldt dokumenteret udvidelseskapacitet.

4. Lyd

Lydmulighederne på en datamat bestemmer om den lyder som en flue i en flaske, eller om den kan lave acceptable lyde som et elektronisk musikinstrument.

CPC464 datamaten bruger en 3 kanals 8 oktaver bred lydgenerator, som kan lave en udmærket musikalsk kvalitet, med fuld kontrol over styrke og tone envelopes. Ydermere, deles lyden ud i et stereolydbillede, hvor en kanal giver venstre side, og en anden kanal giver højre side, og den tredie sidder i midten.

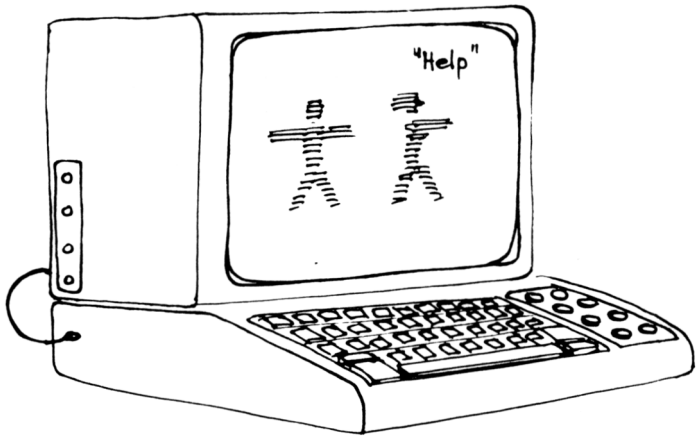
Dette giver betydelige muligheder for at skrive programmer, der styrer lydeffekterne efter en genstands placering på skærmen i et arkadespil.

I sidste ende må du selv bestemme dig til hvad du synes er det vigtigste for dig. vi håber at du vil prøve alle datamatens muligheder for at få mest muligt ud af den.

Hvorfor kan den ikke?

Med al den styrke, moderne teknologi har, undrer brugere sig jævnligt over hvorfor selv en så avanceret datamat som CPC464 ikke kan lave lige så gode billeder, som dem der ses på et fjernsyn.

Spørgsmål: Hvorfor fx kan en datamat ikke bevæge et billede af en der går hen over skærmen på en naturlig måde, hvorfor får alle datamater bevægelse til at foregå med tændstiksfigurer?



Svaret er simpelt, men indviklet. Det simple svar er at du ikke må forledes til at tro at skærmen på datamaten overhovedet kan sammenlignes med en TV skærm. Et TV arbejder med lineær information som kan beskrive en næsten uendelig række af lysstyrker mellem helt sort og helt lyst i alle spektrets farver. Dette betyder i datamatsprog at skærmhukommelsen af et fuldt TV billede kræver omkring 20 gange mere hukommelse end det der bruges på en hjemmedatamat.

Det er kun en del af problemet, eftersom dette at animere en billede kræver at denne enorme mængde hukommelse skal behandles med meget høj hastighed (omkring 50 gange pr. sekund). Det kan gøres - men kun af maskiner der for tiden koster et par tusinde gange mere end en hjemmedatamat, mindst...

Indtil prisen for højhastighedshukommelse falder drastisk (hvilket vil ske), vil små datamater blive nødt til at klare sig med en relativt lille hukommelse til at passe skærmbilledet, hvilket resulterer i lavere opløsning og ujævne bevægelser. Gennemtænkt hardware og god programmering kan hjælpe et godt stykke af vejen, men vi er stadig langt fra billige datamater, der kan reproducere flydende bevægelse og livagtige billeder, på samme måde som selv en halvgod tegnefilm kan gøre det.

Spørgsmål: Hvorfor kan man ikke bare gå hen til en datamat og skrive en side tekst ind i den?

Lad dig ikke snyde af at en datamat ser ud som en skrivemaskine med en elektronisk skærm. Skærmen er ikke et stykke elektronisk papir. Den er et kommandokonsol - hvilket er datasprog for en maskine, der kan lade dig arbejde med et programmeringssprog og de programmer der ligger i maskinens hukommelse.

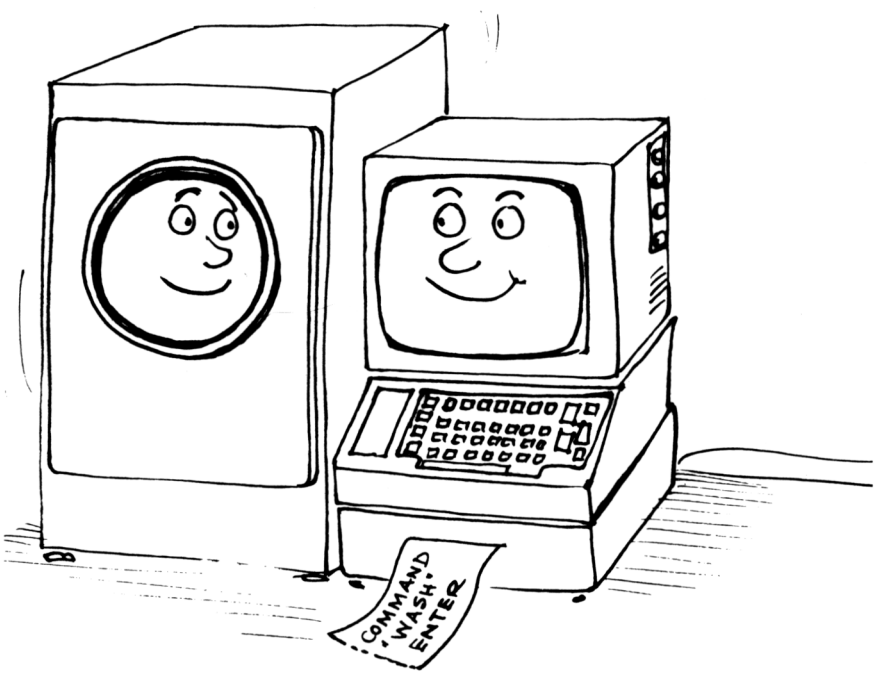
Indtil du fortæller den det modsatte, vil datamaten forsøge at forstå de karakterer du indtaster på tastaturet, som kommandoer. Når du trykker på ENTER tasten, vil datamaten gennemse det, der er blevet skrevet ind, og hvis det ikke er forståelig i BASIC, så vil datamaten skrive:

Syntax error (grammatisk fejl)

Men, det kan jo være at det program, der tilfældigvis ligger i din datamat er et tekstbehandlingsprogram, og du vil da kunne skrive tilfældige ord ind i datamaten, og trykke |**ENTER**| og arbejde videre som om datamaten var et stykke elektronisk papir i en elektronisk skrivemaskine.

Men for at kunne det, skal du først have LOADet et tekstbehandlingsprogram ind i datamaten.

En datamat synes at kombinere flere dele af udstyr som er blevet bekendte i hjemmet og på kontoret - den TV agtige skærm, tastaturet og kassettebåndoptageren - du må huske at ligheden normalt er overfladisk, og at datamaten er en kombination af kendt udseende hardware, som har helt sin egen personlighed.



Appendiks II

En introduktion til baggrunden for datamater

Hvem er bange for jargonen?

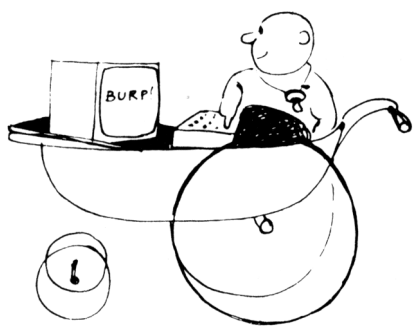
Som med alle specialistindustrier har datamatbrugere udviklet deres egen jargon, som er nogle ord som kort og præcist siger hvad der ellers ville kræve mange ord i almindeligt sprog. Det er ikke kun højteknologi som er skyldig i at gemme sig bag et tilsyneladende røgslør af svære ord, jargon og terminologi. De fleste af os kommer op mod forståelsesbarrierer, som findes i alle fag.

En stor forskel er at forvirringen i tilladt jargon kommer fra den måde ordene bliver brugt-snarere end ordene selv, som fx i når talen er om datamater. De fleste folk som sætter sig ind i datamatsprog vil prøve på at bruge ordene på den mest naturlige måde, for at minimere kommunikationens sværhedsgrad.

Lad dig ikke vildlede af at det naturlige sprog for datamatbrugere ikke er et litterært emne, men en præcis videnskab, som bortset fra ordvalgets grammatik, ikke er meget ligetil, og ikke spor forvirrende eller tvetydigt. Lærere i datalogi har endnu ikke haft held til at lave en kunstart ud af at prøve at analysere den eksakte betydning tilsigtet af programmøren i hans programopbygning.

Når vi har sagt det, uanset om betydningen af et dataprogram er åbenlys eller ej, er der stadig mange områder, der kan analyseres som enten elegante eller rodede, og der bliver lagt mere vægt på en formel indfaldsvinkel til programmering nu da den første storm omkring brugen af mikro er ved at lægge sig.

Datalogi forstås hurtigt af unge mennesker, som påskønner simpelheden og præcisionen af ideerne og den måde de kan kommunikeres- du møder ikke mange 10 år gamle sagførere- men du kan finde mange 10 år gamle programmører.



Basalt om BASIC

Næsten alle hjemmedatamater har indbygget et sprog ved navn BASIC, som tillader programmer at blive skrevet på noget der minder meget stærkt om almindeligt engelsk. BASIC bliver ikke længere, som det oprindeligt var tiltænkt, kun brugt til at lære begyndere op i datalogi, nu bruges sproget også til meget store og komplekse programmer.

Men det er klart at navnet har tiltrukket mange begyndere for dets klang af enkelhed, og lovning af en start i datamtverdenens labyrint, og dette har hjulpet et langt stykke af vejen på BASIC s udbredelse.

Fra nu af vil de almindelige ord i datasprog blive skrevet tydeligt. Lad være med at bekymre dig om at lære deres betydning ud fra det, der står her, der er en hjælp i ordbogen.

Grundlæggende begreber

BASIC er et datamatsprog, der fortolker et antal tilladte kommandoer, og derefter udfører operationer på data mens programmet kører. I modsætning til det almindelige menneskelige ordforråd på 6-8000 ord (plus alle de forskellige måder ord kan bøjes på etc.), skal BASIC klare sig med omkring 200 ord. Dataprogrammer skrevet i BASIC skal følge faste regler for brugen af disse ord. Syntaksen er præcis, og ethvert forsøg på at kommunikere med datamaten med almindeligt sprog vil føre til den kolde og kliniske besked:

Syntax error (syntaks fejl)

Dette er ikke så svært, som det lyder, eftersom sproget BASIC (syntaksen) primært er lavet til at arejde på tal - de numeriske data. Ordene er egentlig en udvidelse af de bekendte matematiske operatorer + /- etc. Det sværeste for en nybegynder at forstå, er, at datamaten kun kan arbejde med tal. Informationer, der sendes til central processoren, CPU'en (datamatens hjerne), er kun tal.

TAL, VENLIGST

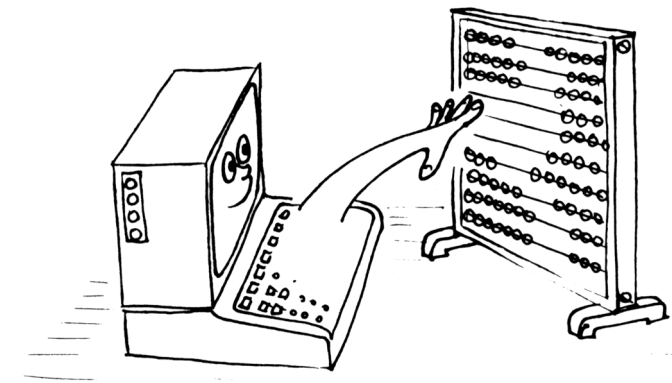
Hvis en datamat bruges til at gemme Shakespeare, er der ikke et eneste ord eller bogstav at finde i systemet. Ethvert stykke information bliver først lavet til et tal som datamaten kan finde og arbejde på som nødvendigt.

BASIC læser ordene som tal, som datamaten derefter kan arbejde på kun ved at bruge addition og subtraktion og hjælp fra Boolsk logik, der tillader datamaten at sammenligne data og udvælge efter bestemte egenskaber. Fx checke for at se om et tal er større end eller lig et andet tal, eller at udføre en defineret opgave hvis et tal eller et andet passer til bestemte kriterier.

Gennem programmet nedbryder datamaten alle opgaver til simple ja/nej operationer.

Fx processen at multiplicere udføres som gentagne additioner - BASIC instruktionen at gange 35 med 10 (35 ★ 10) løses ved at lægge 35 sammen med sig selv ti gange.

En del af CPU en (Central Processor Unit) bliver fyldt med de numeriske data 10 og en anden del af CPU en bliver fyldt med tallet 35. Hvergang 35 bliver lagt til sig selv, vil den hukommelse, der indeholder 10, blive reduceret med 1 indtil den når 0, hvor processen stopper, og det samlede resultat på 350 sendes til en anden del af CPU'en, til udskrift af svaret.



Hvis denne proces lyder besværlig, så har du helt ret, da du har indset den første og mest vigtigte sandhed om datamatik. En datamat er primært et værktøj som kan udføre gentagne simple opgaver meget hurtigt og præcist. Derfor læser BASIC programinstruktionerne og oversætter dem til en form som kan forstås af CPUen. Kun to tilstande forstås af datamatens logik -"ja" og "nej", som skrives i binær notation som "1" og "0". Boolsk logik er simpelthen sand eller falsk, der er ikke noget "måske".

Det der ligger i ordet digital er denne skiften mellem to faste værdier. I naturen flytter ting sig gradvist fra en fuldstændig stabil tilstand til en anden i jævn bevægelse. Med andre ord sker bevægelsen ved at følge en linie mellem de to tilstande. I en ideel digital omgivelse ville alle bevægelser foregå momentant, skiftet fra en tilstand til en anden ville ikke tage nogen tid. Men halvlederteknologien lader ikke dette realisere i praksis, der vil altid gå en lille tid, som kan kaldes bevægelsestid. Det er denne sammenlagte bevægelsestid for mange operationer, der giver svaret på hvorfor en datamat overhovedet bruger nogen tid på at regne før svaret er klar.

Datamaten skal alligevel vente et stykke tid fra den har udført en operation, før den starter på at arbejde på resultatet af den første opgave- så der skal indbygges nogle små pauser alligevel. Den digitale proces er sort og hvid hvor mellemstadierne i grå ikke har nogen betydning overhovedet. Den jævne bevægelse arbejder i nuancer af grå.

Hvis det endelige svar enten er nul eller 1, så er der ingen mulighed for et "næsten korrekt" resultat. Det faktum at datamater sommetider ser ud til at lave fejl når de regner på numeriske data skyldes begrænsningen på antallet af ciffre som datamaten kan arbejde på. Dette kræver at for nøjagtige data bliver presset ind i en fast form, hvilket igen fører til afrundingsfejl. fx bliver 999,999,999 til 1,000,000,000.

I en verden hvor der kun er to tal 0 og 1, hvordan tæller man så til mere end 1?

Bits og bytes

Tilfældigvis er vi blevet vant til at forstå tal som er baseret på titalssytemet, hvor referencetallet er 10- d.v.s. at der er ti cifre til rådighed til at beskrive størrelser i området 0 til 9 (som foretrækkes frem for at sige 1 til 10). Det system hvor tal ligger i området 0 til 1 er det binære talsytem, og enhederne i dette system kaldes bits - som er en forkortelse af (BInary digiT = Binært ciffer). Det kan være at det bliver lettere for dig at forstå, hvis du bruger to helt andre symboler end 0 og 1 fra det decimale system, til at repræsentere deres funktion: A og ★ kan ligesågodt bruges, så længe de bruges konsistent.

Forbindelsen mellem binær og decimal notation er nem at forstå:

Det er normalt at man afsætter en maksimalt antal cifre i et binært tal, og at man sætter nuller foran tallet, indtil alle cifre op til det maksimale er brugt.

7 i decimal bliver til:
00111
hvis man bruger 5 bit notation.
I det binære system, kan tallene anses for kolonner, som at angiver om en given potens af to er til stede eller ej.

$2^0 = 1$
$2^1 = 2 = 2 = 2(2^0)$
$2^2 = 4 = 2x2 = 2(2^1)$
$2^3 = 8 = 2x2x2 = 2(2^2)$
$2^4 = 16 = 2x2x2x2 = 2(2^3)$

På den måde ser kolonnerne således ud:

2^4	2^3	2^2	2^1	2^0	
1	0	0	1	1	
(16	0	0	2	1)	= 19

For at få en kort skrivemåde for binære tal, bruger vi udtrykket en byte, når vi omtaler et binært tal på 8 bits. Det største tal der kan gemmes i en byte er binært 11111111 eller i det decimale system 255. Dette giver 256 mulige tal, da tallet 00000000 også skal medtages.

Datamater flytter som regel rundt på mere end 8 bit af gangen. 256 er ikke noget stort tal, så for at få en effektiv måde at styre hukommelsen, bruges to bytes til at "adressere hukommelsen". Hukommelse er opbygget som en matrice, med lodrette og vandrette adresser med hvilke de enkelte elementer i den matrice kan lokaliseres:

0	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5				1		1			
6									
7									
8									
9									

Denne matrice kan indeholde op til 10★10 dele af information ad gangen, med adresser der kan ligge i området 1 til 9. Tallet, der ligger gemt på position 3,5 er et "1", det er delen på plads 5,5 også.

En binær matrice på 256★256 kan på den måde styre 65536 individuelle steder, ved at bruge 8 bit adresser til de vandrette og lodrette akser i matricen. Så vores "1" og "0" er nu istand til at styre 65536 uafhængige pladser.

Den næste forkortelse er kilobyte (kB, kByte eller "K"), som er 1024 bytes. 1024 er det nærmeste potens af 2 til det almindelige kilo for 1000, og forklarer hvorfor en datamat, der siges at have 64K hukommelse, i virkeligheden har 65536 bytes (64★1024).

Heldigvis foretager BASIC fortolkeren alle de nødvendige konverteringer for dig, og det er muligt at blive en dygtig programmør uden at forstå binære tal helt. Hvis du forstår binære tal, vil dette tilgengæld hjælpe dig til at forstå de mange "hemmelige" eller "magiske" tal, der uundgåeligt dukker op, når du arbejder med datamater.

Det er værd at bruge noget tid på at forstå binære tal og de forskellige betydende tal 255, 1024 etc., eftersom det er usandsynligt at de vil blive ændret fra at være datamaternes grundpille i den forudseelige fremtid. Sikkerheden og simpelheden, der kommer af at arbejde med kun to cifre vil fastholde sin position overfor det komplekse arbejde andre talsystemer ville medføre.

Men.....

Simpelt og elegant er det binære talsystem, men tilgengæld er det langsommeligt og nemt at lave fejl i, når man arbejder med det, da det ikke kan læses med et enkelt blik. Det binære talsystem har et par andre forbundne talsystemer, som gør det nemmere for os programmører at arbejde. Et af disse talsystemer bruges meget af mikrodatamatfolk og kaldes for Hex (en forkortelse af hexadecimal).

Dette talsystems base er 16, og disse repræsenteres i en enkelt karakter.

Decimal:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Hex:
0 1 2 3 4 5 6 7 8 9 A B C D E F

Det hexadecimale talsytem kan nedbryde de 8 bits i en byte til to blokke på hver 4 bits. da 15 er et 4 bit tal: 1111 binært. Den første blok angiver antallet af komplette blokke af 15, og det andet angiver resten. Det er her at elegancen ved det binære system begynder at vise sig.

Vi ser igen på tabellen, der introducerede binær notation

Decimal	Binær	Hexadecimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

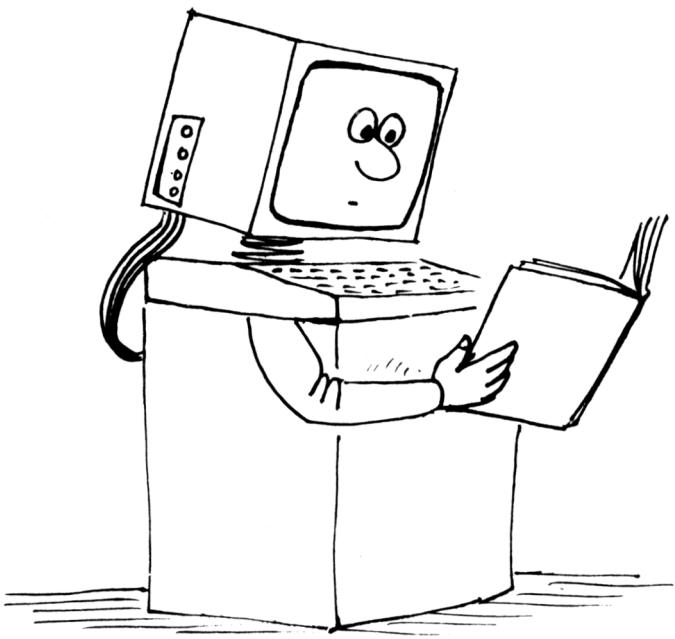
Et 8 bit tal 11010110 kan deles ind i to 4 bit tal, og disse to kan derefter oversættes til hex. I dette tilfælde fås hex D6. Når vi i denne bog bruger et hexadecimalt tal, skriver vi "&" foran fx &D6. Det hexadecimale talsystem bruges meget af programmører, der arbejder i assembler. Et assemblerprogram er det nærmeste de fleste programmører kommer til at arbejde i maskinkode, da et assembler program tillader brugeren at programmere simple bogstav koder "mnemonics" til at give hvad der i virkeligheden er binære tal. Mnemonics er navne til instruktionerne i maskinkode.

Når du bruger hex, så skal du først finde ud af værdien af det første ciffer, for at finde ud af antallet af 16nere i det færdige tal, og derefter finde værdien af det andet ciffer, således at du nu har to hexadecimale cifre. Der er en fristelse i at opfatte &D6 som 13 + 6 eller som 136. Men det er 13 ★ 16 + 6 = 214.

Det er samme fremgangsmåde du bruger, når du læser et decimalt tal. Fx tallet 89 er 8 ★ 10 + 9. Det er tilfældigvis sådan at det at gange med ti er en hel del nemmere end det at gange med 16.

Hvis du er nået så langt uden at blive for forvirret, så er du allerede godt på vej til at fatte de grundliggende principper for datamater. Det kan være du endda undrer dig over hvad alt dette postyr skal til for -og du har helt ret. En datamat er en indretning der bruger meget simple ideer, den gør det bare meget hurtigt (millioner af gange pr sekund), og med en stor kapacitet til at huske både de indlæste data, og mellemresultaterne af de mange tusinder af simple mellemresultater undervejs til resultatet.

Hvis du vil dykke ned i teorien bag din datamat, er der tusinder af bøger tilgængelige om emnet. Nogle vil efterlade dig mere forvirret, end da du startede på dem, og nogle vil faktisk lede dig på vej, ved at afsløre simpelheden og de fundamentale forbindelser der er mellem de forskellige talsystemer, og måden din datamat arbejder.



ORDBOG

Nogle almindeligt brugte ord fra datamaternes verden forklaret for CPC464 brugere.....

Akkumulator
En hukommelse i mikroprocessoren i hjertet på mikrodatamaten, som gemmer data midlertidigt, mens de bliver behandlet. Bruges meget i maskinkodeprogrammering. Brugere af BASIC behøver ikke at vide, at den eksisterer.

Akustisk kobler
Kaldes også for et akustisk modem. En elektronisk forbindelse til en datamat, som forbinder et telefonrør med datamaten og tillader datamaten at kommunikere over telefonnettet. På denne måde kan en datamat tale med de offentlige databaser og andre brugere af hjemmedatamater.

Adresse
Et tal i en instruktion, som betegner en bestemt celle i en datamats hukommelse. Når man kender en bestemt adresse, kan man læse og skrive i den celle, adressen tilhører, hvis addressen ellers er i RAM.

Adventure
En kult for nogle og en kedsommelig oplevelse for andre. Et tekstbaseret datamatspil i hvilket brugeren bydes at deltage i et antal begivenheder som baseres på at finde vej i en labyrint.

Algoritme
Et flot navn for en formel eller opskrift på, hvordan man løser et programmeringsteknisk problem.

Alfanumerisk
Betegnelse for en karakter, som enten er et tal eller et bogstav, og ikke en grafikkarakter.

ALU
Aritmetic Logic Unit. Den del af en mikroprocessor, som udfører beregningerne. Har kun interesse for maskinkode programmører.

AMSOFT
AMSTRADs datamatstøtt-organisation, som leverer software, periferiudstyr og bøger for at støtte de mange brugere af CPC464.

A/D
Analog
En tilstand, hvor skift mellem start og slutværdi sker gradvis i stedet for i endelige spring. Datamater er digitale maskiner - det meste af den virkelige verden er analog. Derfor må en datamat lave en analog til digital (A/D) omformning, før den kan bearbejde data fra den virkelige verden.

Animation
Tegnefilm er den bedst kendte form for animation - datamat-animation baseres på bevægelig grafik for at simulere rigtige bevægelser.

Applikationsprogram
Et program, som er skrevet for at udføre en bestemt opgave snarere end et generelt softwareudviklingsværktøj som fx en assembler.

Arkadespil
Den type bevægelig grafikspil, hvor marsmænd invaderer og bliver slået tilbage, vanvittige monstre løber rundt i en labyrint og spiser de uforsigtige. Figurer, der under styring af brugeren, skal undgå alle mulige ubehagelige dødsfælder. Morsommme og gode for reflekserne, men til ringe nytte for undervisning af datamatbrugere.

Arkitektur
Diagrammet over databussen, yderenhedernes forbindelser og CPU'ens styring i en mikrodatamat. Ikke noget, der vil betyde noget for læsere af denne ordbog indtil videre!

Argument
En uafhængig variabel. Fx i udtrykket X+Y=Z er X og Y argumenterne.

Array
Indiceret variabel
En 2-dimensionel matrice (et gitter), hvor data gemmes og hentes ved hjælp af de lodrette og vandrette koordinater.

Artificial Intelligence
Simuleret intelligens
En struktur i programmeringstekninkken, som tillader programmet at lære af sine fejl.

ASCII
American Standard Code for Information Interchange. En almindeligt brugt standard til at give numre til en datamats bogstaver og tegn. Koderne for en CPC464 findes i Appendiks III.

Assembler
Den praktiske måde at oversætte maskinkodeprogrammer, hvor maskinkodeinstruktionerne kendes med nogle bogstavkoder (mnemonics) i stedet for med de talkoder, de i virkeligheden er.

Bar code
Stregkode
Det, du ser på siden af flere og flere indpakninger i et supermarked. Kan læses af en datamat med brug af laser eller andre optiske instrumenter.

Basis
En grundlæggende overvejelse for al matematik. Talbasen er grundlaget for talsystemerne. Binære tal har basen 2, mens vi normalt bruger base 10 i det decimale system. Hexadecimale tal ligger i base 16.

BASIC
Beginners All-purpose Symbolic Instruction Code. Et fortolket programmeringssprog , som bruges i næsten alle hjemmedatamater. BASIC er skrevet for at være nemt at lære og bruge, da det

tillader, at et program bliver testet og kørt, før det er færdigt. Dette i modsætning til oversatte programmer, som skal oversættes, før de kan afprøves.

Baud
Bit pr sekund. Enheden, hvormed man måler overføringshastigheder imellem datamater og fx printere og diskettestationer i.

BCD
Binary Coded Decimal. Et kodesystem til kodning af decimaltal, hvor hvert decimalt ciffer repræsenteres af 4 binære bit.

Binær
(Se Base). Talsystemet med talbasis 2. Alle binære tal er lavet af cifrene 0 og 1.

Binært tal
Et tal, som vises i binær notation. I CPC464 står der et &X tegn foran et binært tal. Fx &X0101 som er lig med 5 i decimaltal.

Bit
Forkortelse af BInary digiT. Binært ciffer enten 0 eller 1.

Bitsignifikant
Hvor informationen indeholdt i et tal ses som tilstanden af hver bit i tallet. Decimalværdien kan være helt uden mening.

Booting eller
Boostrapping
Programmer og operativsystemer loader ikke sig selv. De bliver hevet ind af en Bootstrap rutine i ROM. Denne starter loadning og videre kørsel af operativsystemet og dets programmer.

Boolsk algebra
Bruges til logiske sammenligninger, hvor der kun kan være et af to svar: Sandt eller falsk.

Buffer
Et midlertidigt lager for data under transport fra fx diskette til hukommelse. En buffer virker som station til opsamling af data fra enheder, der sender med forskellig hastighed.

Bug
Et programmeringsproblem, som ligger på skalaen fra "nej, kan den også det!" (Hvis man trykker fire taster ned samtidigt, skifter skærmen farve) til rene katastrofer, hvor datamaten ikke kommer videre og evt. sletter al hukommelse.

Bus
En gruppe af forbindelser enten indeni datamaten eller mellem den og omverdenen, som bærer information mellem CPUen og de andre hardwareenheder. Du kan se CPC464's bus på bagsiden af datamaten, hvor den er det største af de to stik.

Byte
En gruppe på 8 bit, som udgør den mindste enhed, en 8 bit CPU kan gemme eller hente i hukommelsen. Se yderligere information i Appendiks II.

CAD
Datamatstyret design. Normalt et samarbejde mellem datamatens beregningskraft og dens grafiske evner, som tilsammen udgør et elektronisk tegnebræt.

CAE
Datamatstøttet undervisning. Der findes underafdelingerne CAI datamatstøttet instruktion og CAL datamatstøttet indlæring.

Cartridge
En specialpakket integreret enhed, som indeholder software, der kan stikkes direkte ind i datamaten. Software, der leveres på denne måde, loader hurtigere, men koster også langt mere end almindelig software på bånd.

Cassette
Ud over et bånd til en båndoptager kan dette betyde kassen omkring en ROMpakke ellr lign.

Character
Karakter
Et symbol, der kan vises af en datamat fx bogstaver tal, grafiske symboler o.s.v.

Character cell
Matricen af punkter, der vises på skærmen, som ser ud som den ønskede karakter.

Character set
Karaktersæt
Alle de bogstaver og tal og symboler, der findes indbygget i datamaten.

Character string
Karakterstreng
Et stykke variabel, som indeholder en række af karakterer, som kan gemmes og hentes under et, fx et ord.

Chip
Et populært udtryk for et integreret kredsløb.

Clock
Det indbyggede meget hurtige ur, som findes i alle datamater, og som styrer, hvornår de enkelte operationer skal udføres. Et sandtidsur er et der tæller i timer, minutter og sekunder.

Code
"Kode"
Bruges som kælenavn for maskinkode af programmører.

Command
Kommando
En programinstruktion

Compiler
Et kompliceret program, der omformer hele programmer fra et højniveausprog til maskinkode, hvorved programmet kan køre meget hurtigere.

Computer generations
Datamatgenerationer
Teknologiske landvindinger har givet bestemte spring i udviklingen af datamater. Hvert af disse spring kaldes er generation.

CP/M
Standarden for 8-bit mikroprocessor-operativsystemer.

CPU
Central Processing Unit. Den enhed i datamaten, som udfører tænkerarbejdet. I en mikrodatamat er CPUen en mikroprocessor.

Cursor
Markør
En bevægelig markør, som viser, hvor næste karakter kommer på skærmen.

Cursor control keys
Markørstyretaster
Taster, der flytter markøren rundt på skærmen. Disse bruges fx tit i arkadespil til at styre bevægelsen af spillet.

Daisy-Wheel printer
Typehjulsskriver
En printer, der kan skrive skønskrift som en skrivemaskine. Bogstaverne skrives i eet stykke af printeren.

Database
Et system til at gemme informationer i.

Data Capture
Datafangst
Ord, der beskriver, hvordan datamaten henter data fra enheder uden for sig selv.

Debugging
Den proces, det er at finde fejl og rette disse i en maskine.

Decimal notation
Bruges til at skrive tal med talbase 10 ud, med cifrene 0-9, hver repræsenterende en tierpotens, tiere, hundreder, tusinder o.s.v.

Diagnose
En besked lavet af en datamat for at beskrive en fejl i et program.

Digital
Beskriver en størrelse i endelige spring, i stedet for i en kontinuerlig kurve. Det modsatte af analog.

Digitiser
En måde at plotte analog information ind i en datamat. Bliver også kaldt grafiske borde.

Disk
Diskette
En flad, tynd cirkulær plade af plastic betrukket på en eller begge sider med en magnetisk belægning, som bruges til opbevaring af data. Disken ligger i en flad kvadratisk "konvolut" eller

plastkassette med plads til læsehovedet i et vindue. Se også Floppydiske og Winchesterdiske.

Diskdrive
Diskettedrev
Apparatet, der læser og skriver information magnetisk på en diskette.

Dokumentation
Manualer og bøger, der følger med en datamat for at forklare, hvordan den bruges.

Download
Overførslen af information fra en datamat til en anden. Den datamat, der modtager data, kaldes den maskine, der "downloader". Den anden ende af forbindelsen er den datamat, der "uploader".

DOS
Diskette Operativ System. Den software, der styrer alle operationer på disketten i et diskettedrev. Virker som kontrolløren på en bilparkeringsplads, som holder øje med ledige pladser på disketten.

Dot matrix
Et rektangulært gitter af punkter, på hvilke en karakter kan vises ved at vælge bestemte punkter.

Dumb Terminal
Uintelligent terminal
En dataterminal, der ikke udfører nogen beregning på data. Kan kun sende og modtage data.

Editering
At rette eller ændre i et program.

Editor
Et program, der normalt ligger i ROM på en datamat, som lader dig editere (redigere) et program.

EPROM
Erasable Programable Read Only Memory. Svarer til PROM, men kan slettes ved hjælp af ultraviolet lys. En EEPROM er næsten det samme, men denne kan slettes med elektrisk strøm.

Expression
Udtryk
En simpel eller kompleks formel, der bruges i et program til at udføre en beregning på data. Udtrykket begrænser, hvilke slags data, det kan håndtere.

Femte generations datamater
Hovedsagelig store datamater (mainframes), som loves at komme med indbygget mulighed for at prgrammere sig selv under brug af simuleret intelligens.

Fil
En samling information, generelt gemt på kassette eller disk, selv om RAM filer findes på nogle datamater.

Firmware
Software i ROM, som leveres med datamaten. En krydsning mellem ren software og ren hardware.

Fixed point number
Fast komma tal
Et tal repræsenteret, manipuleret og gemt med decimalpunktummet på et fast sted.

Floating point number
Flydende komma tal
Et reelt tal, manipuleret og gemt med decimalpunktummet flydende, så det kan være på den ønskede position blandt cifrene. Denne måde at gemme tal er særlig nyttig, når man arbejder med meget store tal.

Floppydisk
En udskiftelig 5.25 eller 8 tommer magnetisk diskette, som bruges til at gemme data. Gemt i en beskyttende kvadratisk kuvert. Meget større lagerkapacitet end en kassette, meget hurtigere og meget dyrere end en kassette.

Flowchart
Rutediagram
Et diagramsprog til optegning af et program, således at programmets enkelte trin kan ses og programmets forløb kan gennemskues.

Forth
Et hurtigt programmeringssprog, med hastighed og sværhed mellem højniveausprog og maskinkode.

Funktionstast
En tast på tastaturet, som er blevet tildelt en bestemt opgave. Denne opgave kan være tilføjet udover tastens normale brug, eller funktionen kan erstatte tastens normale brug. CPC464 har et antal brugerdefinerbare taster. Hver af disse kan med et enkelt tryk sende 32 karakterer til datamaten i fom af en almindeligt brugt instruktion til styring af fx printer, datamat eller modem.

Gate
Logiske gates tillader passageen af data, når hvisse betingelser opfyldes. Der er mange forskellige typer (OR,AND,XOR etc). Se under boolsk algebra.

Grafics mode
Tidlige mikrodatamater skulle indstilles til enten at håndtere grafik eller karakterer. Moderne personlige datamater er i stand til at blande tekst og grafik sammen.

Grafik
Den del af skærmen på en datamat, som ikke bruges til at skrive tekst på. D.v.s. linier, cirkler, figurer og kurver etc. Med en egnet printer kan man også lave en papirkopi af ens grafik.

Grafisk karakter
En form eller et mønster, som er designet specielt for at kunne bruges til at tegne mønstre. CPC464 har komplet sæt af disse beskrevet i appendiks III.

Grafikmarkør
Svarer til tekstmarkøren, men tegner på grafikskærmen. Er usynlig på CPC464 - men ikke desto mindre uundværlig til placering af tegnet grafik. Må ikke sammenblandes med grafiske karakterer (Appendiks II), som er en del af karaktersættet, og som skrives ud ved tekstmarkøren.

Grafisk tegnebord
En enhed, som plotter koordinaterne til et givet billede til viderebehandling i en datamat. En form for A/D.

Handshaking
En rækkefølge af elektroniske signaler, som initierer, checker og synkroniserer udvekslingen af data mellem en datamat og en periferienhed, eller mellem to datamater.

Hardcopy
Udskrift på papir af et program eller anden tekst - eller af et grafisk skærmbillede. Skærmbilledet er en "softcopy".

Hardware
Den elektroniske og mekaniske del af en datamat - alt, hvad der ikke er software eller firmware.

Hexadecimal notation
Tal i talbase 16. Se appendisk II. I CPC464 bruges et & eller et &H tegn foran et hexadecimalt tal.

Højniveausprog
Sprog, som er skrevet næsten som almindelige menneskesprog. Selve sproget foretager det meste af fortolkningen. Langsommere end maskinkodeprogrammer, men meget enklere at forstå. BASIC er et højniveausprog.

IEEE-488
En af standarderne for interfaces (snitflader) mellem datamater og ydre enheder. Ligner, men er ikke helt magen til Centronics standarden.

Informationsteknologi
Alt, hvad der har med brugen af elektronik i behandlingen af information og kommunikation. Tekstbehandling, datakommunikation o.s.v.

Initiere
Tænde for et system, eller erklære bestemte værdier før opstarten af en bestemt procedure eller programdel.

Input
Inddata
Alle de ting, der kommer ind i en datamat fra tastaturet, kassetteenheden, disketteenheden, serielt interface eller andre kilder.

Instruktion
En kommando til en datamat om at udføre en bestemt opgave. En samling eller rækkefølge af instruktioner udgør et program.

Instruktionssæt
De grundlæggende logiske og matematiske processer, en mikroprocessor kan udføre. Enhver højniveauinstruktion (herunder mnemonics til en assembler) skal kunne destilleres ned til en række instruktioner i maskinkode, som CPU'en kan forstå. En enkelt højniveauinstruktion kan betyde kald til mange maskinkodeinstruktioner.

Integer
Heltal
Et tal uden decimaldel, d.v.s. et tal uden noget til højre for kommaet. I modsætning til reelle tal, som er heltal + decimaldel.

Integreret kredsløb
En samling af elektroniske komponenter formindsket og indbygget i et enkelt stykke silicium. Se også "Chip".

Intelligent terminal
En terminal, som ud over at kunne håndtere en datamats input og output også giver mulighed for behandling af data, når terminalen ikke bruges som terminal.

Interaktiv
Som regel en betegnelse for et program, der spørger brugeren om inddata, lige fra at styre et rumskib i et arkadespil til at svare på spørgsmål i et undervisningsprogram. Det, brugeren gør, har en sandtidsindflydelse på det, programmet gør.

Interface
Grænseflade
Vejen ind og ud gennem en datamat, både i elektriske og menneskelige begreber. CPC464 har et interface i form af tastaturet (ind) og skærmen (ud) - så vel som faciliteterne bagpå til forbindelse af brugerudstyr.

Interpreter
Fortolker
En yderligere udvidelse af analogien mellem datamatens instruktionssæt og menneskesprog. Elementet i systemsoftwaren, der fortolker højniveausproget til et niveau, hvor det kan udføres af datamatens CPU. Fx forvandles BASIC programmer til datamatens eget sprog.

I/O
Input/Output. Ind og ud.

Iteration
Et af elementerne i beregning. Datamaten udfører alle opgaver ved at nedbryde dem til små bidder, der kan håndteres af CPU'en. For at gøre dette, skal datamaten udføre hvert af disse simple elementer mange gange, indtil en given betingelse er opfyldt.

Joystick
Styrepind
En inputenhed, der generelt er hurtigere at bruge end tasterne på tastaturet og gør det nemmere at spille spil hurtigt.

K
En forkortelse af den metriske enhed Kilo, som betyder 1000. I datamater bruger man udtrykket 1K om en Kilobyte, som er lig 1024 bytes. Dette er forresten 2 opløftet til tiende postens.

Keyboard
Tastatur
Matricen af alfanumeriske kontakter arrangeret, så de kan bruges til indtastning af kommandoer og anden information til datamaten.

Keyword
Nøgleord
Et ord, som bruges i datamatens programsprog til en bestemt funktion.

Least significant bit
Mindst betydende bit
I et binært tal (se appendiks II) er den mindst betydende bit (LSB) den yderste bit til højre i udtrykket.

Linienummer
BASIC og visse andre programmeringssprog bruger at arrangere et program i orden efter linienummer.

LISP
En forkortelse af LISt Processor language. Endnu et højniveau datamatsprog.

Logik
Den elektronik i en datamat, som udfører de elementere beslutninger og funktioner, som en hvilken som helst datamats program i virkeligheden er bygget op af.

Logo
En nemt højniveaugrafiksprog, som bruges i skoler til at lære børn at betjene datamater.

Loop
Løkke
En proces i et program, der udføres hurtigt gentagne gange af datamaten, indtil en bestemt betingelse er opfyldt.

Low level language
Er fx maskinkode. Et programsprog, hvori hver eneste instruktion svarer til datamatens maskinkodeinstruktioner.

LSI
Large scale Integration. Udviklingen af integrede kredse, som fylder flere funktioner ned på endnu mindre stykker silicium.

Lyspen
Endnu en alternativ måde at indlæse data i datamaten.

Machine Code
Maskinkode
Det programmeringssprog, som forstås direkte af en mikroprocessor, eftersom alle dens kommandoer repræsenteres af mønstre af binære cifre.

Machine readable
Et datamedium eller enhver anden information, som kan sendes omgående til en datamat uden yderligere konvertering af tastkoder o.s.v.

Mand - maskine interface
Samarbejdet mellem brugeren og datamaten.

Matrice
Arrangementet af punkter, som udgør karaktercellerne på skærmen eller på printhovedet på en matrixprinter. Også et udtryk i matematik og videnskab om et array eller en indiceret variabel.

Memory
Hukommelse
Datamatens parkeringsplads for information. Arrangeret i pæne lange rækker med adgang til hvert element. Kan være RAM (random access memory), hvor informationen både kan gemmes og hentes, eller ROM, hvor informationen kun kan hentes. Disketter og bånd er eksempler på store "hukommelsesklumper", men udtrykket bruges helst om hukommelse, som CPUen har direkte adgang til.

Memorymap
Et kort over, hvordan datamatens hukommelse er lagt ud, visende forskellige addresser og placeringen af hukommelsen til fx skærmen, styringen af båndet o.s.v.

Menu
En oversigt over, hvilke muligheder man har på et givet sted i programmet.

Mikroprocessor
Et integreret kredsløb, der sidder i hjertet på datamaten og udfører de instruktioner, der presenteres for den af BASIC fortolkeren.

Modem
MOdulator DEModulator, som forbinder datamatens I/O til en telefonlinie eller andre serielle datatransmissions medier fx lysledere. SE også Akustisk kobler

Monitor
Skærmen på et datamatsystem og også en betegnelse for et program, der lader brugeren se datamatens hukommelse og styre datamatens CPU.

MSB
Den mest betydende bit i et udtryk. Bit'en til yderst til venstre i et binært tal.

MUS
En lille enhed, der kan køre rundt på et bord med hånden. En mus bruges til at flytte en markør rundt på skærmen. Oprindelig designet for at overkomme folks skræk for tastaturer og gøre programmerne mere brugervenlige.

Netværk
Når to eller flere datamater er forbundet med hinanden, så de kan bytte information. Enten med ledninger eller via modems.

Nibble
En halv byte: et binært tal med fire bits. Hvert af cifrene i tallet &F6 er en nibble.

Numerisk tastatur
Det område på tastaturet, hvor tal kan indtastes, og som i CPC464s tilfælde også giver mulighed for brugerdefinerbare taster.

OCR
Optical Character Recognition. En måde at læse trykte eller skrevne data med en optisk læser og oversætte dem direkte til data, der kan læses af datamaten.

Octal
Et talsystem med grundtallet 8. Hvert ciffer (0-7) udgøres af tre bit.

Offline
En ydre enhed - en terminal eller printer - som ikke er aktivt forbundet til datamatan siges at være offline.

Online
Det modsatte af offline.

Operativsystem
Kontrolløren på parkeringspladsen, som tidligere er omtalt. Software, der bestemmer over rækkefølge og timing i datamaten.

Output
Alt, hvad der kommer ud af en datamat som et resultat af en eller anden beregningsfunktion.

Operator
Den del af et udtryk, som får en del til at operere på en anden. D.v.s. +-*/ etc.

Overwrite
Overskrive
Slette en del af hukommelsen ved at skrive nye data ind på pladserne.

Paddle
Et andet navn for en joystick. Bliver også kaldt for et "games paddle".

Paperware
Endnu et navn for den på papir udskrevne del af en datamats arbejde. Nogle gange beskrives en datamat, som ikke eksisterer ved dens lancering, som paperware.

Parallelt interface
CPC464 printerinterfacet kan styre en parralleprinter. Dette betyder, at hver datalinie fra datamaten føres over til en tilsvarende linie på printeren. Data kan overføres meget hurtigt med et parallelt interface.

Pascal
Et højniveau struktureret programmeringssprog, som skal oversættes, før det kan køres. Derfor arbejder det meget hurtigt, når programmet er færdigudviklet. Dette er det sprog, man som regel vil kaste sig over, når man er træt af at programmere i BASIC.

PEEK
Den BASIC-funktion, der kigger direkte ind i en datamats hukommelse og giver værdien af indholdet i en bestemt celle (byte).

Periferienhed
Printere, plottere, joysticks, diskettestationer og alt andet, der stikkes direkte ind i datamaten for at udvide dens evner.

Pixel
Billedpunkt
Det mindste område på skærmen som kan styres af hardwaren.

Plotter
En bestemt type printer, der kan tegne med penne i stedet for med punkter. Bruges til at tegne tegninger og grafik med.

POKE
Den sætning i BASIC, der bruges til at placere en værdi i en bestemt celle i hukommelsen. Se kapitel 8.

Port
Et bestemt adresserbart punkt på datamatens interface til input og output.

Portabilitet
Betyder, at programmel, skrevet på en datamat, kan flyttes over på en anden datamat uden ændringer. Dette kræver som regel et fælles operativsystem som fx CP/M.

Printer
Enhver maskine til at udskrive tekst på papir.

Program
En samling instruktioner til datamaten, som når de udføres, får datamaten til at udføre en bestemt funktion.

Programmeringssprog
Det sprog, som programmer skrives i. Dette består af stive regler for, hvordan hver kommando skal skrives for at virke.

PROM
Programable Read Only Memory. Et integreret hukommelseskredsløb, der ikke kan slettes, når det først er fyldt med data.

Prompt
Et symbol eller en besked, som datamaten skriver ud for at bede brugeren fortælle datamaten, hvad den nu skal gøre. CPC464 BASIC bruger ordet Ready, når programmet er færdigt, og et simpelt spørgsmålstegn ? hvis programmet venter på inddata.

PSU
Power Supply Unit. Strømforsyningen til datamaten.

QWERTY tastatur
Bruges for at beskrive et tastatur, hvor tasterne ligger som almindelige skrivemaskinetaster.

RAM
Random Access Memory. Hukommelse, der både kan skrives og læses i. Her gemmer datamaten bl.a. dit BASIC program.

Random number
Tilfældigt tal
Et tal, der laves af datamaten, som ikke kan forudsiges eller gentages.

Raster
Et system til skrivning på skærmen, hvor mønstrene er opbygget af et antal vandrette scanne-linier.

Realtid
Begivenheder, der sker foran dine øjne, i modsætning til dem, der kun bliver synlige, når den proces, der har skabt dem, er afsluttet.

Reelt tal
Et tal, der har både en heltalsdel og en decimaldel. D.v.s., der står noget på begge sider af kommaet. Man kan godt opfatte en variabel som reel, selv om den kun bruges til heltal i et program.

Recursion
En serie af gentagne trin i et program eller i en rutine, i hvilken resultatet af hver ny omgang afhænger af den forgående.

Refresh
At opdatere information enten på skærmen eller i hukommelsen. Behøver ikke at være en destruktiv proces, men kan fx bare opfriske det, der allerede var på skærmen.

Register
En hukommelse i CPU'en, der bruges til at gemme data midlertidigt i.

Remark
Bemærkning
En ikke udførbar kommentar indbygget i et program, som indbygges for at huske på en given programdels funktion.

Reserveret ord
Et ord, der har en bestemt betydning for datamatens program, og som derfor ikke kan bruges i andre end dets foruddefinerede betydning. Fx vil BASIC ikke acceptere ordet NEW som en variabel. Ordet er allerede reserveret til andre formål.

Resolution
Opløsning
Muligheden for at se, hvor et element på skærmen slutter, og hvor et nyt starter. Bruges også løst om en datamats nøjagtighed ved beregninger på store tal.

Reverse Polish notation
Omvendt polsk notation
(RPN). En metode til at skrive regnestykker op, så de kan udregnes udelukkende fra venstre mod højre. Bruges bl.a. på visse lommeregnere.

RF modulator
HF modulator
Er den enhed, som datamaten sender information gennem, når dens billeder skal vises på en TV-skærm.

ROM
Read Only Memory. Bruges normalt om halvlederhukommelser. Når der er skrevet i en ROM, kan den hverken slettes eller rettes.

RS232C
En specifik standard til seriel kommunikation. Enhederne i begge ender af forbindelsen skal konfigureres til at tale i samme format, svarende til den del af standarden der bruges. Sammenlign dette med Centronics standarden, hvor der ikke skal indstilles noget.

Rutine
En del af et program, der udfører en bestemt fast opgave. Et underprogram, der enten er en del af hovedprogrammet, eller som eksisterer som et selvstændigt modul til sammenbygning med forskellige programmer. Et program, der laver et tolvtimers ur til at vise på skærmen, kan opfattes som en rutine.

Sandhedstabel
Resultaterne af en logisk operation er enten "sand" eller "falsk". Datamaten opfatter dette som enten 1 eller 0, og sandhedstabellen angiver de mulige resultater af en logisk operation fx (IF A=B THEN C).

Screen Editor
Skærmeditor
En tekst eller program editor, hvor markøren kan flyttes til enhver del af skærmen for at ændre de karakterer, der står der.

Scrolling
Rulning
Udtrykket beskriver den måde skærmbilledet ruller op, når skærmen bliver fyldt forneden, og der bliver brug for at have plads til det næste, der skal skrives.

Separator
En separator adskiller et reserveret ord fra de data, der følger efter, eller fra det næste reserverede ord.

Serielt interface
Selv om dette næsten altid betyder RS232C interface, kan det henvise til andre serielle standarder, der findes for overførsel af data.

Simulering
En teknik, som ofte bruges for at efterligne rigtige livsprocesser med en datamat. Det kan være flysimulering, kørselssimulering etc.

Soft key
Se User defined key (brugerdefineret tast).

Software
Programmer

Software engineering
Betyder datamatprogrammering på en strukturetret og velovervejet måde.

Sound generator
Lydgenerator
Den del af en datamat, der laver lyde. (Kan være hardware eller software).

Spreadsheet
Et program, der tillader brugeren at opstille rækker og søjler af tal og bearbejde dem. Hvis man ændrer et tal, betyder det, at alle andre tal ændres, således at man kan se indflydelsen af ændringen på de øvrige. Bruges til at beregne store sammenhænge som fx budgetter.

Sprite
En karakter på skærmen, som kan bevæges frit rundt. Kan styres af hardware eller software til at rende rundt og dukke op helt tilfældigt.

Stak
Et område i hukommelsen, der bruges til at stable information, men hvor kun øverste element på stakken nemt kan genkaldes.

Statement
Sætning
En eller flere instruktioner i et datamatprogram.

Streng
En type af data som består af nogle karakterer som ikke kan opfattes som en numerisk variabel. Karaktererne kan være numeriske, men bliver ikke behandlet som sådan, medmindre de med vilje omformes til dette brug.

Struktureret programmering
En logisk og forud gennemtænkt måde at programmere på, som resulterer i programmer der forløber fra toppen til bunden på en klar og overskueig måde i klare trin.

Strøm
Den rute som bruges til at sende data til eller fra datamaten d.v.s. skærmen, printeren o.s.v.

Subrutine
Se rutine.

Syntaksfejl
Når reglerne fra prgrammet bliver overtrådt af ukorrekt brug af nøgleord og variabler, vil BASIC give en sådan melding.

Talegenkendelse
Omformning fra talte ord til tal som datamater kan forstå.

Talesyntese
Dannelse af syntetisk tale v.h.a. en kombination af hardware og software.

Terminal
En enhed der tillader indlæsning fra tastaturet, med enten en skærm eller en printer som udskriftsenhed.

Trunkeret
Et tal eller en streng bliver kortet af ved, at der bliver fjernet indledende eller efterfølgende karakterer. Hvor dette er ønskeligt, bliver værdien eventuelt rundet af. Hvor dette ikke er ønskeligt, bliver de ekstra karakterer simpelthen slettet for at tillade tallet eller strengen at være i den afsatte plads.

User Defined Keys
UDK
CPC464 har op til 32 taster som kan defineres af brugeren til at udføre bestemte opgaver, incl. at skrive strenge på op til 32 karakterer.

Unsigned number
Tal uden fortegn
Et tal, som ikke har noget fortegn til at fortælle, om det er positivt eller negativt.

Utility
Ethvert komplet program der bruges til at udføre en almindelig operation, såsom at sortere data eller at kopiere filer.

Variabel
En enhed indeholdt i datamaten til at gemme informationer under et navn, og hvis værdi kan variere under programmets forløb.

Appendiks III

ASCII karakterne og grafikkarakterne på CPC464

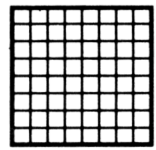
III.1 ASCII

For at give udførlig vejledning, gengiver vi her standard ASCII referencekaraktersættet, idet vi bruger decimal, oktal og hexadecimal notation, sammen med ASCII koden, hvor denne eksisterer. Hver af CPC464 karakterne er også givet i detaljer.

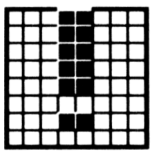
DEC	OCTAL	HEX	ASCII characters
0	000	00	NUL ([CTRL]A)
1	001	01	SOH ([CTRL]B)
2	002	02	STX ([CTRL]C)
3	003	03	ETX ([CTRL]C)
4	004	04	EOT ([CTRL]D)
5	005	05	ENQ ([CTRL]E)
6	006	06	ACK ([CTRL]F)
7	007	07	BEL ([CTRL]G)
8	010	08	BS ([CTRL]H)
9	011	09	HT ([CTRL]I)
10	012	OA	LF ([CTRL]J)
11	013	OB	VT ([CTRL]K)
12	014	OC	FF ([CTRL]L)
13	015	OD	CR ([CTRL]M)
14	016	OE	SO ([CTRL]N)
15	017	OF	SI ([GTRL]O)
16	020	10	DLE ([CTRL]P)
17	021	11	DC1 ([CTRL]Q)
18	022	12	DC2 ([CTRL]R)
19	023	13	DC3 ([CTRL]S)
20	024	14	DC4 ([CTRL]T)
21	025	15	NAK ([CTRL]U)
22	026	16	SYN ([CTRL]V)
23	027	17	ETB ([CTRL]W)
24	030	18	CAN ([CTRL]X)
25	031	19	EM ([CTRL]Y)
26	032	1A	SUB ([CTRL]Z)
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	Ø
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	↑
95	137	5F	←
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~

III.2 CPC464s indbyggede karaktersæt

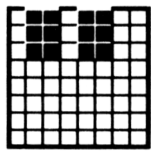
Karaktererne, som gengives her, bliver tegnet på den standard 8★8 matrice som hver celle på skærmen udgør, på CPC464. Brugerdefinerbare karakterer kan grupperes for at opnå specielle virkninger, i form af sammensatte billeder. Se beskrivelsen af kommandoen SYMBOL i kapitel 8.



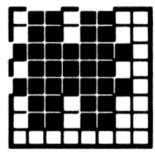
32 &H20
&X00100000



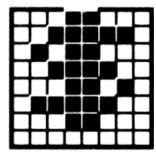
33
&H21
&X00100001



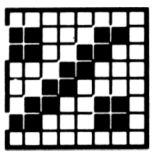
34
&H22
&X00100010



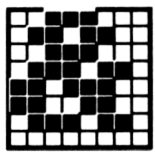
35
&H23
&X00100011



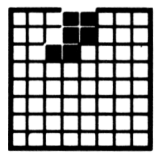
36
&H24
&X00100100



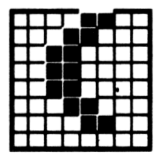
37
&H25
&X00100101



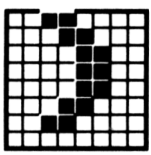
38
&H26
&X00100110



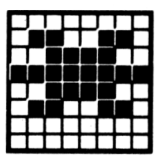
39
&H27
&X00100111



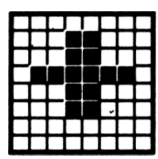
40
&H28
&X00101000



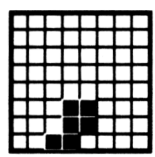
41
&H29
&X00101001



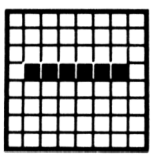
42
&H2A
&X00101010



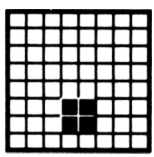
43
&H2B
&X00101011



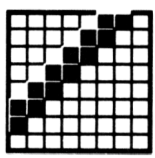
44
&H2C
&X00101100



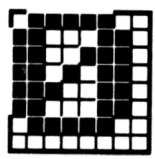
45
&H2D
&X00101101



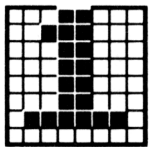
46
&H2E
&X00101110



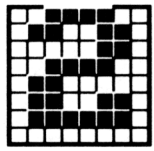
47
&H2F
&X00101111



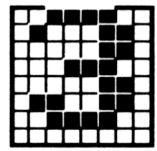
48
&H30
&X00110000



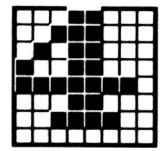
49
&H31
&X00110001



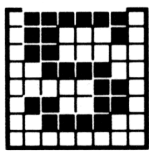
50
&H32
&X00110010



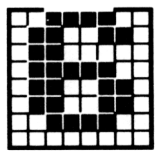
51
&H33
&X00110011



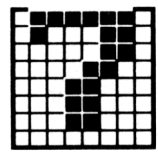
52
&H34
&X00110100



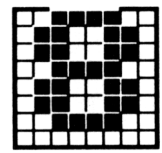
53
&H35
&X00110101



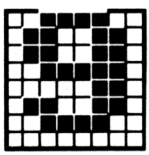
54
&H36
&X00110110



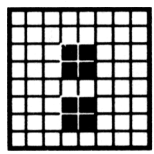
55
&H37
&X00110111



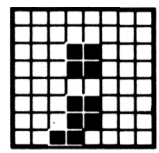
56
&H38
&X00111000



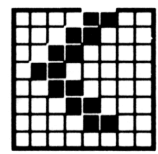
57
&H39
&X00111001



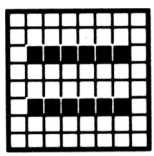
58
&H3A
&X00111010



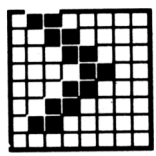
59
&H3B
&X00111011



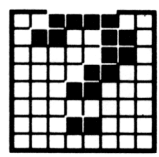
60
&H3C
&X00111100



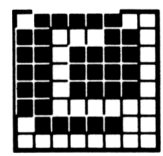
61
&H3D
&X00111101



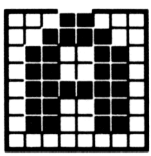
62
&H3E
&X00111110



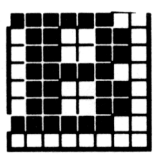
63
&H3F
&X00111111



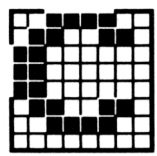
64
&H40
&X01000000



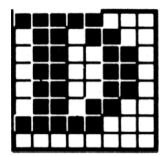
65
&H41
&X01000001



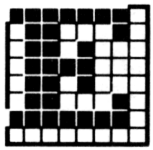
66
&H42
&X01000010



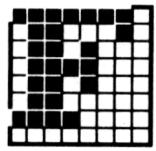
67
&H43
&X01000011



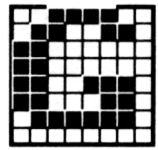
68
&H44
&X01000100



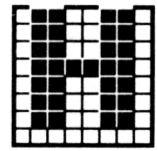
69
&H45
&X01000101



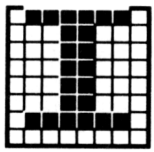
70
&H46
&X01000110



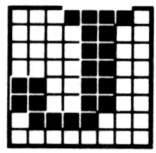
71
&H47
&X01000111



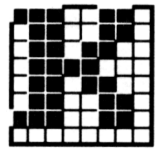
72
&H48
&X01001000



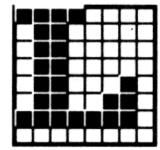
73
&H49
&X01001001



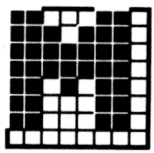
74
&H4A
&X01001010



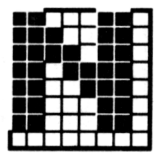
75
&H4B
&X01001011



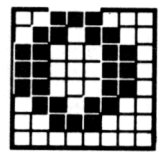
76
&H4C
&X01001100



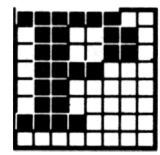
77
&H4D
&X01001101



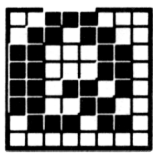
78
&H4E
&X01001110



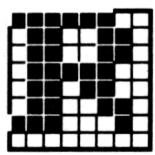
79
&H4F
&X01001111



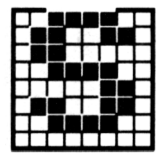
80
&H50
&X01010000



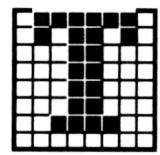
81
&H51
&X01010001



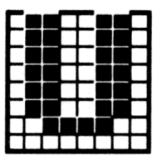
82
&H52
&X01010010



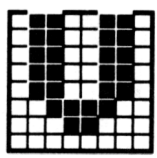
83
&H53
&X01010011



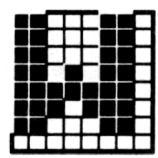
84
&H54
&X01010100



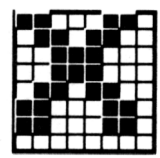
85
&H55
&X01010101



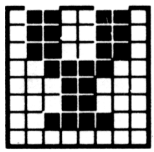
86
&H56
&X01010110



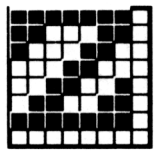
87
&H57
&X01010111



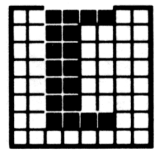
88
&H58
&X01011000



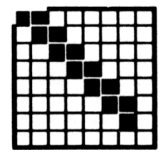
89
&H59
&X01011001



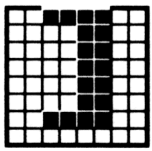
90
&H5A
&X01011010



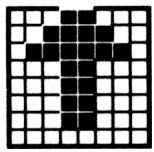
91
&H5B
&X01011011



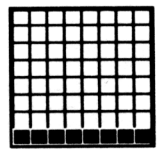
92
&H5C
&X01011100



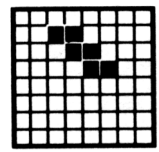
93
&H5D
&X01011101



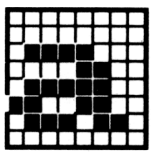
94
&H5E
&X01011110



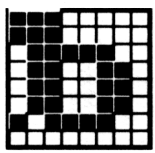
95
&H5F
&X01011111



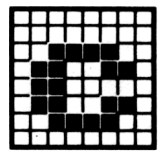
96
&H60
&X01100000



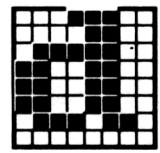
97
&H61
&X01100001



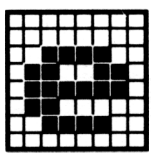
98
&H62
&X01100010



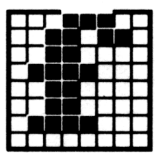
99
&H63
&X01100011



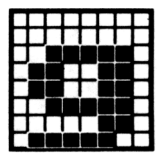
100
&H64
&X01100100



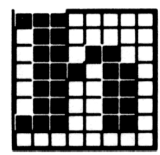
101
&H65
&X01100101



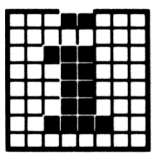
102
&H66
&X01100110



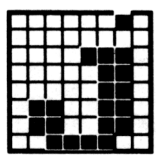
103
&H67
&X01100111



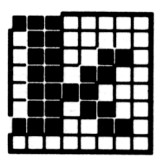
104
&H68
&X01101000



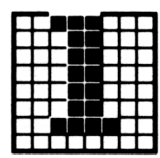
105
&H69
&X01101001



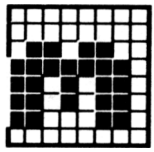
106
&H6A
&X01101010



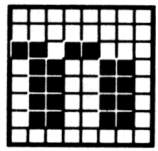
107
&H6B
&X01101011



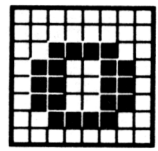
108
&H6C
&X01101100



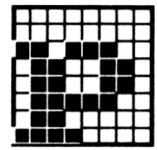
109
&H6D
&X01101101



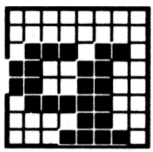
110
&H6E
&X01101110



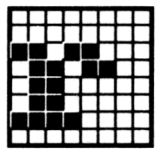
111
&H6F
&X01101111



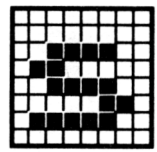
112
&H70
&X01110000



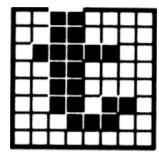
113
&H71
&X01110001



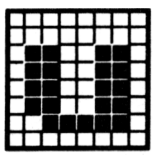
114
&H72
&X01110010



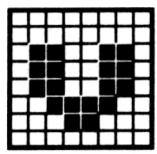
115
&H73
&X01110011



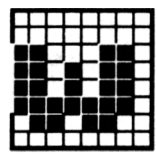
116
&H74
&X01110100



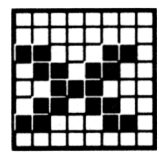
117
&H75
&X01110101



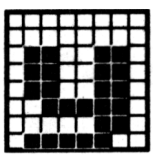
118
&H76
&X01110110



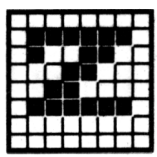
119
&H77
&X01110111



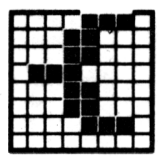
120
&H78
&X01111000



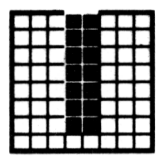
121
&H79
&X01111001



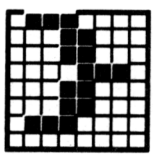
122
&H7A
&X01111010



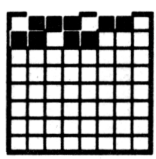
123
&H7B
&X01111011



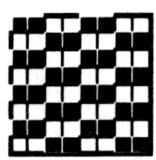
124
&H7C
&X01111100



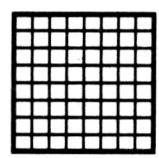
125
&H7D
&X01111101



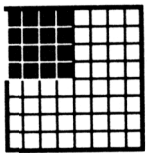
126
&H7E
&X01111110



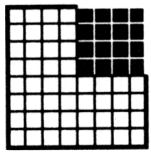
127
&H7F
&X01111111



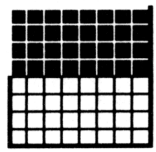
128
&H80
&X10000000



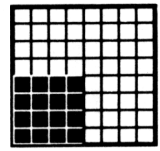
129
&H81
&X10000001



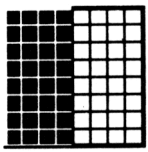
130
&H82
&X10000010



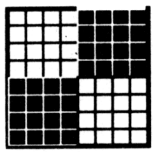
131
&H83
&X10000011



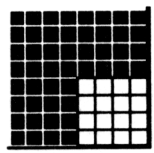
132
&H84
&X10000100



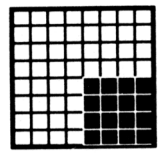
133
&H85
&X10000101



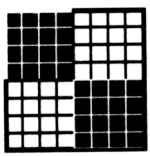
134
&H86
&X10000110



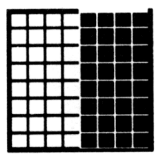
135
&H87
&X10000111



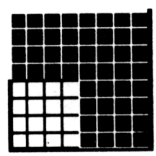
136
&H88
&X10001000



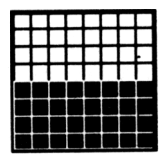
137
&H89
&X10001001



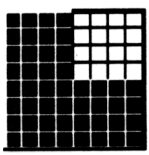
138
&H8A
&X10001010



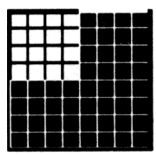
139
&H8B
&X10001011



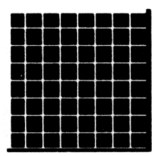
140
&H8C
&X10001100



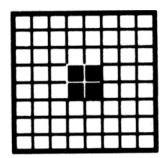
141
&H8D
&X10001101



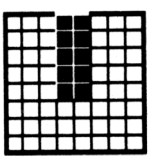
142
&H8E
&X10001110



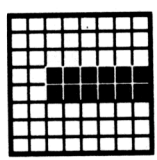
143
&H8F
&X10001111



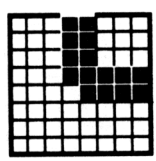
144
&H90
&X10010000



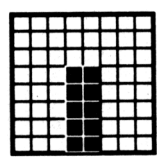
145
&H91
&X10010001



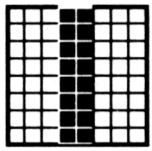
146
&H92
&X10010010



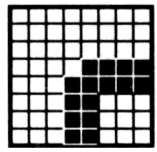
147
&H93
&X10010011



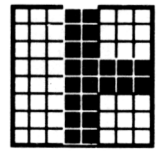
148
&H94
&X10010100



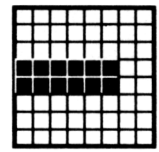
149
&H95
&X10010101



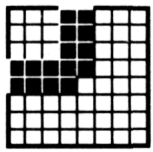
150
&H96
&X10010110



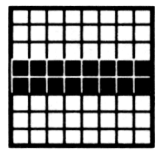
151
&H97
&X10010111



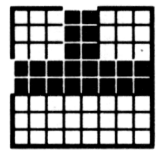
152
&H98
&X10011000



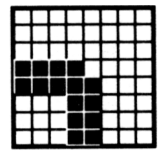
153
&H99
&X10011001



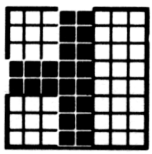
154
&H9A
&X10011010



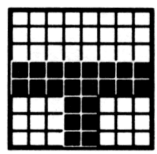
155
&H9B
&X10011011



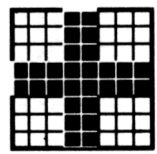
156
&H9C
&X10011100



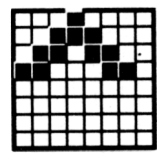
157
&H9D
&X10011101



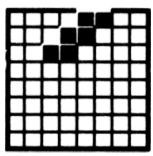
158
&H9E
&X10011110



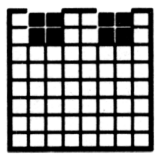
159
&H9F
&X10011111



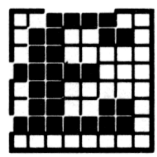
160
&HA0
&X10100000



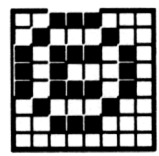
161
&HA1
&X10100001



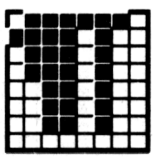
162
&HA2
&X10100010



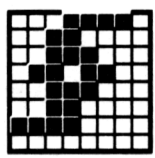
163
&HA3
&X10100011



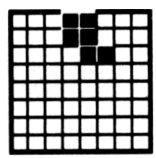
164
&HA4
&X10100100



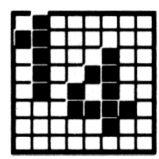
165
&HA5
&X10100101



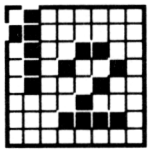
166
&HA6
&X10100110



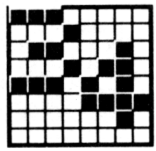
167
&HA7
&X10100111



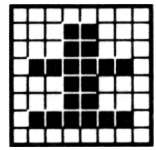
168
&HA8
&X10101000



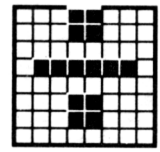
169
&HA9
&X10101001



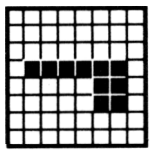
170
&HAA
&X10101010



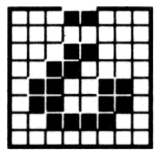
171
&HAB
&X10101011



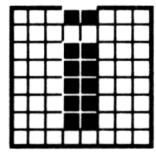
172
&HAC
&X10101100



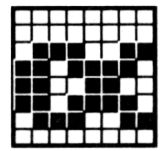
173
&HAD
&X10101101



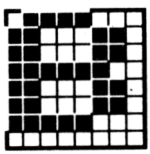
174
&HAE
&X10101110



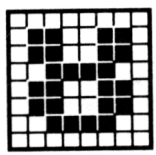
175
&HAF
&X10101111



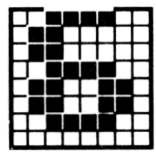
176
&HB0
&X10110000



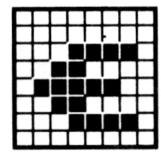
177
&HB1
&X10110001



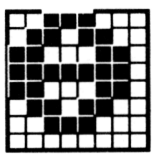
178
&HB2
&X10110010



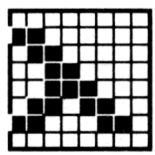
179
&HB3
&X10110011



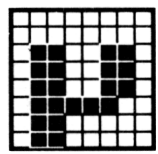
180
&HB4
&X10110100



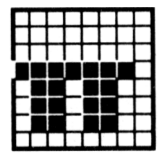
181
&HB5
&X10110101



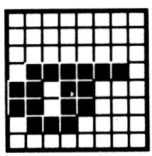
182
&HB6
&X10110110



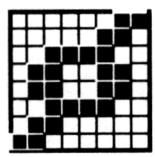
183
&HB7
&X10110111



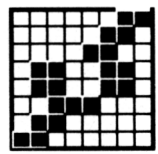
184
&HB8
&X10111000



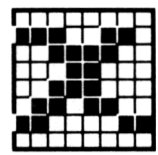
185
&HB9
&X10111001



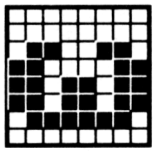
186
&HBA
&X10111010



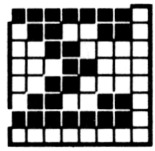
187
&HBB
&X10111011



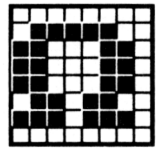
188
&HBC
&X10111100



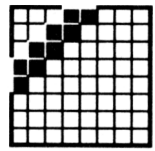
189
&HBD
&X10111101



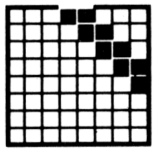
190
&HBE
&X10111110



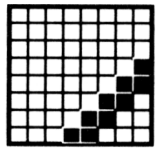
191
&HBF
&X10111111



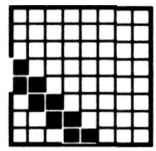
192
&HC0
&X11000000



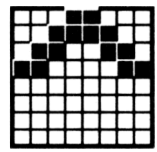
193
&HC1
&X11000001



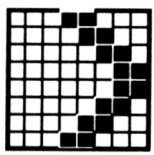
194
&HC2
&X11000010



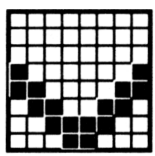
195
&HC3
&X11000011



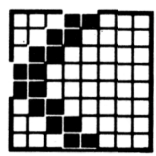
196
&HC4
&X11000100



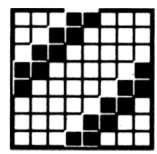
197
&HC5
&X11000101



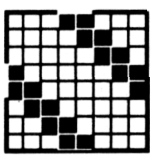
198
&HC6
&X11000110



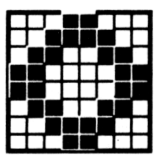
199
&HC7
&X11000111



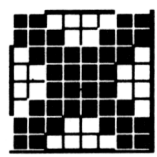
200
&HC8
&X11001000



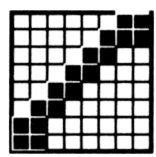
201
&HC9
&X11001001



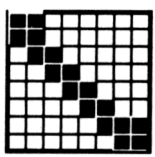
202
&HCA
&X11001010



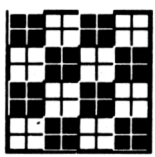
203
&HCB
&X11001011



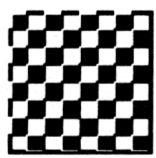
204
&HCC
&X11001100



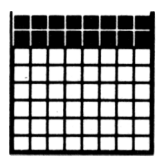
205
&HCD
&X11001101



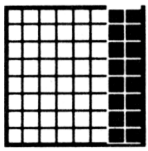
206
&HCE
&X11001110



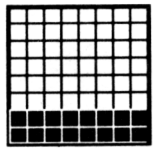
207
&HCF
&X11001111



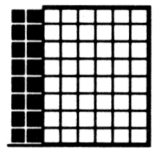
208
&HD0
&X11010000



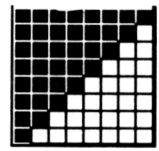
209
&HD1
&X11010001



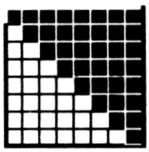
210
&HD2
&X11010010



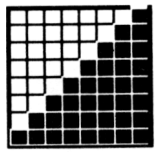
211
&HD3
&X11010011



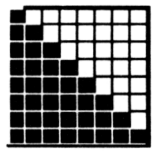
212
&HD4
&X11010100



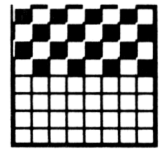
213
&HD5
&X11010101



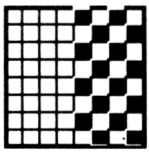
214
&HD6
&X11010110



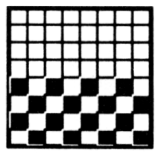
215
&HD7
&X11010111



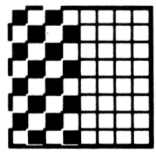
216
&HD8
&X11011000



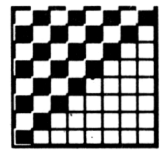
217
&HD9
&X11011001



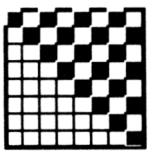
218
&HDA
&X11011010



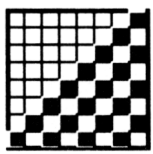
219
&HDB
&X11011011



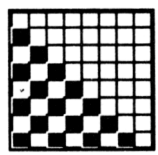
220
&HDC
&X11011100



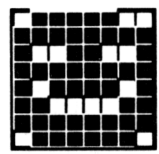
221
&HDD
&X11011101



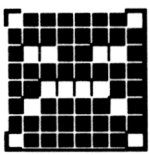
222
&HDE
&X11011110



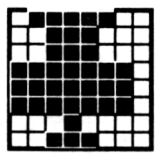
223
&HDF
&X11011111



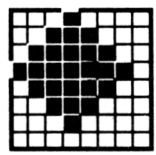
224
&HE0
&X11100000



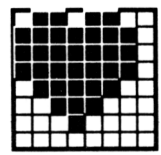
225
&HE1
&X11100001



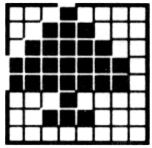
226
&HE2
&X11100010



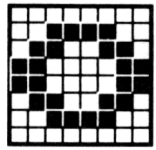
227
&HE3
&X11100011



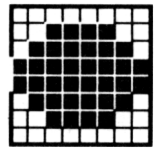
228
&HE4
&X11100100



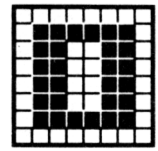
229
&HE5
&X11100101



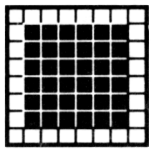
230
&HE6
&X11100110



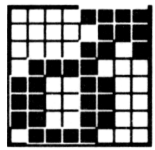
231
&HE7
&X11100111



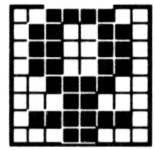
232
&HE8
&X11101000



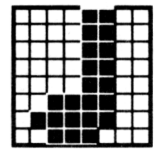
233
&HE9
&X11101001



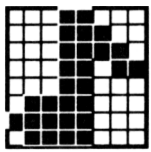
234
&HEA
&X11101010



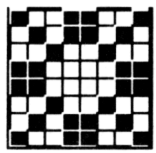
235
&HEB
&X11101011



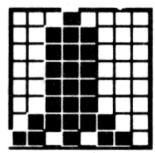
236
&HEC
&X11101100



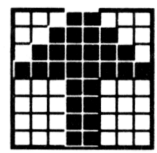
237
&HED
&X11101101



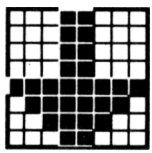
238
&HEE
&X11101110



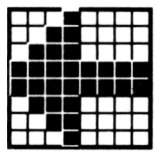
239
&HEF
&X11101111



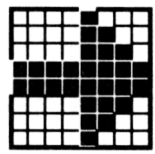
240
&HF0
&X11110000



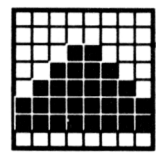
241
&HF1
&X11110001



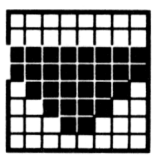
242
&HF2
&X11110010



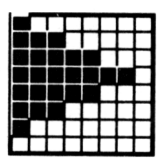
243
&HF3
&X11110011



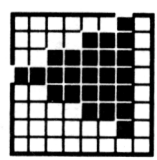
244
&HF4
&X11110100



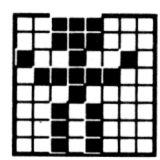
245
&HF5
&X11110101



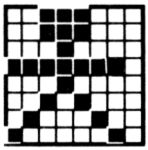
246
&HF6
&X11110110



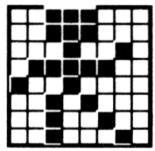
247
&HF7
&X11110111



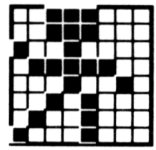
248
&HF8
&X11111000



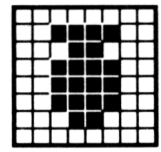
249
&HF9
&X11111001



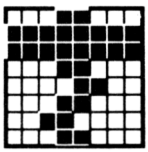
250
&HFA
&X11111010



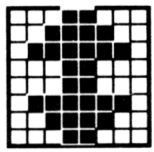
251
&HFB
&X11111011



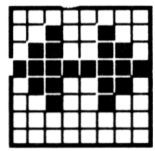
252
&HFC
&X11111100



253
&HFD
&X11111101

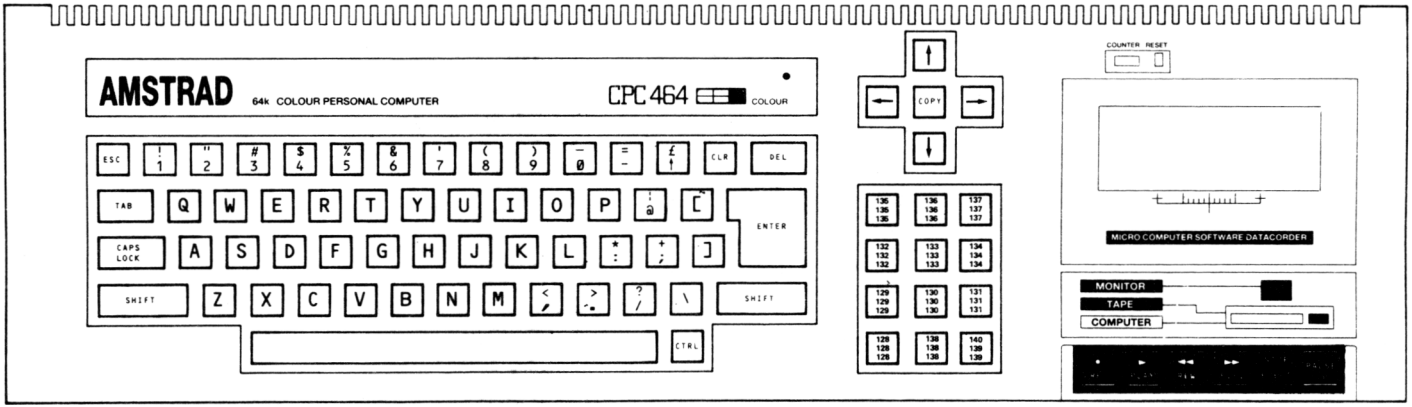


254
&HFE
&X11111110



255
&HFF
&X11111111

III Standard ASCII værdier



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

Expansion characters, default locations and values

III: Tast- og styrepindstastnumre

Appendiks IV:

Introduktion til CPC464 for erfarne brugere

CPC464 er en billig personlig farvedatamat med en rigelig, effektiv og gennemført implementation af kendte teknologier, præsenteret i et format designet til at give ekstraordinær værdi for pengene forbundet med rigelig mulighed for udvidelser, for derved at være attraktiv for både begyndere og viderekomne brugere.

Hardwaren og softwaren i ROM er designet til at give nybegyndere og erfarne programmører en nem maskine, til hvilken eksisterende software nemt kan tilpasses, og ny software skrives, for at udnytte de mange muligheder der er i CPC464.

Hovedtrækkene ved systemet er:

Z80 CPU

Den mest brugte mikroprocessor i mikrodatamater i verden, og med den bedste understøtning i for af software- især eftersom CPC464 har mulighed for indbygning af CP/M. Den unikke interruptstyringsstruktur har givet mulighed for indbygningen af nyskabelserne AFTER og EVERY i CPC464s BASIC, og andre sandtids faciliteter, som styrer lyd og timere.

64K RAM

Som standard leveres maskinen med rigelig RAM, over 42K er til rådighed for brugeren, da brugen af skygge ROM teknikker er medtaget ved indbygningen af BASICen.

Skærm

CPC464 har indbygget 3 grundlæggende skærmformater, incl. 80 kolonner tekst skærm, 27 indbyggede farver, og højopløsning op til 640★200 punkter.

Rigtigt tastatur

Et rigtigt fuld størrelse skrivemaskinetastatur med logisk markørtast placering og standard numerisk tastatur, som også bruges til brugerdefinerbare taster, er indbygget.

Indbygget kassette

Kassettebåndoptager til brug for gemning af data og programmer findes indbygget i datamaten. Derved er datamaten driftsklar uden yderligere ydre forbindelser og problemer med styrkeindstilling. Skrivehastigheden kan vælges til 1000 eller 2000 baud med enkle ordrer, idet hastigheden ved læsning bestemmes af den indbyggede software.

BASIC

En engelsk skrevet industristandard BASIC, hurtigere og kraftigere end man forventer fra den slags BASICs, med mange udvidelser for grafik og lyd.

Udvidet karaktersæt

Et fuldt 8-bit karaktersæt incl. symboler og grafiktegn er tilgængeligt hovedsageligt via tastaturet og v.h.a. CHR$(n) funktionen.

Forløben tid

Interrupts styres af billedescanningen, således at der er mulighed for at indbygge sandtids operation.

Brugerdefinerbare taster

Op til 32 taster kan brugerdefineres med op til 32 karakterstrenge. Omdefinering indeholder også gentagelseshastigheden. Alle 255 karakterer (alle ASCII plus over 100 mere), kan omdefineres af brugeren.

Subrutiner

Mange assembler subrutiner er tilgængelige for kald fra BASIC.

Skærmformater

Der er tre skærmformater til rådighed:

a) Normal

MODE 1:40 kolonner ★ 25 linier, 4 farver INK til tekst 320 ★ 200 punkter grafik, individuelt adresserbare i 4 farver.

b) Mangefarve

MODE 0:20 kolonner ★ 25 linier, 16 farver til tekst 160 ★ 200 punkter grafik, individuelt adresserbare 16 farver.

c) Højopløsning

MODE 2:80 kolonner ★ 25 linier, to farver til tekst 640 ★ 200 punkter grafik, i to farver.

Farvevalg

(NB i hele denne bog bliver sort = ingen lys, opfattet som en farve for at forenkle beskrivelsen)

Kanten kan sættes til to farver uanset skærmens format: D.v.s. blinkende; eller en enkelt farve: D.v.s. konstant.

Antallet af tilgængelige farver afhænger af skærmformatet. Hver INK kan sættes til to farver, d.v.s. blinkning; eller en enkelt farve, d.v.s. konstant farve. Antallet af blæk (INK), der kan bruges til enhver tid afhænger af skærmformat. Tekst PAPER og tekst PEN, og grafik PEN kan da sættes til en ledig INK.

Tekstudskrift kan sættes til at være gennemsigtig eller normal: D.v.s. enten af ignorere PAPER farven og overskrive grafikken, eller overskrive hele baggrunden.

Vinduer

Brugeren kan vælge op til 8 vinduer ind i hvilke karakterer kan skrives, og et grafikvindue ind i hvilket grafik kan plottes.

Vinduer nulstilles til standardværdien, når man ændrer skærmformat.

NB: Hvis tekstvinduet er lig med hele skærmen (standard), så er hurtig rulning udført af hardware. Hvis tekstvinduet mindre end hele skærmen, så udføres rulningen af software, som er tilsvarende langsommere.

Markør

Markøren slås fra i de situationer, hvor der ikke skal indtastes noget, dermed angiver markøren automatisk, at der skal indtastes noget. Markøren repræsenteres af et farvet inverst kvadrat.

Flerstemmig lyd

Lydfaciliteterne i CPC464 genereres med standardlydgenerator kredse i AY8910 familien fra General Instruments. Enheden arbejder med 3 kanaler (stemmer), som individuelt kan sættes i tone og styrke. Hvid støj kan tilføjes efter behag.

De tre kanaler fremtræder som højre, venstre og midte, når man bruger stereojack stikket. Den indbyggede højtaler giver blandet monofonisk lyd.

Softwaren giver envelope kontrol for styrke og tone. Lydgeneratorens indbyggede envelopes bruges normalt ikke.

Printerport

En standard Centronics kompatibel port er til rådighed for brugeren, hvor "busy" signalet bruges til handshake operationer.

Udvidelsesmuligheder

Mange interfaces til hardwaren kan bruges v.h.a. hopblokke, for at give mulighed for udvidelse af softwaren. AMSTRAD vil komme med bla. diskettestationer, serielle interfaces med styresoftware i ROM etc.

Koordinater

Tekstnulpunktet er øverste venstre hjørne af skærmen, og fysiske placeringer på skærmen, ændrer sig med skærmformatet.

Grafiknulpunktet er nederste venstre hjørne af skærmen, og antager at skærmen er i højopløsningsformatet hele tiden- selvom beregninger for farver vil blive udført korrekt i alle skærmformater.

NB

I normal formatet, har hvert punkt på skærmen TO vandrette adresser, og begge kan bruges. I MODE 0 har hvert punkt FIRE vandrette adresser, og enhver af disse kan bruges til at angive positionen af et punkt.

Den lodrette akse har koordinater fra 0 til 399, som deles med to og trunkeres for at give fysiske positioner i området 0-199. Dette sikrer, at skalforholdene på skærm bliver fornuftige.

Udvidelses ROMer

Alle ROMer fylder 16K hukommelse (hvor BASIC ligger) og der er faciliteter i programmellet til at kalde op til 240 ekstra 16K ROMer (idet der skal sættes en bestemt ekstra mængde adressekodnings hardware på maskinen, som en del af udvidelsesROM hardwaren).

Overblik

En kort opsumering af hovedtrækkene ved CPC464s hardware og firmware.

1) Hardware

1.1) Indeni CPC464.

Datamat, tastatur, kassettebåndoptager og højtaler. RGB og lysstyrke udgange.

1.1.1) LSI Chips

Z80A processor som kører på 4 MHz.

64Kb af 64K ★ 1 dynamisk RAM resfreshet af aflæsningen af skærmbilledet.

32Kb af ROM indeholdende BASIC og operativsystemet (OS).

Et kundedesignet logisk array indeholder næsten al logik, som ikke allerede sidder indeni LSI chips; især timing, farvegenerering og DMA kredsløb.

6845 CRT kontrol kredsen laver scaningen af TV billedet.

NB
Skærmhukommelsen er kompliceret og ændrer sig med skærmformatet. CRTCen kan bruges til at udføre sidelæns rulning (sidelæns med 1/40 skærmbrede) og rulning op og ned med 8 scanninglinier, hvis programmellet kræver dette. Parametre inden i CRTRen vælger antallet af linier, antallet af halvbilleder, bredde og position af kanterne.

GI lydgeneratorkredsen AY-3-8912: 3 stemmer. Lyd tages fra de tre kanaler og blandes ligeligt for at give en mono udgang på den indbyggede højttaler, som styres af en volumenkontrol. Der er også en ekstern udgang hvor:

Venstre = kanal A + 1/2 kanal C. Højre = kanal B + 1/2 kanal C. Denne kreds modtager også tastatur og styrepinds scanning information.

En 8255 parallel I/O kreds forbinder bussen til GI lydkredsen. Den læser også styrepindene, tastaturet, udvidelseslusene og kassetten.

1.1.2) Eksterne stik.

På printet er der to kantkonnektorer for generelle udvidelser (12) og en printer (12) (parallel, Centronics). Stik for styrepind (14) (9bens D-type), stereo lyd udgang (15), videoudgang (10) (RGB og sync eller composite video eller lysstyrke og sync.)

1.2) Udenfor kassen.

CPC464 findes med to forskellige typer monitorer. Hver indeholder en 5V strømforsyning til datamaten, konstrueret sådan at de lokale standarder for nettet i salgslandet overholdes. Derudover findes der en valgfri strømforsyning og UHF TV modulator, MP1.

Kabler er nødvendige for at forbinde datamaten til printere og stereoforstærker.

1.3) Skærmspecifikation

Skærmen arbejder på 16K hukommelse. Maskinen har et sæt på 27 farver, som kan vælges frit fra en palette. Antallet af blæk på paletten er afhængig af skærmformatet. Et antal blæk (INK) kan sættes til samme farve, hvis det er nødvendigt. Punkter på skærmen er defineret prikker af en bestemt farve. (læg mærke til, at baggrunden eller PAPER skal bruge en af blækkene fra paletten). Der er en BORDER kant rundt om den aktive del af billedet, som uafhængigt kan sættes til enhver af 27 farver.

MODE	Antal INK	lodret pkt	vandret pkt	vandret karakterer
1	4	200	320	40
2	2	200	640	80
0	16	200	160	20

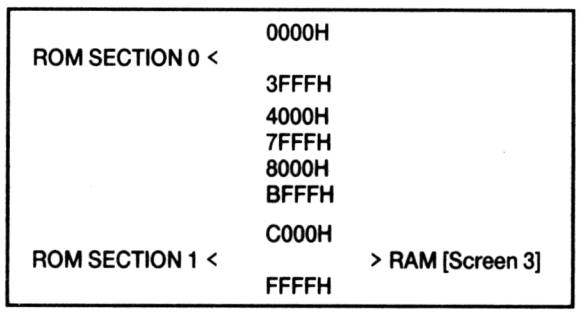
NB: Datamaten vil give en gråskala, hvis den bruges sammen med en sort/hvid monitor. Farvernes rækkefølge efter stigende lysstyrke er:

Gråskala	Farve	Gråskala	Farve
0	SORT	13	HVID
1	BLÅ	14	PASTEL BLÅ
2	DYBBLÅ	15	ORANGE
3	RØD	16	PINK
4	MAGENTA	17	PASTEL MAGENTA
5	LILLA	18	STÆRK GRØN
6	HØJRØD	19	SØ GRØN
7	VIOLET	20	STÆRK LYSEBLÅ
8	STÆRK MAGENTA	21	LIME GRØN
9	GRØN	22	PASTEL GRØN
10	LYSEBLÅ	23	PASTEL LYSEBLÅ
11	HIMMELBLÅ	24	STÆRK GUL
12	GUL	25	PASTEL GUL
		26	STÆRK HVID

1.4) Hukommelsens udlægning

De 64K RAM er fordelt således.

Bemærk at en del ROM deler adresser med skærmens RAM, for derved at frigøre størst muligt areal til brugeren af BASIC.



Når der er både RAM og ROM på en adresse, så vil læsning bruge ROMen og skrivning vil bruge RAMen. ROM sektionerne kan slås fra, hvorved RAMen kan læses fra samme adresse.

1.5) Muligheder for ekstraudstyr

1.5.1) Parallel ROMer

Der er lavet plads til tilslutning af ekstra ROM, som kan vælges i stedet for enhver del af standard ROMerne. Den nødvendige ekstra logik skal ligge i et modul tilsluttet udvidelsesbussen, men alle nødvendige signaler er ført ud til denne.

1.5.2) Ekstra RAM

Ekstra RAM kan indsættes istedet for enhver del af den indbyggede RAM. Den nødvendige ekstra logik skal ligge i et modul tilsluttet til udvidelsesbussen, men alle nødvendige signaler er ført ud til denne. Denne hukommelse kan kun læses, og en speciel teknik, med bla. I/O mapping er nødvendig for at kunne skrive i denne ekstra RAM fra datamaten.

1.5.3) Ekstra I/O

De fleste I/O porte er reserveret af datamaten, især adresser under 7Fxx må ikke bruges overhovedet. Følgende kan bruges af ekstern hardware.

F8xx,F9xx,FAxx og FBxx

Ydre enheder tilsluttet udvidelsesbussen skal dekode adresserne på A0 til A7, mens adresselinie A10 er lav. Ekspansionsbus I/O kanaler i adresseområdet F800 til FBFF er reserveret som følger:

Adresselinier A0 til A7

00-7B	**Brug ikke disse**
7C-7F	Reserveret til disketteinterface
80-BB	**Brug ikke disse**
BC-BF	Reserveret til fremtidig brug
C0-DB	**brug ikke disse**
DC-DF	Reserveret til kommunikations interface
E0-FF	Til rådighed til brugerudstyr

Bemærk at Z80 instruktioner, der placerer B registret på øverste halvdel af adressebussen (A8-A15) skal bruges.

2) Tastatur

En fuld reset kan aktiveres ved at trykke [**CTRL**] [**SHIFT**] [**ESC**] ned samtidig. Taster, der udskriver karakterer eller flytter markøren vil repetere sig selv under firmware kontrol undtagen alle taster på det numeriske tastatur.

[**ESC**] suspenderer programudførelsen. Efterfulgt af endnu et [**ESC**] afbrydes programudførelsen. Efterfulgt af enhver anden tast genoptages programudførelsen.

CAPS LOCK er en lås, som aktiveres af caps lock tasten. Shift lock er en lås som aktiveres af [**CTRL**] [**CAPS LOCK**] trykket ned samtidig.

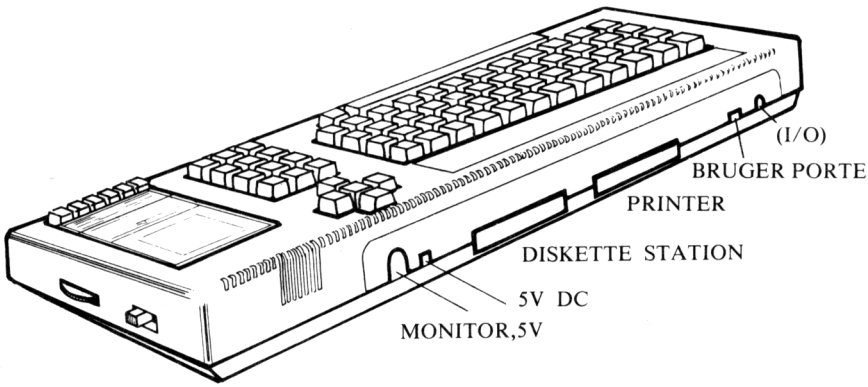
[**COPY**] CURSOR (kopi markøren) løsnes fra den almindelige markør ved at trykke [**SHIFT**] samtidig med piltasterne. Indlæsning kan opnås fra karakteren under kopimarkøren ved at trykke tasten [**COPY**].

Markørtasterne er beregnet til at tillade editering i indlæsningsbufferen, som kan fylde flere linier på skærmen. Markør tasterne kan bruges til at placere starten af en tastaturindskrivning på et givet sted på skærmen før der indskrives noget. I det øjeblik, der indtastes noget på tastaturet, vil skærmpositionen være fast. Den nye indlæste tekst vil blive skrevet oveni enhver tekst på den skærmposition.

[**DEL**] sletter baglæns og [**CLR**] sletter forlæns.

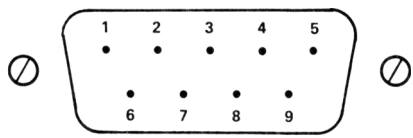
Appendiks V

Forbindelser på bagsiden af CPC464



Styreport (9 ben DIN)

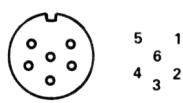
SET BAGFRA



Ben 1 OP	Ben 6 Fyr 2
Ben 2 NED	Ben 7 Fyr 1
Ben 3 Venstre	Ben 8 stel
Ben 4 Højre	Ben 9 stel 2
Ben 5 Fri	

Videoudgangs stik (6 ben DIN)

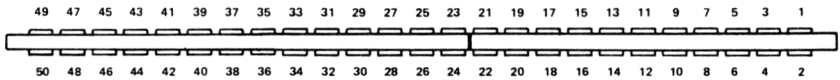
Set bagfra



Ben 1 RØD	Ben 4 Sync
Ben 2 GRØN	Ben 5 GND
Ben 3 Blå	Ben 6 lum

Udvidelses port 50 bens 0.1 Kant konnektor

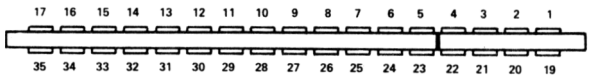
SET BAGFRA



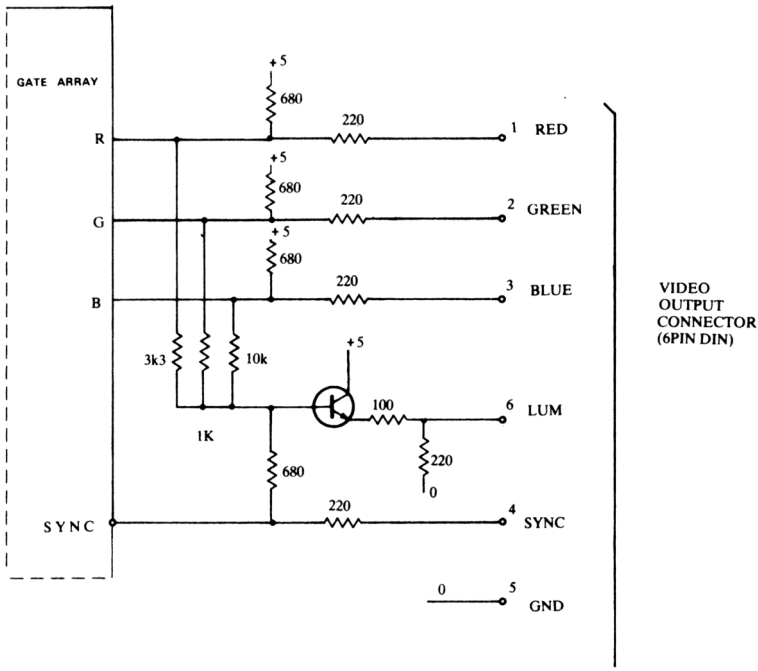
PIN	1	SOUND	PIN	18	A0	PIN	35	$\overline{\text{INT}}$
PIN	2	GND	PIN	19	D7	PIN	36	$\overline{\text{NMI}}$
PIN	3	A15	PIN	20	D6	PIN	37	$\overline{\text{BUSRD}}$
PIN	4	A14	PIN	21	D5	PIN	38	$\overline{\text{BUSAK}}$
PIN	5	A13	PIN	22	D4	PIN	39	READY
PIN	6	A12	PIN	23	D3	PIN	40	$\overline{\text{BUS RESET}}$
PIN	7	A11	PIN	24	D2	PIN	41	$\overline{\text{RESET}}$
PIN	8	A10	PIN	25	D1	PIN	42	$\overline{\text{ROMEN}}$
PIN	9	A9	PIN	26	D0	PIN	43	ROMDIS
PIN	10	A8	PIN	27	+ 5v	PIN	44	$\overline{\text{RAMRD}}$
PIN	11	A7	PIN	28	$\overline{\text{MREQ}}$	PIN	45	RAMDIS
PIN	12	A6	PIN	29	$\overline{\text{M1}}$	PIN	46	CURSOR
PIN	13	A5	PIN	30	$\overline{\text{RFSH}}$	PIN	47	L. PEN
PIN	14	A4	PIN	31	$\overline{\text{IORQ}}$	PIN	48	$\overline{\text{EXP}}$
PIN	15	A3	PIN	32	$\overline{\text{RD}}$	PIN	49	GND
PIN	16	A2	PIN	33	$\overline{\text{WR}}$	PIN	50	ϕ
PIN	17	A1	PIN	34	$\overline{\text{HALT}}$			

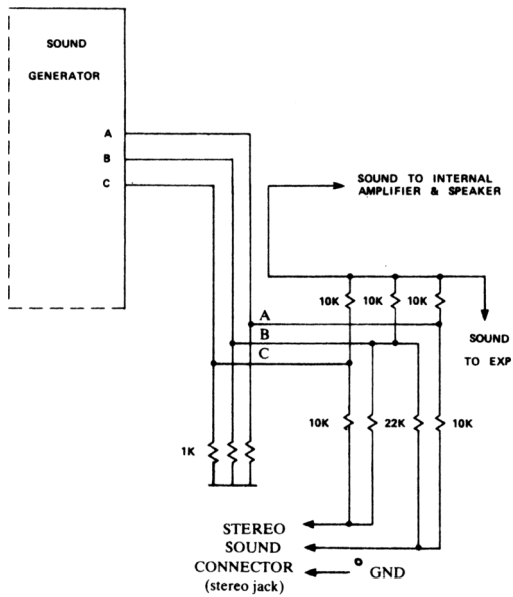
Printer port 34 bens 0.1 kantkonnektor

Set bagfra



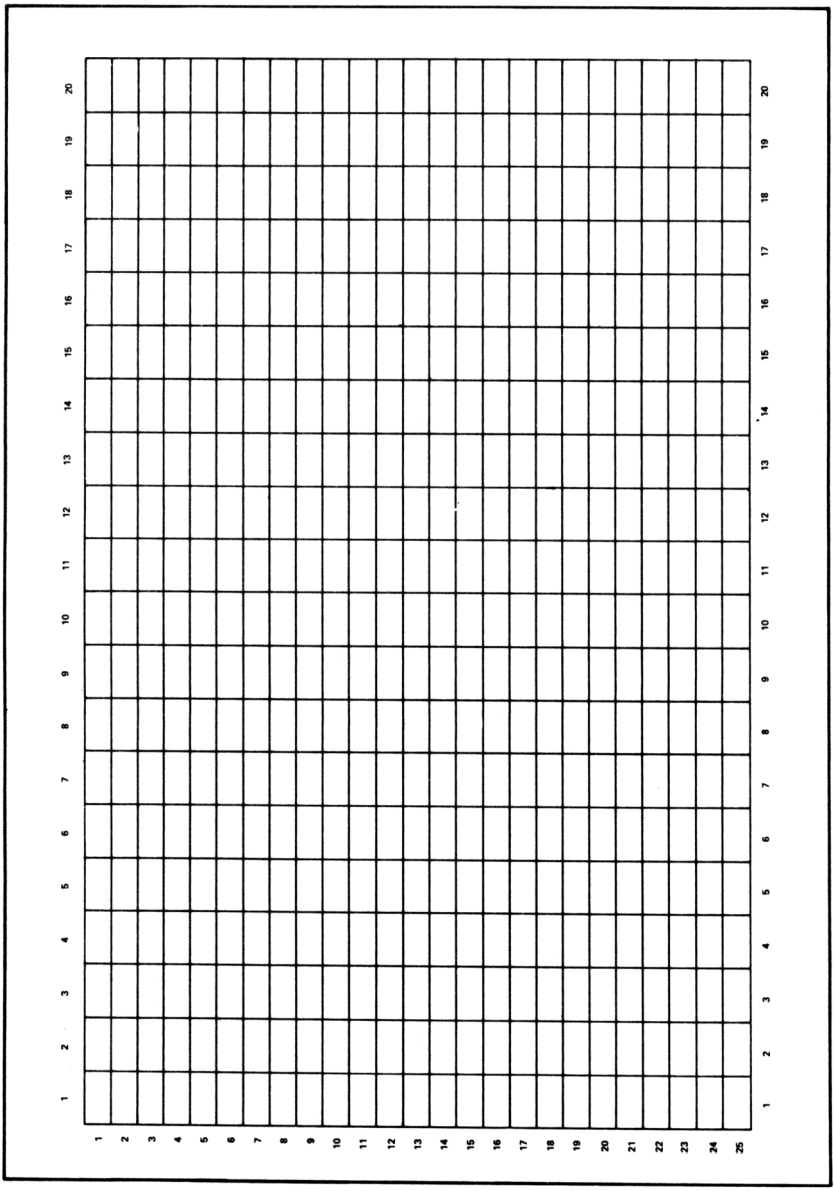
PIN	1	$\overline{\text{STROBE}}$	PIN	19	GND
PIN	2	D0	PIN	20	GND
PIN	3	D1	PIN	21	GND
PIN	4	D2	PIN	22	GND
PIN	5	D3	PIN	23	GND
PIN	6	D4	PIN	24	GND
PIN	7	D5	PIN	25	GND
PIN	8	D6	PIN	26	GND
PIN	9	D7	PIN	27	GND
PIN	11	BUSY	PIN	28	GND
PIN	14	GND	PIN	33	GND
PIN	16	GND	All other pins NC		


GATE ARRAY
R
G
B
SYNC
+5
680
220
1 RED
2 GREEN
3 BLUE
3k3
10k
1K
100
6 LUM
220
0
4 SYNC
5 GND
VIDEO
OUTPUT
CONNECTOR
(6PIN DIN)

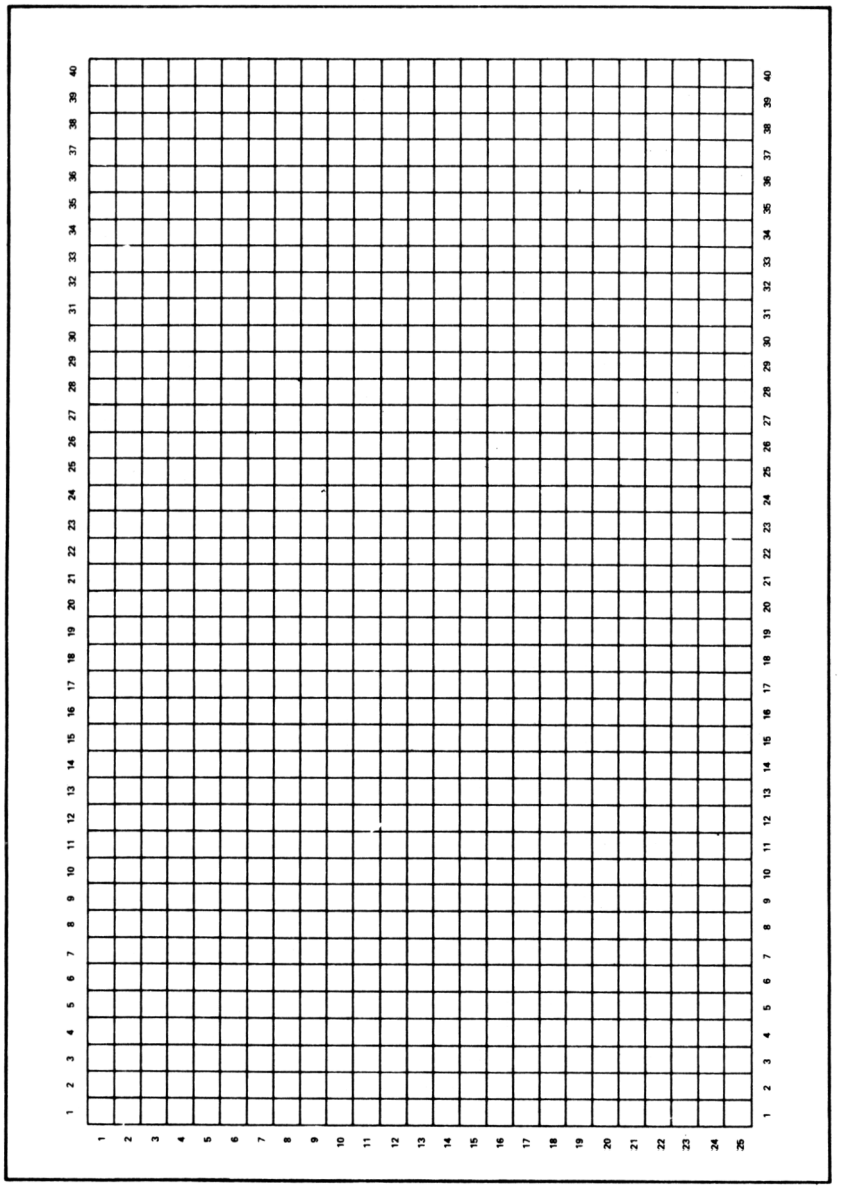

SOUND
GENERATOR
A
B
C
SOUND TO INTERNAL
AMPLIFIER & SPEAKER
10K
10K
10K
SOUND
TO EXP
10K
22K
10K
1K
STEREO
SOUND
CONNECTOR
(stereo jack)
GND

Appendiks VI:

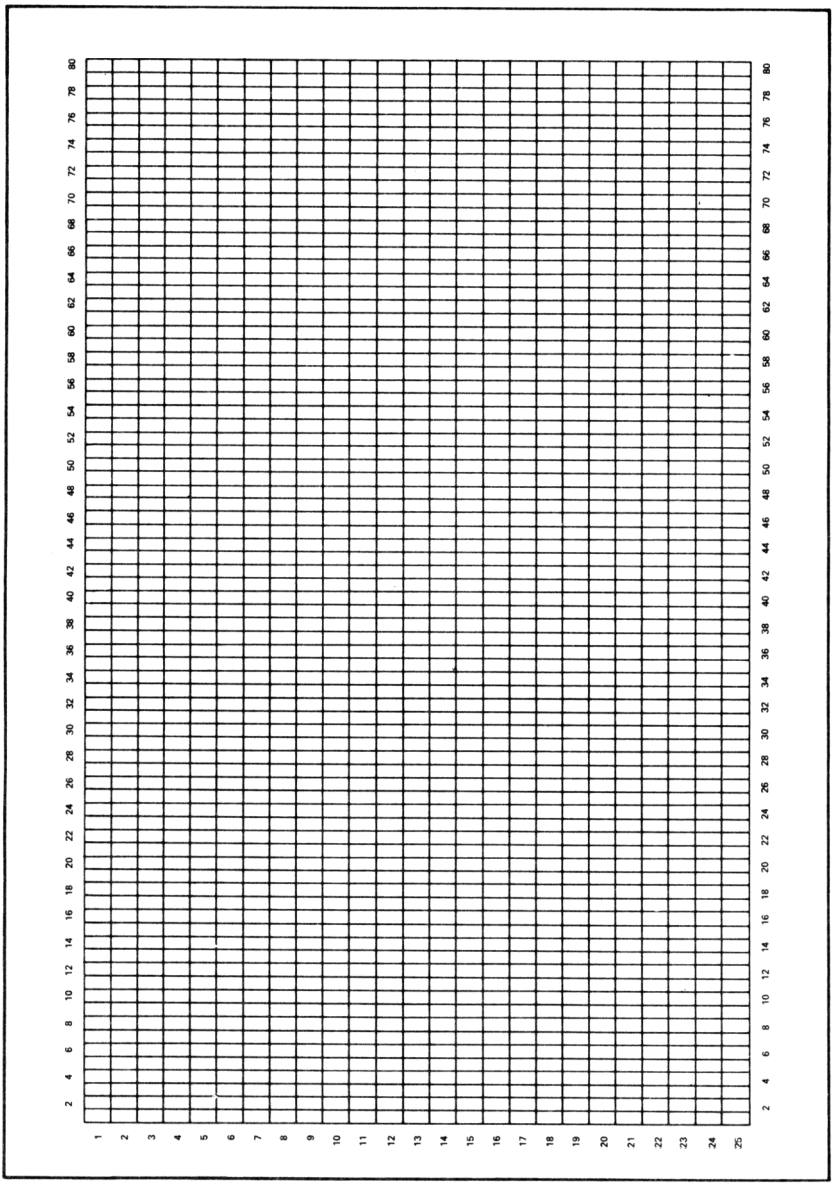
Tekst og Vindues skema
MODE 0 20 Karakterer



Tekst og Vindues skema
MODE 1 40 Karakterer



Tekst og Vindues skema
MODE 2 80 Karakterer



Appendiks VII: Toner og toneperioder

I den følgende tabel angives de anbefalede tone perioder for toner i den sædvanlige veltempererede skala i alle 8 oktaver.

De fremkomne frekvenser er ikke helt korrekte, fordi toneperioden skal være et heltal. Den relative fejl er størrelsen af differencen mellem det frekvensen er, og det frekvensen burde være, divideret med det frekvensen burde være. D.v.s. (Burde være - er/burde være).

I skemaerne herunder er NOTE = Tone, FREQENCY = FREKVENS, PERIOD = PERIODE og RELATIVE ERROR = RELATIV FEJL.

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	32.703	3822	-0.007%	
C#	34.648	3608	+0.007%	
D	36.708	3405	-0.007%	
D#	38.891	3214	-0.004%	
E	41.203	3034	+0.009%	
F	43.654	2863	-0.016%	Octave -3
F#	46.249	2703	+0.009%	
G	48.999	2551	-0.002%	
G#	51.913	2408	+0.005%	
A	55.000	2273	+0.012%	
A#	58.270	2145	-0.008%	
B	61.735	2025	+0.011%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	65.406	1911	-0.007%	
C#	69.296	1804	+0.007%	
D	73.416	1703	+0.022%	
D#	77.782	1607	-0.004%	
E	82.407	1517	+0.009%	
F	87.307	1432	+0.019%	Octave -2
F#	92.499	1351	-0.028%	
G	97.999	1276	+0.037%	
G#	103.826	1204	+0.005%	
A	110.000	1136	-0.032%	
A#	116.541	1073	+0.039%	
B	123.471	1012	-0.038%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	130.813	956	+0.046%	
C#	138.591	902	+0.007%	
D	146.832	851	-0.037%	
D#	155.564	804	+0.058%	
E	164.814	758	-0.057%	**Octave -1**
F	174.614	716	+0.019%	
F#	184.997	676	+0.046%	
G	195.998	638	+0.037%	
G#	207.652	602	+0.005%	
A	220.000	568	-0.032%	
A#	233.082	536	-0.055%	
B	246.942	506	-0.038%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	261.626	478	+0.046%	Middle C
C#	277.183	451	+0.007%	
D	293.665	426	+0.081%	
D#	311.127	402	+0.058%	
E	329.628	379	-0.057%	**Octave 0**
F	349.228	358	+0.019%	
F#	369.994	338	+0.046%	
G	391.995	319	+0.037%	
G#	415.305	301	+0.005%	
A	440.000	284	-0.032%	International A
A#	466.164	268	-0.055%	
B	493.883	253	-0.038%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	523.251	239	+0.046%	
C#	554.365	225	-0.215%	
D	587.330	213	+0.081%	
D#	622.254	201	+0.058%	
E	659.255	190	+0.206%	
F	698.457	179	+0.019%	**Octave 1**
F#	739.989	169	+0.046%	
G	783.991	159	-0.277%	
G#	830.609	150	-0.328%	
A	880.000	142	-0.032%	
A#	932.328	134	-0.055%	
B	987.767	127	+0.356%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	1046.502	119	-0.374%	
C#	1108.731	113	+0.229%	
D	1174.659	106	-0.390%	
D#	1244.508	100	-0.441%	
E	1318.510	95	+0.206%	
F	1396.913	89	-0.543%	Octave 2
F#	1479.978	84	-0.548%	
G	1567.982	80	+0.350%	
G#	1661.219	75	-0.328%	
A	1760.000	71	-0.032%	
A#	1864.655	67	-0.055%	
B	1975.533	63	-0.435%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	2093.004	60	+0.462%	
C#	2217.461	56	-0.662%	
D	2349.318	53	-0.390%	
D#	2489.016	50	-0.441%	
E	2637.021	47	-0.855%	
F	2793.826	45	+0.574%	
F#	2959.955	42	-0.548%	Octave 3
G	3135.963	40	+0.350%	
G#	3322.438	38	+0.992%	
A	3520.000	36	+1.357%	
A#	3729.310	34	+1.417%	
B	3951.066	32	+1.134%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	4186.009	30	+0.462%	
C#	4434.922	28	-0.662%	
D	4698.636	27	+1.469%	
D#	4978.032	25	-0.441%	
E	5274.041	24	+1.246%	
F	5587.652	22	-1.685%	
F#	5919.911	21	-0.548%	Octave 4
G	6271.927	20	+0.350%	
G#	6644.875	19	+0.992%	
A	7040.000	18	+1.357%	
A#	7458.621	17	+1.417%	
B	7902.133	16	+1.134%	

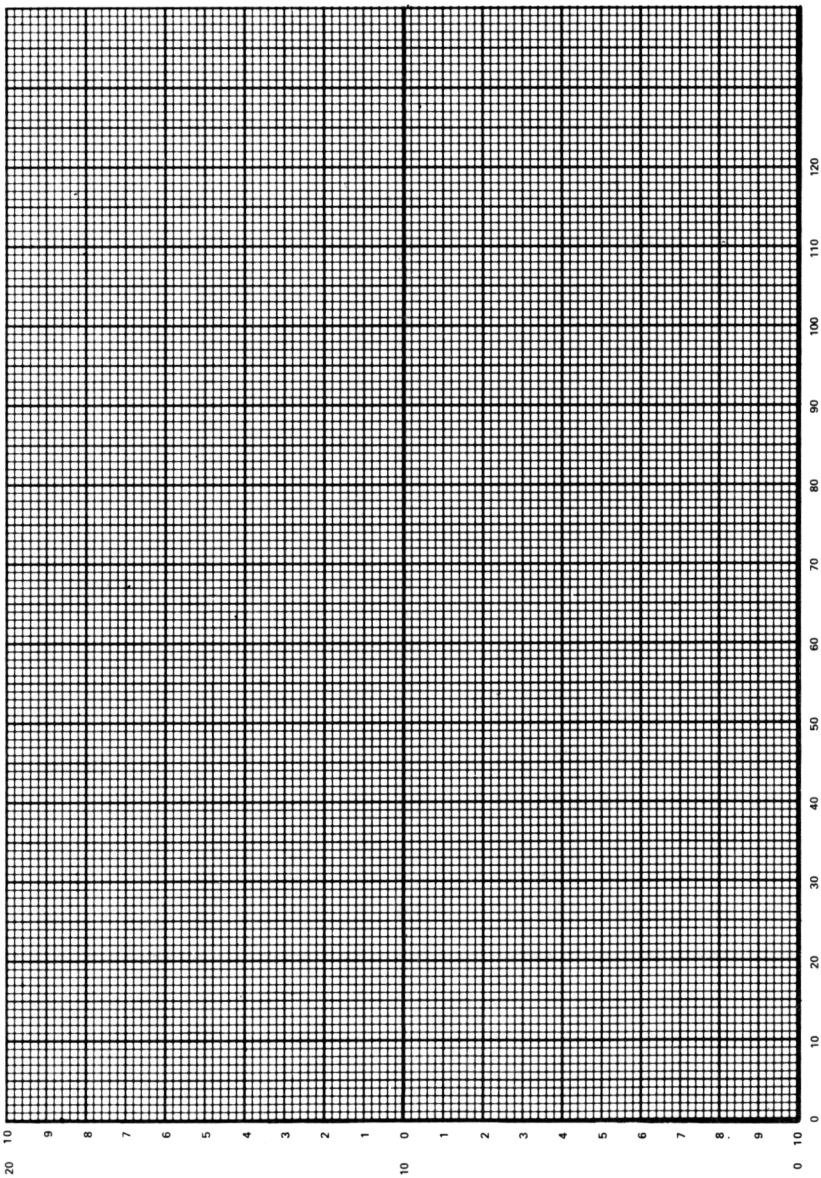
Disse værdier er alle beregnet ud fra det internationale A som følger:

Frekvens = 400 ★ (2↑(Oktav + (10-N)/12))

Periode = (125000/Frekvens) afrundet som under ROUND

Hvor N = 1 for C, N = 2 for C#, N = 3 for D etc.

Lyd envelope/ musik planlægnings ark



Appendiks VIII: Fejlkoder og reserverede ord

Fejlnumre og fejlbeskeder

Når BASIC møder en programsætning, et ord eller en variabel, som ikke kan forstås eller bearbejdes, vil det stoppe og udskrive en fejlmelding. Formen på beskeden vil normalt vise, hvad der er gået galt - og sommetider, hvis fejlen er en typografisk fejl under programindtastning, vil BASIC gå til EDIT funktionen, med linien hvori fejlen ligger.

Den mest almindelige fejl, en unøjagtigt indskrivning vil føre til, er SYNTAX ERROR (fejl nummer 2), og BASIC vil give linien ud til rettelse, hvis den mødes i et program. Hvis linien forsøges udført direkte, udskrives blot at en fejl er opstået, idet det håbes at linien er synlig, så fejlen kan ses med det samme.

Hvis ON ERROR GOTO kommandoen er indbygget i begyndelsen af programmet, kan den sende datamaten til en bestemt linie, når der opdages en fejl. I det følgende eksempel, vil datamaten gå til linie 1000, hvis der opdages en fejl:

```
10 ON ERROR GOTO 1000
program
1000 PRINT CHR$(7):MODE 2:INK 1,0:INK 0,9:CLS :LIST
```

Når denne linie mødes vil CPC464 give lyd fra sig, slette skærmen, skifte til et passende farvevalg for 80 karakterers skærmformatet, og udskrive programmet klar til undersøgelse. Hvis fejlen er en SYNTAX ERROR vil linien dukke op nederst under programudskriften, klar til at blive rettet. Beskeden SYNTAX ERROR vil dog ikke komme frem.

Husk at bruge sætningen END i sidste linie før linie 1000, hvis du ønsker at resultatet skal blive stående på skærmen.

BASIC vil ikke udskrive fejlbeskeder for korrekte inddata - så det må antages, at når en fejlkode dukker op, så kan den føres tilbage til en fejl i programmet. Det er normalt en hjælp at bruge den udskrevne besked når man skal finde fejlen. Som med de fleste ting, vil du lære af dine fejl, så udnyt det faktum, at CPC464 er den mest tålmodige lærer, du kan finde: Du vil blive træt længe før den mister sin tålmodighed!

Alle fejl som BASIC kan give er skrevet op her i rækkefølge efter fejlnummeret. Beskederne, som BASIC udskriver er med, så vel som en kort beskrivelse af de mulige fejl.

1 Unexpectet NEXT

En NEXT kommando er blevet givet udenfor en FOR løkke, eller variablen i NEXT kommandoen er ikke den samme som i FOR sætningen.

2 Syntax Error

BASIC forstår ikke den givne linie, fordi en konstruktion i linien er forbudt.

3 Unexpected RETURN

En RETURN kommando er dukket op mens datamaten ikke var i en subrutine.

4 DATA exhausted

En READ sætning har forsøgt at læse ud over den sidste DATA sætning.

5 Improper argument

Dette er en generel fejl. Værdien af en funktions argument, eller en kommando parameter er ulovlig på en eller anden måde.

6 Overflow

Resultatet af en aritmetrisk operation er blevet for stort. Dette kan være et reelt tal, og i så fald har et udtryk antaget en værdi over 1.7E-38 (tilnærmet). Alternativt, kan det være resultatet af et fejlagtigt forsøg på at ændre et reelt tal til et heltal.

7 Memory full

Det aktuelle program, eller dets variabler kan være for store, eller kontrolstrukturen kan være delt i for mange lag (lagdelte GOSUBs, WHILEs eller FORs).

En MEMORY kommando vil give denne fejl ved et forsøg på at flytte toppen af BASIC hukommelsen for langt ned, eller til en for høj værdi. Husk at åbne kassettefiler har en buffer tilknyttet, og at denne kan begrænse værdierne, som MEMORY kan bruge.

8 Line does not exist

Den kaldte linie kan ikke findes.

9 Subscript out of range

Et af indekserne i en indiceret variabel er for stort eller for lille.

10 Array already dimensioned

En af de indicerede variable i en DIM sætning er allerede erklæret.

11 Division by zero

Kan optræde i reel division, heltals divison, heltals modulus eller i potensopløftning.

12 Invalid direct command

Den sidst forsøgte direkte kommando er ikke brugbar som direkte kommando.

13 Type mismatch

En tal er angivet hvor en streng skulle bruges, eller omvendt, eller et ulovligt formet tal er fundet i en READ eller INPUT sætning.

14 String space full

Der er lavet så mange strenge at der ikke er mere plads tilbage, selv efter en "oprydning".

15 String too long

Strengen er over 255 karakterer lang. Kan opstå hvor flere strenge lægges sammen.

16 String expression too complex

Streng udtryk kan have et antal "mellemregninger". Når antallet af disse mellemregninger overskrider en rimelig grænse, så opgiver BASIC og denne fejl bliver resultatet.

17 Cannot CONTinue

Af en eller anden grund, kan programmet ikke genoptages med CONT. Bemærk at CONT er beregnet til at starte op efter en STOP kommando, [**ESC**] [**ESC**] eller en fejl, og at enhver ændring af programmet i mellemtiden, gør en genopstart umulig.

18 Unknown user funktion

Der er ikke udført nogen DEF FN sætning, til den FN der lige er blevet brugt.

19 RESUME missing

Programslutningen er blevet mødt, mens programmet behandlede en fejl (d.v.s. i en ON ERROR GOTO rutine).

20 Unexpected RESUME

RESUME kan kun bruges når datamaten behandler en fejl (d.v.s. i en ON ERROR GOTO rutine).

21 Direct command found

Når et program skal LOADes fra kassette, men på båndet blev der fundet en linie uden linienummer.

22 Operand missing

BASIC har fundet et ukomplet udtryk.

23 Line too long

En linie, omsat til den interne form, bliver for stor.

24 EOF met

Der er gjort et forsøg på at læse udover slutningen på en fil, på kassette indlæsnings strømmen.

25 File type error

Den kassettefil, der læses, er ikke af den rigtige type. OPENIN er kun forberedt på at åbne ASCII tekstfiler. LOAD, RUN etc. er kun beregnet til at læse de filtyper SAVE laver.

26 NEXT missing

Der mangler et NEXT til en FOR løkke.

27 File already open

En OPENIN eller OPENOUT kommando er blevet udført før den foregående fil blev lukket.

28 Unknown command

BASIC kan ikke finde en modtager til en ydre kommando.

29 WEND missing

Der mangler et WEND til at afsluttet en WHILE løkke.

30 Unexpected WEND

Der er kommet et WEND uden en tilhørende WHILE løkke, eller et WEND der ikke hører til den aktuelle WHILE løkke.

BASIC Nøgleord

De følgende er BASIC nøgleord, de er reserveret og kan ikke bruges til variabelnavne.

ABS,AFTER AND,ASC,AUTO

BIN$,BORDER

CALL,CAT,CHAIN,CHR$,CINT,CLEAR,CLG,CLOSEIN,CLOSEOUT,CLS,CONT,COS, CREAL

DATA,DEF,DEFINT,DEFREAL,DEFSTR,DEG,DELETE,DI,DIM,DRAW,DRAWR

EDIT,EI,ELSE,END,ENT,ENV,EOF,ERASE,ERL,ERR,ERROR,EVERY,EXP

FIX,FN,FOR,FRE

GOSUB,GOTO

HEX$,HIMEM

IF,INK,INKEY,INKEY$,INP,INPUT,INSTR,INT

JOY

KEY

LEFT$,LEN,LET,LINE,LIST,LOAD,LOCATE,LOG,LOG10,LOWER$

MAX,MEMORY,MERGE,MID$,MIN,MOD,MODE,MOVE,MOVER

NEXT,NEW,NOT

ON,ON BREAK,ON ERROR GOTO, ON SQ, OPENIN,OPENOUT,OR,ORIGIN,OUT

PAPER,PEEK,PEN,PI,PLOT,PLOTR,POKE,POS,PRINT

RAD,RANDOMIZE,READ,RELASE,REM,REMAIN,RENUM,RESTORE,RESUME,RETURN, RIGTH$,RND,ROUND,RUN

SAVE,SGN,SIN,SOUND,SPACE$,SPC,SPEED,SQ,SQR,STEP,STOP,STR$,STRING$, SWAP,SYMBOL

TAB,TAG,TAGOFF,TAN,TEST,TESTR,THEN,TIME,TO,TROFF,TRON

UNT,UPPER$,USING

VAL,VPOS

WAIT,WEND,WHILE,WIDTH,WINDOW,WRITE

XOR,XPOS

YPOS

ZONE

Appendiks IX
Danske karakterer på din AMSTRAD

Med følgende program er det muligt at få danske karakterer på din AMSTRAD CPC464.

Programmet omdefinerer udseendet af karaktererne på de taster, hvor en dansk skrivemaskine har Æ,Ø og Å.

Udover at ændre datamatens tegnsæt, bliver tasternes betydning også ombyttet, således at placeringen bliver som på en dansk skrivemaskine.

Da datamaten ikke ved at disse tegn skal opfattes som bogstaver, vil tasten [**CAPS LOCK**] ikke virke på disse bogstaver. Derfor vil programmet spørge dig, om du vil have store eller små bogstaver som "uskiftet" tegn. Svar på dette med det, du er vant til at skrive med (store eller små bogstaver som standard).

Gem det indtastede program på en tom kassette, så du nemt kan finde programmet og køre det ind, hver gang du tænder for datamaten.

```
10 SYMBOL AFTER 91
20 SYMBOL 91,126,216,216,254,216,216,222
30 SYMBOL 92,118,200,214,214,214,38,220
40 SYMBOL 93,56,0,124,198,254,198,198
50 SYMBOL 123,0,0,116,26,126,216,110
60 SYMBOL 124,0,0,118,204,214,102,220
70 SYMBOL 125,48,0,120,12,124,204,118
80 INPUT "VIL DU HAVE STORE BOGSTAVER SOM DIREKTE TRYK ELLER
SOM SKIFTET TRYK (D/S) ?";A$
90 A0=91:B0=92:C0=93
100 A1=123:B1=124:C1=125
110 IF A$="S" THEN X=A1:A1=A0:A0=X:X=B1:B1=B0:B0=X:X=C1:
C1=C0:C0=X
120 KEY DEF 29,1,A0,A1
130 KEY DEF 28,1,B0,B1
140 KEY DEF 26,1,C0,C1
150 KEY DEF 22,1,60,62
160 KEY DEF 19,1,64,39
170 KEY DEF 17,1,43,42
180 KEY DEF 39,1,44,59
190 KEY DEF 31,1,46,58
```

Indeks

CPC 464 BRUGER-VEJLEDNING