COMMODORE C 16 & plus/4

Brugervejledning

Commodore

FORORD

Denne brugervejledning anvendes til de to nyeste hjemmecomputere fra Commodore:

Commodore 16 og Commodore Plus/4

Disse to computere adskiller sig kun fra hinanden på 3 punkter: 1) Hukommelsesstørrelse, 2) Design og tastatur, 3) Plus/4 har indbygget software og rs-232 interface.
Programmeringssprog (BASIC 3.5) og hjælpefaciliteter er helt identiske for de to computere.

Denne brugervejledning henvender sig til alle brugere. Både den der for første gang får en computer mellem hænderne, den der har „snust" lidt til EDB og den der har det hele i rygraden og kan håndtere en computer med to fingre og et halvt øje.
Uanset hvilket niveau du befinder dig på, bør du læse de to første kapitler i brugervejledning, som gennemgår opstilling, tastaturbetjening og brug af færdigt programmel. Ønsker du kun at bruge færdigt programmel, kan du nøjes med at læse disse to kapitler.
Har du mod på at lære at programmere, kan du fortsætte med den første indføring i BASIC. Efterhånden som du kommer frem i bogen, får du også at se hvor let det er at lave grafik og lyd.
Hører du til de mere erfarne programmører, er det nok ting du kan gå let henover.
Commodore Basic 3.5 præsenterer så mange nyheder til Basic sproget, at det under alle omstændigheder er nødvendigt at lære de nye ting. I denne brugervejledning er der masser af eksempler, der giver en hurtig indføring i Basic 3.5
Når du er blevet dus med din computer, findes der bag i bogen en række tillæg, som kan bruges som opslagværker.

God fornøjelse!

Terminologi og „fælder“

Pas på at et 1 (ettal) ikke forveksles med bogstavet l (som i „lille“) og tallet 0 (nul) ikke forveksles med bogstaverne o og O (som i „ord“ og „Ole“). Der er taster for både 1 (et) og 0 (nul).
Pas på: en BASIC-linie kan strække sig over to skærmlinier, ja endog over flere, hvis der anvendes forkortelser.
Enkelte steder bliver tegnene <> brugt for at vise, at der er tale om en tast med den pågældende inskription. F.eks. betyder <SHIFT>, at der skal trykkes på SHIFT-tasten.
NB! Når du taster programeksempler ind, skal du huske at trykke på RETURN-tasten efter hver programlinie.

Tegnforklaring til programlister.

Farver

Tastatur	Farve	Gengivelse Sæt1	Gengivelse Sæt2	Tastatur	Farve	Gengivelse Sæt1	Gengivelse Sæt2
<CTRL><1>	sort			C=+<1>	orange		
<CTRL><2>	hvid			C=+<2>	brun		
<CTRL><3>	rød			C=+<3>	gulgrøn		
<CTRL><4>	cyan			C=+<4>	lyserød		
<CTRL><5>	violet			C=+<5>	blågrøn		
<CTRL><6>	grøn			C=+<6>	lyseblå		
<CTRL><7>	blå			C=+<7>	mørkeblå		
<CTRL><8>	gul			C=+<8>	lysegrøn		

Hvis der er monteret dansk karaktersæt, bliver pundtegnet (£) til ø.

Markørfunktioner

Funktion	Gengivelse Sæt1	Gengivelse Sæt2	Funktion	Gengivelse Sæt1	Gengivelse Sæt2
Markør til venstre			HOME		
Markør til højre			RVS ON		
Markør op			RVS OFF		
Markør ned			FLASH ON		
CLR			FLASH OFF		

Hvis der er monteret dansk karaktersæt, bliver [til æ,] til å.

Indholdsfortegnelse

Sats: Satsform, Åbyhøj

Tryk: A-offset, Holstebro

KAPITEL 1

Udpakning og tilslutning af Commodore C 16

* Udpakning af Commodore C 16.
* Stik, sokler og kontakter.
* Opstilling af Commodore C 16.
* Fejlsøgning.
* Brug af skærm og tastatur.

Udpakning og tilslutning

Det er meget vigtigt, at din **C 16** tilsluttes korrekt til dit fjernsyn eller din monitor, og til lysnettet.
Derfor bør du vide, hvad de forskellige tilslutninger, på din computer, kan bruges til.
Læs derfor hele dette kapitel og følg instruktionerne meget nøje.

Før du går igang med at stille din **C 16** op, bør du lige kontrollere, at kassen du fik din computer i, indeholder de rigtige ting. Udover denne brugervejledning skal der være følgende

1. Commodore **C 16**.
2. Strømforsyning.
3. Videokabel.

Hvis der mangler noget, bedes du kontakte din Commodoreforhandler.

Stik, sokler og kontakter

1. Joy 1 og Joy 2: Spil-stik

Her tilslutter du joystik. **C 16** bruger special designede joystik, der kan fås hos din Commodore forhandler.

2. RESET-knappen

Du kan bruge RESET-knappen til to ting:

1. Du kan bruge den til at nulstille (RESETte) din computer; nøjagtig som hvis du slukker den, og tænder den igen lige efter.
   Du skal bare trykke RESET-knappen ind een gang. Husk: Når du trykker RESET-knappen ind, mister du det BASIC-program, du har i hukommelsen.*)

2. Hvis du ønsker at nulstille din computer og samtidig beholde dit BASIC-program, skal du trykke på RUN/STOP-tasten og holde den nede, samtidig med at du trykker RESET-knappen ind.
   Når du gør dette, kommer du ind i din **C 16**'s indbyggede maskinkode monitor. For at komme ud igen trykker du X og derefter <RETURN>. Nu er du tilbage i BASIC, og du har stadig dit BASIC-program intakt i hukommelsen. Prøv selv at taste LIST og <RETURN> for at få dit program frem på skærmen.

*) Når du trykker RESET-knappen ind, udfører din **C 16** automatisk en NEW-kommando, som sletter dit program.

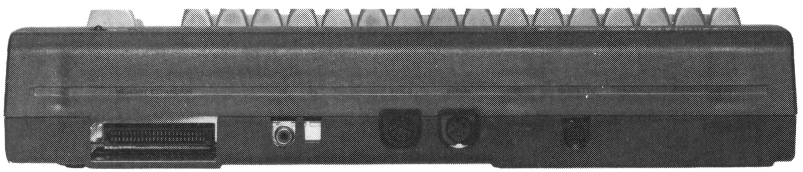
3. ON/OFF (Tænd/Sluk) kontakten

Når du isætter eller fjerner programkapsler eller stikforbindelser til ydre enheder, som f.eks. en printer eller en diskettestation, skal din **C 16** være slukket. Der sidder en rød lampe øverst til højre på din computer, så du kan se om den er slukket.

4. Strømtilslutning

Her sætter du det lille runde stik fra din strømforsyning i. Selve strømforsyningen sætter du til en almindelig stikkontakt.

Bagsiden af computeren



5. Modul-stik

Heri kan du montere Commodores programkapsler.

6. Antenne-stikket

Det er her, du forbinder din **C 16** med dit fjernsyn. Du stikker den ene ende af dit antennekabel ind her, og den anden ende tilslutter du dit fjernsyn.
Bemærk: Dette stik kan ikke bruges til forbindelse med en monitor.

7. Monitor-stikket

Her kan du forbinde din **C 16** med en monitor. Selvom dette stik er et 8-bens stik, kan du også bruge et 5-bens DIN-stik. Med Commodores farve-monitorer følger et 8-bens stik til brug sammen med **C 16**

8. Den serielle bus

Her kan du tilslutte en Commodore diskettestation eller en Commodore printer direkte. Hvis du ønsker begge tilsluttet, sætter du først din diskettestation til, og derefter sætter du kablet til fra printer ind bag i diskettestationen.

9. Datasette

Her kan du tilslutte en Commodore 1531 Datasette båndoptager.

Opstilling af din C 16

For at du kan bruge din computer, skal du bruge mindst to stikkontakter, en til din **C 16 og en til dit fjernsyn eller din monitor.**

Hvis du også skal installere diskettestation og printer, skal du bruge tilsvarende flere stikkontakter.

For at skåne øjnene bør du placere din C 16 i en behagelig afstand fra dit fjernsyn.

Før du begynder tilslutningen, bør du sørge for, at din computer er slukket. Se efter om den røde lampe øverst til højre på din computer er slukket.

Tilslutning af TV-apparat

1. Tilslut den ene ende af antenne-kablet, der medfølger, til antenne-stikket på bagsiden af computeren. Stikket passer kun på en måde.

2. Slut den anden ende af kablet til dit TV's UHF-antenneindgang.

3. Indstil nu dit TV på kanal 36 (ca.).

Tilslutning til lysnettet

1. Slut strømforsyningen til lysnettet.

2. Slut det lille runde stik fra strømforsyningen til indgangen på højre side af din **C 16**. Stikket kan ikke sidde forkert.

Nu kan du tænde din **C 16**. Kontakten sidder på højre side af computeren, når du sidder foran den.

Hvis det hele er korrekt tilsluttet, står nu dette på din skærm:

COMMODORE BASIC 3.5 12277 BYTES FREE.

READY.

Den blinkende markør under READY fortæller dig, at **C 16** er klar til at modtage instruktioner. Baggrundsfarven er hvid, bogstaverne er sorte og kantfarven er lyseblå.

Det er nok nødvendigt at finindstille dit TV. Du kan eventuelt også justere kontrasten, farverne og lydstyrken, så du får den bedst mulige gengivelse.

Hvis du herefter endnu ikke har fået det rigtige billede frem på skærmen, kan du bruge nedenstående oversigt som hjælp til at finde fejlen.

Fejlsøgning

SYMPTOM	GRUND	AFHJÆLPES VED
Den røde kontrollampe på computeren lyser ikke.	Computeren er ikke tændt.	Kontroller at både kontakten på computeren og på strømforsyningen er tændt.
	Kabel fra strømforsyningen er ikke tilsluttet.	Kontroller strømindgangen m.h.t. løst eller forkert forbundet strømkabel.
	Strømforsyningen er ikke tilsluttet.	Kontroller forbindelsen til stikkontakten.
	En sikring i systemet er sprunget.	Henvend dig til din Commodoreforhandler og få sikringen skiftet.
Den røde lampe lyser, men der er intet billede.	TV er ikke justeret.	Kontroller finindstillingen på dit TV.
	Forkert tilslutning.	Tilsluttes på UHF.
	Antennekabel ikke tilsluttet.	Kontroller kablet til dit TV.
Mærkelige mønstre på TV-skærmen med programkapslen på plads.	Programkapslen er ikke indsat korrekt.	Afbryd strømmen og indsæt kapslen igen.
Billede uden farver.	TV dårligt indstillet.	Indstil TV.
Billede med dårlige farver.	Dårlig farvejustering på dit TV.	Juster farverne på dit TV.
Billede med susende baggrundsstøj.	For kraftig lyd på TV.	Skru ned!
Billede OK, men ingen lyd.	TV lyd sat for lav.	Skru op!
	Forstærkerforbindelse ikke i orden.	Efterse ledningen til forstærker-indgang.

VIGTIGT: Der findes enkelte ældre fjernsyn, der ikke kan vise hele Commodore **16**'s skærmbillede. Det viser sig ved, at man ikke kan se yderste venstre og yderste højre kolonne på skærmen. Vi vil anbefale brug af et andet fjernsyn eller en monitor.

Er dette ikke muligt, kan problemet løses ved at trykke ESC og R. Det reducerer skærmstørrelsen til 38 kolonner, så alle kolonner kan ses på skærmen. Du skal gentage dette, hver gang du tænder eller nulstiller din computer.

Brug af skærm og tastatur

* En tur rundt på tastaturet.
* Specielle taster.
* Grafik taster.
* Programmerbare funktionstaster.
* HELP-tasten.

En tur rundt på tastaturet

De fleste af tasterne på **C 16**'s tastatur er identiske med tasterne på en almindelig skrivemaskine, men hver enkelt tast kan gøre mere end blot lave et bogstav. I dette kapitel kan du lære, hvordan man bruger alle de ekstra ting, hver tast kan. Du kan også lære, hvordan de specielle taster, som f.eks.<C=>-tasten og markør-tasterne, fungerer.

Mens du får beskrevet, hvordan de forskellige taster virker, bør du finde tasterne og bruge dem, så du ved præcis, hvordan de fungerer,

Specielle taster

RETURN-tasten

RETURN-tasten sidder yderst til højre, i tredie række på tastaturet.

Du skal trykke RETURN-tasten ned, hver gang du har indtastet en linie med instruktioner til din **C 16**. Betragt denne tast som indgangstast, idet RETURN faktisk sørger for at informationer og instruktioner "går ind" i computerens hukommelse.

SHIFT-tasterne

SHIFT- og SHIFT-LOCK-tasterne sidder på samme sted som på en skrivemaskine, altså i tredie og fjerde række.

Denne tast virker på samme måde som skifte-tasten på en almindelig skrivemaskine. Din **C 16** har to SHIFT-taster og en SHIFT-LOCK, som virker på samme måde som skifte-låsen på en skrivemaskine.

C 16 har to tegnsæt: et med store bogstaver og grafik og et med store og små bogstaver (ligesom en skrivemaskine). Når du trykker SHIFT ned sammen med en tast i tegnsæt 1 (store bogstaver og grafik), får du det grafiske tegn, der sidder til højre, forrest på tasten. Gør du det samme i tegnsæt 2, får du det store bogstav, der svarer til tasten.

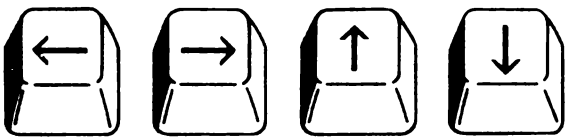
C 16 starter automatisk med tegnsæt 1, men i afsnittet om de grafiske tegn får du forklaret, hvordan du kan skifte mellem de to tegnsæt.

RUN/STOP-tasten
RUN/STOP-tasten sidder yderst til venstre i tredie række på tastaturet.

Denne tast skal du trykke på, hvis du ønsker at stoppe et program, der kører.
Hvis du trykker på denne tast, mens et program kører, er det igen dig, og ikke dit program, der har kontrollen over computeren.

Hvis du trykker SHIFT og RUN/STOP tasterne ned samtidig, henter og udfører **C 16** automatisk det første program på disketten i diskettestationen.

Markør-tasterne

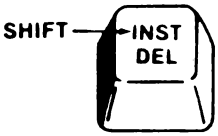


Markør-tasterne er de fire taster med pile på, der sidder i øverste række, som nr. 2,3,4 og 5 fra højre.

Det er let at flytte markøren hurtigt rundt på skærmen i alle retninger. Du skal blot trykke på den markørtast med den pil, der peger i den retning, du ønsker markøren skal flytte sig i. Som alle taster på **C 16** tastaturet, repeterer markør-tasterne automatisk, mens de holdes nede. Denne automatiske repetition gør, at markøren flytter sig, indtil du slipper tasten.

Bemærk: Du kan flytte markøren hen over alle tegn på skærmen, uden at disse ændres.

INST/DEL-tasten



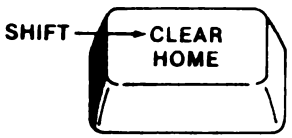
INST/DEL-tasten sidder yderst til højre i øverste række på tastaturet.

Ved hjælp af denne tast kan du indsætte og slette tegn og tal i en linie. Når du trykker tasten ned alene, forsvinder karakteren umiddelbart til venstre for markøren, og markøren rykker en plads til venstre for at dække det "hul" der bliver.
Hvis du bruger markør-tasterne til at gå ind midt i en linie for at slette med DEL, vil du kunne se, at den del af linien, der er under og til højre for markøren, også rykker en plads til venstre.

Du kan også åbne plads til at indsætte tegn. Det gør du ved at trykke SHIFT og INST/DEL ned samtidig, hvorved der åbnes plads til højre for markøren. Markøren selv bliver stående.
Når du indsætter i midten af en linie på denne måde, rykker resten af linien til højre.

INST/DEL-tasten kan spare dig for en masse arbejde, når du retter i det du allerede har skrevet.

CLEAR/HOME-tasten



CLEAR/HOME-tasten sidder yderst til højre i anden række på tastaturet. Lige over RETURN-tasten.

Denne tast tjener tre formål: hjem, sletskærm og slet vinduer. Når du trykker på denne tast, flyttes markøren straks op i øverste venstre hjørne. Dette kaldes markørens hjem. Resten af skærmen ændres ikke.
Hvis du trykker SHIFT og CLEAR/HOME tasterne ned samtidig, flyttes markøren igen hjem OG skærmen slettes. Det eneste der nu er tilbage på skærmen, er den blinkende markør i øverste venstre hjørne.
Hvis du trykker CLEAR/HOME tasten ned to gange, vil ethvert skærmvindue, du har defineret, blive slettet. Skærmvinduer er afgrænsede ar-

bejdsområder på skærmen, men det kan du læse mere om senere i denne bog.

CTRL-tasten
CTRL-tasten sidder yderst til venstre i anden række på tastaturet.

CTRL-tasten virker altid kun sammen med en anden tast. Den virker ligesom SHIFT-tasten: Du trykker den ned og holder den nede, mens du trykker på en anden tast.

1. Ved at bruge CTRL-tasten sammen med en af farvetasterne, kan du selv bestemme farven på din tekst. Afsnittet om farvetasterne forklarer nøjagtig, hvordan du bærer dig ad.

2. Du kan midlertidigt stoppe en LISTning eller et program der PRINTer på skærmen. Dette gør du ved at trykke på CTRL og holde den nede, mens du trykker på S. For at starte LISTningen eller PRINTningen igen, trykker du en tilfældig tast ned. Vi foreslår, at du altid bruger mellemrumstasten (den aflange tast nederst på tastaturet).

3. CTRL bruges også sammen med RVS ON/RVS OFF og FLASH ON/ FLASH OFF tasterne.

Udover dette bruges CTRL-tasten også af nogle af de programmer, du kan købe til din **C 16**.

<C=>-tasten
<C=>-tasten sidder yderst til venstre i fjerde række på tastaturet.

<C=>-tasten bruges ligesom CTRL-tasten også sammen med andre taster.

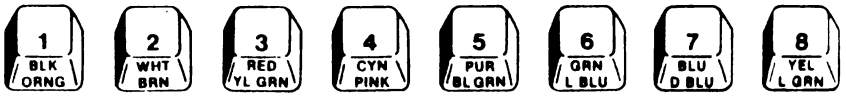
1. Når du bruger SHIFT og <C=>-tasterne sammen, kan du skifte mellem de to tegnsæt på din **C 16** – store bogstaver/grafik og store/små bogstaver.

2. Uanset hvilket tegnsæt du er i, fungerer <C=>-tasten som en slags skiftetast til de grafiske tegn, der sidder til venstre på forsiden af den pågældende tast. Du skal blot holde <C=>-tasten nede, mens du trykker den tast ned, hvor det ønskede grafiktegn sidder.

3. <C=>-tasten kan ligesom CTRL-tasten også bruges til at vælge farve på din tekst. Ved brug af <C=>-tasten i stedet for CTRL-tasten, har du adgang til yderligere otte farver.

4. Hvis du ønsker at sænke farten på en program LISTning, skal du trykke <C=>-tasten ned alene. Når du gør dette under en LISTning af et program, ruller programmet langsomt op over skærmen. Så snart du slipper tasten, ruller programmet op over skærmen med normal fart igen. Denne funktion er praktisk, idet det kan være svært at følge med, når programmet ruller hurtigt op over skærmen.

Farvetasterne



Farvetasterne er tallene fra 1 til 8. Hvis du kigger på forsiden af disse taster, står der to forskellige farver på hver tast.

Som før omtalt har du mulighed for at ændre farve på bogstaver, tal og grafiske tegn. Du har 16 farver at vælge imellem. For at ændre farven gør du følgende:

Hvis du ønsker en af de otte farver, der står øverst på tasterne (f.eks. BLK = black = sort), skal du holde CTRL-tasten nede, mens du trykker på tasten med den farve, du ønsker.

Hvis du ønsker en af de otte farver, der står nederst på tasterne (f.eks. ORNG = orange), skal du holde <C=>-tasten nede, mens du trykker på den ønskede farve.

Øv dig i at skifte farve, så du er sikker på, hvordan det virker. Du vil opdage, at når du har skiftet farve, har alle bogstaver, tal og grafiske tegn du taster herefter, den farve, du sidst har valgt.

C 16 har ialt 121 farver, men med <C=>- og CTRL tasterne kan du kun bruge de 16. Du får senere forklaret, hvordan du bruger alle de andre farver.

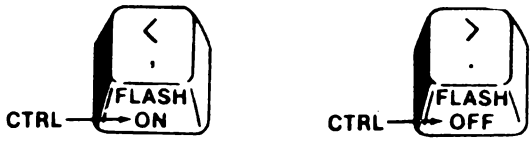
REVERSE ON og REVERSE OFF tasterne

RVS ON sidder på tasten med 9-tallet, og RVS OFF sidder på tasten med nullet.

Med **C 16** kan du også skrive tegn negativt. Med andre ord: hvis du bruger sorte bogstaver på hvid baggrund, kan du få skrevet hvide bogstaver på sort baggrund.

Det eneste du behøver at gøre er at trykke CTRL og RVS ON tasterne ned samtidig. Herefter bliver alt skrevet med negativ skrift, indtil du trykker CTRL og RVS OFF (samtidig), RETURN-tasten eller ESC og O, hvorefter du igen skriver med normal skrift.

FLASH ON og FLASH OFF tasterne



FLASH ON sidder på fjerde tast fra højre i fjerde række. FLASH OFF sidder på tasten til højre for FLASH ON.

Ligesom du kan få tegn skrevet negativt, kan du også få tegn til at blinke. Du skal blot trykke CTRL og FLASH ON ned samtidig og derefter vil alt, hvad du indtaster, blinke. Hvis du trykker CTRL og FLASH OFF (samtidig), RETURN eller ESC og O, står teksten igen normalt.

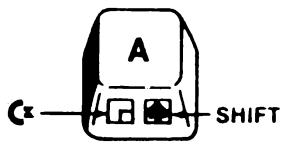
Grafik-tasterne

Som vi har nævnt før, er din **C 16** i tegnsættet med store bogstaver og grafik (tegnsæt 1), når du tænder den. Når du er i dette tegnsæt, kan du frit bruge de mere end 60 grafiske tegn, du kan se på forsiden af mange af tasterne.

Der sidder to grafiske tegn på hver grafiktast.

For at skrive det grafiske tegn til højre på tasten skal du holde SHIFT-tasten nede, mens du trykker på grafiktasten.

For at skrive det grafiske tegn til venstre skal du holde <C=>-tasten nede, mens du trykker på grafiktasten.



Ved at bruge de grafiske tegn kan du lave billeder, kort og alle mulige andre figurer, ved enten at skrive tegnene ved siden af hinanden og/eller

oven på hinanden; nøjagtig ligesom hvis du brugte byggeklodser.
Prøv at bruge de grafiske tegn så du kan se, hvordan de virker. I kapitel 7 får du endnu mere at vide om de grafiske tegn.

Du kan skifte mellem de to tegnsæt, store bogstaver/grafik og store og små bogstaver, ved at trykke SHIFT og <C=>-tasterne ned samtidig.

Ligegyldigt hvilket tegnsæt du er i, skal du altid skrive BASIC-instruktioner UDEN brug af SHIFT.

Når du er i tegnsæt 2, kan du skrive store og små bogstaver. Du kan ikke bruge grafiktegnene til højre på tasterne, men du kan stadigvæk bruge grafiktegnene til venstre. Det gør du på samme måde som før: Du holder <C=>-tasten nede, mens du trykker på den ønskede grafiktast. Grafiktegnene til venstre på tasterne er særligt velegnede til blokgrafik, rammer og andet, man specielt bruger til budgetsimulering, lagerstyring etc.

Lad os lige se det hele skematisk:

	Tegnsæt 1	Tegnsæt 2
tast alene	store bogstaver	små bogstaver
SHIFT+tast	grafik til højre	store bogstaver
<C=>+tast	grafik til venstre	grafik til venstre

Bemærk: når du skifter mellem de to tegnsæt, skifter tegnene på skærmen også. F.eks.: Skriver du et A i tegnsæt 1 og herefter skifter til tegnsæt 2, bliver det store A til et lille a. De eneste tegn, der ikke ændres ved skift mellem tegnsættene, er de grafiske tegn til venstre på tasterne.

ESC-tasten
ESC-tasten sidder til venstre i øverste række på tastaturet.

Med ESC-tasten kan du lave mange specielle rettefunktioner. Det er også denne tast du bruger til at definere skærmvinduer med. Som før nævnt får du senere mere at vide om skærmvinduer. De rettefunktioner, du kan udføre med ESC-tasten, er f.eks. indsætning og sletning af hele linier, slet resten af linien o.s.v.

ESC-tasten bruges normalt sammen med et bogstav. For at aktivere funktionen, skal du først trykke ESC-tasten ned, slippe den igen og derefter trykke et af de følgende bogstaver ned:

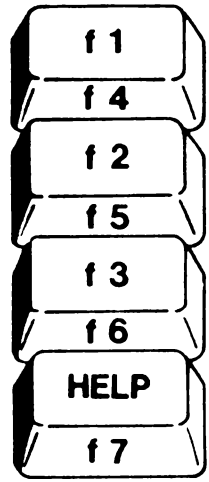
Bogstav	Funktion
A	Automatisk indsætning.
B	Sæt nederste højre hjørne af skærmvinduet her (markørens nuværende position).
C	Annuller automatisk indsætning.
D	Slet linie.
I	Indsæt linie.
J	Flyt til start af linie.
K	Flyt til slut af linie.
L	Tænd for rulning.
N	Returner til normalt skærmbillede (40*25).
O	Annuller indsæt, quote, negative og blinke kontroller.
P	Slet alt fra starten af linien til markøren.
Q	Slet alt fra markøren og linien ud.
R	Reducer skærmbilledet til 38*23 tegn.
T	Sæt øverste venstre hjørne af skærmvinduet her.
V	Skærmbilledet ruller op.
W	Skærmbilledet ruller ned.
X	Annuller ESC-tasten.

Specielle symboler
På **C 16** tastaturet kan du også se en række specielle symboler, du normalt ikke finder på en skrivemaskine. Disse symboler er: nummertegnet (#), pundtegnet (£), pi (π), større end og mindre end tegnene (><), skarpe paranteser ([ ]) og pil op (↑) (ikke markør-tasten, men den pil, der sidder på tasten med nullet (0)). Disse symboler bliver brugt til at programmere **C 16** med.
Bemærk: Har du en **C 16** med dansk tastatur, erstattes pund-tegnet og de skarpe paranteser med æ, ø, og å.

Programmerbare funktionstaster
De fire taster, der sidder særskilt til højre på dit tastatur, er specielle funktions-taster, der sparer tid for dig ved at udføre hele kommandoer ved tryk på een tast.

Du kan se hvad hver enkelt tast udfører, hvis du taster KEY og trykker på RETURN-tasten, hvorefter skærmen skriver:

```
KEY 1, "GRAPHIC"
KEY 2, "DLOAD"+CHR$(34)
KEY 3, "DIRECTORY"+CHR$(13)
KEY 4, "SCNCLR"+CHR$(13)
KEY 5, "DSAVE"+CHR$(34)
KEY 6, "RUN"+CHR$(13)
KEY 7, "LIST"+CHR$(13)
KEY 8, "HELP"+CHR$(13)
```



Her er en forklaring på hver enkelt tast's funktion:

F1 fører dig ind i en af computerens grafiktilstande, når du taster nummeret på tilstanden (f.eks. GRAPHIC 2, som er høj opløsning med delt skærmbillede) og trykker på RETURN-tasten

F2 skriver DLOAD" på skærmen, hvorefter du indtaster navnet på det program, du ønsker fra disketten, og trykker på RETURN-tasten.

F3 skriver en indholdsfortegnelse over de programmer, der ligger på disketten i diskettestationen, på skærmen.

F4 sletter skærmen (også i en af grafiktilstandene).

F5 skriver DSAVE" på skærmen, hvorefter du indtaster navnet på det program, du ønsker gemt på disketten, og trykker på RETURN-tasten.

F6 udfører det program, du har liggende i computeren.

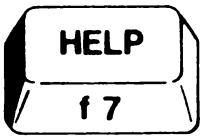
F7 skriver en listning af det program, du har liggende i computeren, på skærmen.

K8 HELP-tasten eller Hjælpe-tasten får fejl i programinstruktioner til at blinke.

For at bruge disse funktioner trykker du blot på den pågældende funtionstast. Du skal bruge SHIFT-tasten, hvis du vil bruge F4, F5, F6 eller F7.

Du kan til enhver tid omdefinere funktionstasterne, så de udfører det, du har brug for. Dette gør du ved at bruge KEY instruktionen. Du kan omdefinere funktionstasterne i et BASIC-program eller direkte på skærmen. Du kan omdefinere alle funktionstaster, ligeså mange gange du vil.
Bemærk: De omdefinerede funktioner forsvinder, så snart du slukker for computeren, og når du tænder den, er det igen ovenstående definitioner der gælder.

HELP- eller Hjælpetasten



Hjælpetasten er den funktionstast, der sidder nederst.

Hvis du laver fejl i et program, skriver **C 16** en fejlmeddelelse ud, for at fortælle dig, hvad du har lavet forkert. Disse fejlmeddelelser står yderligere forklaret i afsnittet om fejlmeddelelser i anden halvdel af denne brugervejledning.

Udover disse fejlmeddelelser kan du få mere hjælp til at finde fejl ved at trykke på hjælpetasten. Når du trykker på hjælpetasten efter en fejl, skriver computeren den linie, der er fejl i, og der hvor fejlen er, blinker tegnene. F.eks.:

?SYNTAX ERROR IN 10	**C 16** skriver, at der er syntaksfejl i linie 10.
HELP	Du trykker på hjælpetasten.
10 PRGNT "COMMODORE COMPUTERS"	**C 16** blinker, der hvor fejlen er.

KAPITEL 2

Udpakning og tilslutning af Commodore Plus/4

* Udpakning af Commodore Plus/4.
* Stik, sokler og kontakter.
* Opstilling af Commodore Plus/4.
* Fejlsøgning.
* Brug af skærm og tastatur.

Udpakning og tilslutning

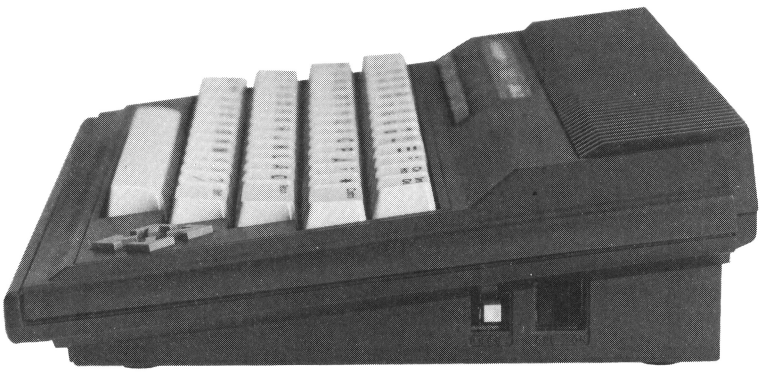
Det er meget vigtigt, at din **Plus/4** tilsluttes korrekt til dit fjernsyn eller din monitor, og til lysnettet. Derfor bør du vide, hvad de forskellige tilslutninger på din computer kan bruges til.
Læs derfor hele dette kapitel og følg instruktionerne meget nøje.

Før du går igang med at stille din **Plus/4** op, bør du lige kontrollere, at kassen du fik din computer i, indeholder de rigtige ting. Udover denne brugervejledning skal der være følgende:

1. Commodore **Plus/4**.
2. Strømforsyning.
3. Antennekabel
4. Dokumentation til de indbyggede programmer i din **Plus/4**.

Hvis der mangler noget, skal du straks kontakte din Commodoreforhandler.

Stik, Sokler og Kontakter



1. ON/OFF (Tænd /Sluk) kontakten

Når du isætter eller fjerner programkapsler eller stikforbindelser til ydre enheder, som f.eks en printer eller en diskettestation, skal din **Plus/4** være slukket. Der sidder en rød lampe nederst til venstre på din computer, så du kan se, om den er slukket.

2. RESET-knappen

Du kan bruge RESET-knappen til to ting:

1. Du kan bruge den til at nulstille (RESETte) din computer; nøjagtig som hvis du slukker den, og tænder den igen lige efter.
   Du skal bare trykke RESET-knappen ind een gang. Husk: Når du trykker RESET-knappen ind, mister du det BASIC-program, du har i **Plus/4**'s hukommelse. *)

2. Hvis du ønsker at nulstille din computer og samtidig beholde dit BASIC-program, skal du trykke på RUN/STOP-tasten og holde den nede, samtidig med at du trykker RESET-knappen ind.
   Når du gør dette, kommer du ind i din **Plus/4**'s indbyggede maskinkode-monitor. For at komme ud igen trykker du X og derefter <RETURN>. Nu er du tilbage i BASIC, og du har stadig dit BASIC-program intakt i hukommelsen. Prøv selv at taste LIST og <RETURN> for at få dit program frem på skærmen.

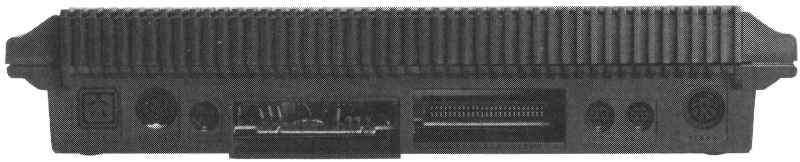
*) Når du trykker RESET-knappen ind, udfører din **Plus/4** automatisk en NEW-kommando, som sletter dit program.

Venstre side af computeren.

3. HF-Antenne-stikket

Det er her, du forbinder din **Plus/4 med dit fjernsyn.**
Du stikker den ene ende af dit antennekabel ind her, og den anden ende tilslutter du dit fjernsyn.
Bemærk: Dette stik kan ikke bruges til forbindelse med en monitor.

Bagsiden af computeren



4. Strømtilslutning

Her sætter du det firkantede stik fra din strømforsyning i. Den anden ledning på din strømforsyning sætter du til en almindelig stikkontakt.

5. Den serielle bus

Her kan du tilslutte en Commodore diskettestation eller en Commodore printer direkte.
Hvis du ønsker begge tilsluttet, sætter du først din diskettestation til, og derefter sætter du kablet til din printer ind bag i diskettestationen.

6. Datasette

Her kan du tilslutte en Commodore 1531 Datasette båndoptager.

7. RS-232 sokkel

Her kan du tilslutte en RS-232 niveau-omformer, der gør det muligt at kommunikere med enheder, der ikke er tilpasset standard for Commodore tilbehør.

8. Modul-stik

Heri kan du montere Commodores programkapsler.

9. Joy 1 og Joy 2: Spil-stik

Her tilslutter du joystik. **Plus/4** bruger special designede joystik, der kan fås hos din Commodore forhandler.

10. Monitor-stikket

Her kan du forbinde din **Plus/4** med en monitor. Selvom dette stik er et 8-bens stik, kan du også bruge et 5-bens stik. Med Commodores farvemonitorer følger et 8-bens stik til brug med **Plus/4**.

Opstilling af din Plus/4

For at du kan bruge din computer, skal du bruge mindst to stikkontakter, en til din **Plus/4** og en til dit fjernsyn eller din monitor.

Hvis du også skal installere diskettestation og printer, skal du bruge tilsvarende flere stikkontakter.

For at skåne øjnene, bør du placere din **Plus/4** i en behagelig afstand fra dit fjernsyn.

Før du begynder tilslutningen, bør du sørge for, at din computer er slukket. Se efter om den røde lampe nederst til venstre på din computer er slukket.

Tilslutning af TV-apparat

1. Tilslut den ene ende af antenne-kablet, der følger med, til video-stikket på den venstre side af computeren. Stikket passer kun på een måde.

2. Slut den anden ende af kablet til dit TV's UHF-antenneindgang.

3. Indstil nu dit TV på kanal 36 (ca.).

Tilslutning til lysnettet

1. Slut strømforsyningen til lysnettet.

2. Slut det firkantede stik fra strømforsyningen til indgangen bag på din **Plus/4**. Stikket passer kun på een måde.

3. På din strømforsyning sidder en sikkerhedsafbryder. Den skal være tændt, for at du kan arbejde.
   Når du ikke bruger din **Plus/4**, bør denne kontakt være slukket.

Nu kan du tænde din **Plus/4**. Kontakten sidder på højre side af computeren, når du sidder foran den.

Hvis det hele er korrekt tilsluttet, står der dette på din skærm:

COMMODORE BASIC 3.5 60671 BYTES FREE.

READY.

Den blinkende markør under READY fortæller dig, at **Plus/4** er klar til at modtage instruktioner. Baggrundsfarven er hvid, bogstaverne er sorte og kantfarven er lyseblå.

Det er nok nødvendigt at finindstille dit TV. Du kan eventuelt også justere kontrasten, farverne og lydstyrken, så du får den bedst mulige gengivelse.

Hvis du herefter endnu ikke har fået det rigtige billede frem på skærmen, kan du bruge nedenstående oversigt som hjælp til at finde fejlen.

Fejlsøgning

SYMPTOM	GRUND	AFHJÆLPES VED
Den røde kontrollampe på computeren lyser ikke.	Computeren er ikke tændt.	Kontroller at både kontakten på computeren og på strømforsyningen er tændt.
	Kabel fra strømforsyningen er ikke tilsluttet.	Kontroller strømindgangen m.h.t. løst eller forkert forbundet strømkabel.
	Strømforsyningen er ikke tilsluttet.	Kontroller forbindelsen til stikkontakten.
	En sikring i systemet er sprunget.	Henvend dig til din Commodoreforhandler og få sikringen skiftet.
Den røde lampe lyser, men der er intet billede.	TV er ikke justeret.	Kontroller finindstillingen på dit TV.
	Forkert tilslutning.	Tilsluttes på UHF.
	Antennekabel ikke tilsluttet.	Kontroller kablet til dit TV.
Mærkelige mønstre på TV-skærmen med programkapslen på plads.	Programkapslen er ikke indsat korrekt.	Afbryd strømmen og indsæt kapslen igen.
Billede uden farver.	TV dårligt indstillet.	Indstil TV.
Billede med dårlige farver.	Dårlig farvejustering på dit TV.	Juster farverne på dit TV.
Billede med susende baggrundsstøj.	For kraftig lyd på TV.	Sku ned!
Billede OK, men ingen lyd.	TV lyd sat for lav.	Skru op!
	Forstærkerforbindelse ikke i orden.	Efterse ledningen til forstærker-indgang.

VIGTIGT: Der findes enkelte ældre fjernsyn, der ikke kan vise hele Commodore **Plus/4**'s skærmbillede. Det viser sig ved, at man ikke kan se yderste venstre og yderste højre kolonne på skærmen. Vi vil anbefale brug af et andet fjernsyn eller en monitor.

Er dette ikke muligt, kan problemet løses ved at trykke ESC og R. Det reducerer skærmstørrelsen til 38 kolonner, så alle kolonner kan ses på skærmen. Du skal gentage dette, hver gang du tænder eller nulstiller din computer.

Brug af Skærm og Tastatur

* En tur rundt på tastaturet.
* Specielle taster.
* Grafik taster.
* Programmerbare funktionstaster.
* HELP-tasten.

En tur rundt på tastaturet

De fleste af tasterne på **Plus/4**'s tastatur er identiske med tasterne på en almindelig skrivemaskine, men hver enkelt tast kan gøre mere end blot lave et bogstav. I dette kapitel kan du lære, hvordan man bruger alle de ekstra ting, hver tast kan. Du kan også lære hvordan de specielle taster, som f.eks. <C=>-tasten og markør-tasterne, fungerer.

Mens du får beskrevet, hvordan de forskellige taster virker, bør du finde tasterne og bruge dem, så du ved præcis, hvordan de fungerer.

Specielle taster

RETURN-tasten

RETURN-tasten sidder yderst til højre, i tredie række på tastaturet.

Du skal trykke RETURN-tasten ned hver gang du har indtastet en linie med instruktioner til din **Plus/4**. Betragt denne tast som indgangstast, idet RETURN faktisk sørger for at informationer og instruktioner „går ind„ i computerens hukommelse.

SHIFT-tasterne

SHIFT- og SHIFT-LOCK-tasterne sidder på samme sted som på en skrivemaskine, altså i tredie og fjerde række i begge sider.

Denne tast virker på samme måde som skifte-tasten på en almindelig skrivemaskine. Din **Plus/4** har to SHIFT-taster og en SHIFT-LOCK, som virker på samme måde som skifte-låsen på en skrivemaskine.

Plus/4 har to tegnsæt: et med store bogstaver og grafik, og et med store og små bogstaver (ligesom en skrivemaskine). Når du trykker SHIFT ned sammen en tast i tegnsæt 1 (store bogstaver og grafik), får du det grafiske tegn, der sidder til højre, forrest på tasten. Gør du det samme i tegnsæt 2, får du det store bogstav, der svarer til tasten.

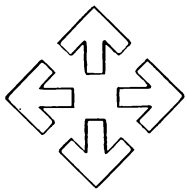
Plus/4 starter automatisk med tegnsæt 1, men i afsnittet om de grafiske tegn får du forklaret, hvordan du kan skifte mellem de to tegnsæt.

RUN/STOP-tasten
RUN/STOP-tasten sidder yderst til venstre i tredie række på tastaturet.

Denne tast skal du trykke på, hvis du ønsker at stoppe et program, der kører.
Hvis du trykker på denne tast, mens et program kører, er det igen dig, og ikke dit program, der har kontrollen over computeren.

Hvis du trykker SHIFT og RUN/STOP-tasterne ned samtidigt, henter og udfører **Plus/4** automatisk det første program på disketten i diskettestationen.

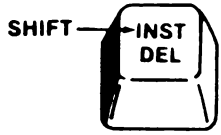
Markør-tasterne



Markør-tasterne er de piletaster, der sidder nederst i højre hjørne på tastaturet.

Det er let at flytte markøren hurtigt rundt på skærmen i alle retninger. Du skal blot trykke på den markør pil, der peger i den retning, du ønsker, markøren skal flytte sig i.
Som alle taster på **Plus/4** tastaturet, repeterer markør-tasterne automatisk, mens de holdes nede. Denne automatiske repetition gør, at markøren flytter sig, indtil du slipper tasten.
Bemærk: Du kan flytte markøren hen over alle tegn på skærmen, uden at disse ændres.

INST/DEL-tasten



INST-DEL-tasten sidder yderst til højre i øverste række på tastaturet.

Ved hjælp af denne tast kan du indsætte og slette tegn og tal i en linie. Når du trykker tasten ned alene, forsvinder karakteren umiddelbart til venstre for markøren, og markøren rykker en plads til venstre for at dække det "hul", der bliver.
Hvis du bruger markør-tasterne til at gå ind midt i en linie for at slette med DEL, vil du kunne se, at den del af linien, der er under og til højre for markøren, også rykker en plads til venstre.

Du kan også åbne plads til at indsætte tegn. Det gør du ved at trykke SHIFT og INST/DEL ned samtidigt, hvorved der åbnes plads til højre for markøren. Markøren selv bliver stående.
Når du indsætter i midten af en linie på denne måde, rykker resten af linien til højre.

INST/DEL-tasten kan spare dig for en masse arbejde, når du retter i det, du allerede har skrevet.

CLR/HOME-tasten

CLR/HOME-tasten sidder til højre i øverste række på tastaturet, lige ved siden af INST/DEL-tasten.

Denne tast tjener tre formål: hjem, sletskærm og slet vinduer. Når du trykker på denne tast, flyttes markøren straks op i øverste venstre hjørne. Dette kaldes markørens hjem. Resten af skærmen ændres ikke.
Hvis du trykker SHIFT og CLR/HOME-tasterne ned samtidigt, flyttes markøren igen hjem OG skærmen slettes. Det eneste det nu er tilbage på skærmen, er den blinkende markør i øverste venstre hjørne.
Hvis du trykker <CTRL>- og CLR-HOME-tasterne ned samtidigt, vil ethvert skærmvindue, du har defineret, blive slettet. Skærmvinduer er afgrænsede arbejdsområder på skærmen, men det kan du læse mere om senere i denne bog.

CONTROL-tasterne
CONTROL-tasterne sidder yderst til venstre og yderst til højre i anden række på tastaturet. De to taster virker på samme måde, men af praktiske grunde er der placeret en i hver side.

CONTROL-tasten virker altid kun sammen med en anden tast. Den virker ligesom SHIFT-tasten: Du trykker den ned og holder den nede, mens du trykker på den anden tast.

1. Ved at bruge CONTROL-tasten sammen med en af farvetasterne, kan du selv bestemme farven på din tekst. Afsnittet om farvetasterne forklarer nøjagtig, hvordan du bærer dig ad.

2. Du kan midlertidigt stoppe en LISTning eller et program der PRINTer på skærmen. Dette gør du ved at trykke på CONTROL og holde den nede, mens du trykker på S. For at starte LISTningen eller PRINTningen igen, trykker du en tilfældig tast ned. Vi foreslår, at du altid bruger mellemrumstasten (den aflange tast nederst på tastaturet).

3. CONTROL bruges også sammen med REV ON/REV OFF og FLASH ON/FLASH OFF tasterne.

Udover dette bruges CONTROL-tasten også af nogle af de programmer, du kan købe til din **Plus/4**.

<C=>-tasten
<C=>-tasten sidder yderst til venstre i fjerde række på tastaturet.

<C=>-tasten bruges ligesom CONTROL-tasten også sammen med andre taster.

1. Når du bruger SHIFT og <C=>-tasterne sammen, kan du skifte mellem de to tegnsæt på din **Plus/4** – store bogstaver/grafik eller store/små bogstaver.

2. Uanset hvilket tegnsæt du er i, fungerer <C=>-tasten som en slags skiftetaste til de grafiske tegn, der sidder til venstre på forsiden af den pågældende tast. Du skal blot holde <C=>-tasten nede, mens du trykker den tast ned, hvor det ønskede grafiktegn sidder.

3. <C=>-tasten kan ligesom CONTROL-tasten også bruges til at vælge farve på din tekst. Ved brug af <C=>-tasten i stedet for CONTROL-tasten, har du adgang til yderligere otte farver.

4. Hvis du ønsker at sænke farten på en program LISTning, skal du trykke <C=>-tasten ned alene. Når du gør dette, under en LISTning af et program, ruller programmet langsomt op over skærmen. Så snart du slipper tasten, ruller programmet op over skærmen med normal fart igen. Denne funktion er praktisk, idet det kan være svært at følge med, når programmet ruller hurtigt op over skærmen.

Farvetasterne

Farvetasterne er tallene fra 1 til 8. Hvis du kigger på forsiden af disse taster, står der to forskellige farver på hver tast.

Som før omtalt, har du mulighed for at ændre farve på bogstaver, tal og grafiske tegn. Du har 16 farver at vælge imellem. For at ændre farven gør du følgende:

Hvis du ønsker en af de otte farver, der står øverst på tasterne (f.eks. BLK=black=sort), skal du holde CONTROL-tasten nede, mens du trykker på den tast med den farve, du ønsker.

Hvis du ønsker en af de otte farver, der står nederst på tasterne (f.eks. ORNG=orange), skal du holde <C=>-tasten nede, mens du trykker på den ønskede farve.

Øv dig i at skifte farve, så du er sikker på hvordan det virker. Du vil opdage, at når du har skiftet farve, har alle bogstaver, tal og grafiske tegn den farve, du sidst har valgt.

Plus/4 har ialt 121 farver, men med <C=>- og CONTROL-tasterne kan du kun bruge de 16. Du får senere forklaret, hvordan du bruger alle de andre farver.

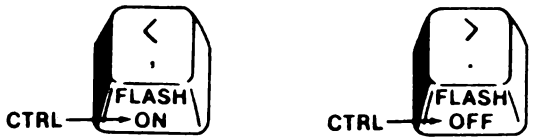
REVERSE ON og REVERSE OFF tasterne

RVS ON sidder på tasten med 9-tallet, og RVS OFF sidder på tasten med nullet.

Med **Plus/4** kan du også tegne negativt. Med andre ord: hvis du bruger sorte bogstaver på hvid baggrund, kan du få skrevet hvide bogstaver på sort baggrund.

Det eneste du behøver at gøre, er at trykke CONTROL og RVS ON tasterne ned samtidigt. Herefter bliver alt skrevet med negativ skrift, indtil du trykker CONTROL OG RVS OFF (samtidig), RETURN-tasten eller ESC og O, hvorefter du igen skriver med normal skrift.

FLASH ON og FLASH OFF tasterne



FLASH ON sidder på fjerde tast fra højre i fjerde række. FLASH OFF sidder på tasten til højre for FLASH ON.

Ligesom du kan få tegn skrevet negativt, kan du også få tegn til at blinke. Du skal blot trykke CONTROL og FLASH ON ned samtidigt og derefter vil alt, hvad du indtaster, blinke. Hvis du trykker CONTROL og FLASH OFF (samtidig), RETURN eller ESC, skriver du igen normalt.

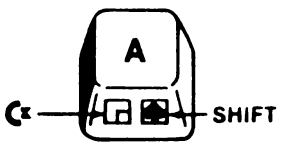
Grafik-tasterne

Som vi har nævnt før, er din **Plus/4** i tegnsættet med store bogstaver og grafik (tegnsæt 1), når du tænder den. Når du er i dette tegnsæt, kan du frit bruge de mere end 60 grafiske tegn, du kan se på forsiden af mange af tasterne.

Der sidder to grafiske tegn på hver grafiktast.

For at skrive det grafiske tegn til højre på tasten, skal du holde SHIFT-tasten nede, mens du trykker på grafiktasten.

For at skrive det grafiske tegn til venstre, skal du holde <C=>-tasten nede, mens du trykker på grafiktasten.



Ved at bruge de grafiske tegn kan du lave billeder, kort og alle mulige andre figurer, ved enten at skrive tegnene ved siden af hinanden og/eller oven på hinanden; nøjagtig ligesom hvis du brugte byggeklodser.

Prøv at bruge de grafiske tegn, så du kan se, hvordan de virker. I kapitel 7 får du endnu mere at vide om de grafiske tegn.

Du kan skifte mellem de to tegnsæt, store bogstaver/grafik og store og små bogstaver, ved at trykke SHIFT og <C>-tasterne ned samtidig. Ligegyldigt hvilket tegnsæt du er i, skal du altid skrive BASIC-instruktioner UDEN brug af SHIFT.

Når du er i tegnsæt 2, kan du skrive store og små bogstaver. Du kan ikke bruge grafiktegnene til venstre. Det gør du på samme måde som før: Du holder <C>-tasten nede, mens du trykker på den ønskede grafiktast. Grafiktegnene til venstre på tasterne er særligt velegnede til blokgrafik, rammer og andet man specielt bruger til budgetsimulering, lagerstyring etc.

Lad os lige se det hele skematisk:

	Tegnsæt 1	Tegnsæt 2
tast alene	store bogstaver	små bogstaver
SHIFT + tast	grafik til højre	store bogstaver
<C> + tast	grafik til venstre	grafik til venstre

Bemærk: når du skifter mellem de to tegnsæt, skifter tegnene på skærmen også. F.eks.: Skriver du et A i tegnsæt 1 og herefter skifter til tegnsæt 2, bliver det store A til et lille a. De eneste tegn, der ikke ændres ved skift mellem tegnsættene, er de grafiske tegn til venstre på tasterne.

ESC-tasten
ESC-tasten sidder til venstre i øverste række på tastaturet.

Med ESC-tasten kan du lave mange specielle rettefunktioner. Det er også denne tast du bruger til at definere skærmvinduer med. Som før nævnt, får du senere mere at vide om skærmvinduer. De rettefunktioner, du kan udføre med ESC-tasten, er f.eks. indsætning og sletning af hele linier, slet resten af linien o.s.v.

ESC-tasten bruges normalt sammen med et bogstav. For at aktivere funktionen, skal du først trykke ESC-tasten ned, slippe den igen og derefter trykke et af de følgende bogstaver ned:

Bogstav	Funktion
A	Automatisk indsætning.
B	Sæt nederste højre hjørne af skærmvinduet her (markørens nuværende position).
C	Annuller automatisk indsætning.
D	Slet linie.
I	Indsæt linie.
J	Flyt til start af linie.
K	Flyt til slut af linie.
L	Tænd for rulning.
M	Sluk for rulning.
N	Returner til normalt skærmbillede (40*25).
O	Annuller indsæt, quote, negative og blinke kontroller.
P	Slet alt fra starten af linien til markøren.
Q	Slet alt fra markøren og linien ud.
R	Reducer skærmbilledet til 38*23 tegn.
T	Sæt øverste venstre hjørne af skærmvinduet her.
V	Skærmbilledet ruller op.
W	Skærmbilledet ruller ned.
X	Annuller ESC-tasten.

Specielle symboler
På **Plus/4** tastaturet kan du også se en række specielle symboler, du normalt ikke finder på en skrivemaskine. Disse symboler er: nummertegnet (#), pundtegnet (£), pi (π), større end og mindre end tegnene (><), skarpe paranteser ([ ]), og pil op (↑). Disse symboler bliver brugt til at programmere **Plus/4**.

Bemærk: Har du en **Plus/4** med dansk tastatur, erstattes pundtegnet og de skarpe paranteser med æ, ø og å.

Programmerbare funktionstaster

De fire taster øverst på dit tastatur er specielle funktionstaster, som sparer tid for dig ved at udføre hele kommandoer ved tryk på een tast.

Du kan se, hvad hver enkelt tast udfører, hvis du taster KEY og trykker på RETURN-tasten, hvorefter skærmen skriver:

```
KEY 1, "SYS1525: 3-PLUS-1"
KEY 2, "DLOAD"+CHR$(34)
KEY 3, "DIRECTORY"+CHR$(13)
KEY 4, "SCNCLR"+CHR$(13)
KEY 5, "DSAVE"+CHR$(34)
KEY 6, "RUN"+CHR$(13)
KEY 7, "LIST"+CHR$(13)
KEY 8, "HELP"+CHR$(13)
```

Her er en forklaring på hver enkelt tast's funktion:

F1 aktiverer de indbyggede programmer i din **Plus/4**.

F2 skriver DLOAD" på skærmen, hvorefter du indtaster navnet på det program, du ønsker fra disketten, og trykker på RETURN-tasten.

F3 skriver en indholdsfortegnelse over de programmer, der ligger på disketten i diskettestationen, på skærmen.

F4 sletter skærmen (også i en af grafiktilstandene).

F5 skriver DSAVE" på skærmen, hvorefter du indtaster navnet på det program, du ønsker gemt på disketten, og trykker på RETURNtasten.

F6 udfører det program, du har liggende i computeren.

F7 skriver en listning af det program, du har liggende i computeren, på skærmen.

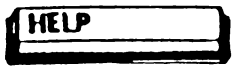
F8 HELP-tasten eller hjælpe-tasten får fejl i programinstruktioner til at blinke.

For at bruge disse funktioner, trykker du blot på den pågældende funktionstast. Du skal anvende SHIFT-tasten, hvis du vil bruge F4, F5, F6 eller F7.

Du kan til enhver tid omdefinere funktionstasterne, så de udfører det, du har brug for. Dette gør du ved at bruge KEY instruktionen. Du kan omdefinere funktionstasterne i et BASIC-program eller direkte på skærmen. Du kan omdefinere alle funktionstaster ligeså mange gange, du vil.

Bemærk: De omdefinerede funktioner forsvinder, så snart du slukker for computeren, og når du tænder den, er det igen ovenstående definitioner der gælder.

HELP- eller Hjælpetasten



Hjælpetasten er den funktionstast, der sidder længst til højre.

Når du laver fejl i et program, skriver **Plus/4** en fejlmeddelelse ud for at fortælle dig hvad du har lavet forkert. Disse fejlmeddelelser står yderligere forklaret i afsnittet om fejlmeddelelser i anden halvdel af denne brugervejledning.

Udover disse fejlmeddelelser, kan du få mere hjælp til at finde fejl ved at trykke på hjælpetasten. Når du trykker på hjælpetasten efter en fejl, skriver computeren den linie, der er fejl i og der hvor fejlen er, blinker tegnene. F.eks.:

?SYNTAX ERROR IN 10	**Plus/4** skriver, at der er syntaksfejl i linie 10.
HELP	Du trykker på hjælpetasten.
10 PRONT "COMMODORE COMPUTERS"	**Plus/4** blinker, der hvor fejlen er.

KAPITEL 3

Brug af programmel

* Introduktion
* Indbygget programmel
* PROGRAMMODULER
* Kassetter
* Disketter

Introduktion

Programudvalget til din **C 16-plus/4** vokser hastigt. Din forhandler kan holde dig a'jour med hensyn til nye produkter samt oplyse om eksisterende programmel.

Din Commodore **C 16-plus/4** kan anvende programmel på PROGRAMMODULER, KASSETTEBÅND og DISKETTER, som fås hos din Commodore forhandler. Du behøver blot at indsætte dem i din **C 16-plus/4** . Du kan også konstruere dine egne programmer og lagre dem på kassettebånd eller disketter.

Indbygget programmel i Plus/4

Plus/4's indbyggede programmer ligger i to kredse i computeren. Disse to kredse kan skiftes ud, således at der kan være andre indbyggede programmer i **Plus/4**.

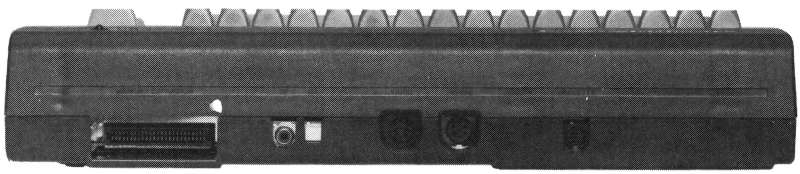
PROGRAMMODULER. Isætning af programmoduler

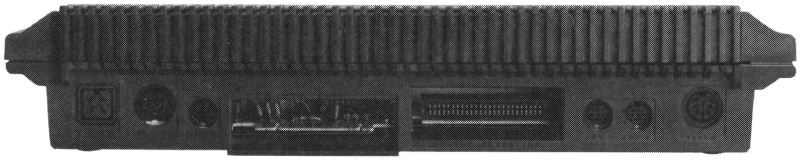
Commodore fremstiller et stort udvalg af programmoduler til din **C 16-plus/4**. Der er mange forskellige programmer til personlige, uddannelsesmæssige og forretningsmæssige formål såvel som spændende spil til din **C 16-plus/4**. Gør som følger, når du bruger programmoduler.

TRIN 1 SLUK for din **C 16-plus/4**.

VIGTIGT: DU SKAL SLUKKE FOR DIN COMPUTER, FØR DU ISÆTTER ELLER FJERNER PROGRAMMODULER. HVIS DU IKKE GØR DET, KAN DU BESKADIGE KASSETTEN ELLER COMPUTEREN.

TRIN 2 Fat modulet med etiketten OPAD og anbring den med et fast tryk i spalten på bagsiden af computeren.





TRIN 3 TÆND for din **C 16-plus/4**.

TRIN 4 Start spillet eller programmet efter de medfølgende instrukser. Et programmodul starter omgående, medens indbygget program starter, når man trykker på funktionstasten.

KASSETTER. Indlæsning af kassettebånd

Der findes et stort udvalg af programmer til **C 16-plus/4** i form af kassettebånd. Kassettebåndene er meget lig de musikkassetter, du afspiller på din båndoptager eller stereoanlæg. Computerbånd køres på Datasette båndoptageren, som fås hos din Commodore forhandler.

Du kan også bruge kassettebånd og Datasette til at lagre programmer, som du selv konstruerer. I det følgende afsnit forklarer vi, hvordan programmer lagres på bånd. Fremgangsmåden for indlæsning af bånd er den

samme, uanset om der anvendes færdigt programmel eller programmer, du selv har lagret.

TRIN 1 Sæt kassetten i din Datasette og luk klappen.

TRIN 2 Spol båndet tilbage til start ved at trykke på Datasettens REWIND-tast (tilbagespolingstast)

TRIN 3 Når båndet er spolet tilbage til start, skriver du ordet LOAD og RETURN-tasten trykkes ned. Computeren reagerer med følgende meddelelse:

PRESS PLAY ON TAPE

TRIN 4 Tryk PLAY-tasten på Datasetten. Skærmen bliver tom, når Datasetten starter. Når et program er fundet, vil skærmen vise følgende:

FOUND "programnavn"

TRIN 5 Tryk på Commodore tasten for indlæsning af det FUNDNE program. Hvis der er mere end et program på båndet og det program, **C 16-plus/4** har fundet, ikke er det, du ønsker, trykker du på mellemrumstasten for fortsat søgefunktion.

Når programmet er indlæst, kommer ordet READY frem på skærmen. Hvis du ønsker at afbryde indlæsningen, før den er færdig, trykker du på RUN/STOP tasten. Når programmet er indlæst, indtaster du ordet RUN for at starte programmet. Du kan også "LISTE" programmet eller ændre det, hvis det er et BASIC program.

BEMÆRK: Ved indlæsning af et specifikt program på båndet anvendes LOAD "programnavn".

Lagring af programmer på kassettebånd

Når du konstruerer et program og ønsker at lagre det på kassettebånd, gør du følgende:

TRIN 1 Indtast:
SAVE "programnavn"

Det anvendte programnavn kan være hvad som helst, du ønsker, men ikke mere end ialt 16 tegn (bogstaver og/eller tal).

TRIN 2 Tryk RETURN tasten. Computeren viser følgende:

PRESS RECORD AND PLAY ON TAPE

TRIN 3 Tryk RECORD og PLAY tasterne på din Datasette. Skærmen bliver tom. Når programmet er lagret, fremkommer ordet READY på skærmen.

Eksempler på SAVE kommandoer til kassettebånd:

SAVE "MITJOB" Dette navn er det specifikke navn på det program, der lagres.

SAVE "3 TEST"

BEMÆRK: Når et program lagres på kassettebånd, skal man altid holde øje med hvor på båndet, man befinder sig. Især skal man passe på ikke at lagre et program helt fremme ved båndets begyndelse, da mange bånd har en magnetisk førestrimmel, hvorpå man ikke kan lagre informationer. En del af programmet ville derfor ikke blive lagret.

Ved INDLÆSNING eller LAGRING af et program skal man, hvis man ønsker at afbryde processen, før den er færdig, først trykke RUN/STOP tasten. Når man har trykket RUN/STOP på tastaturet, trykkes stoptasten på Datasetten.

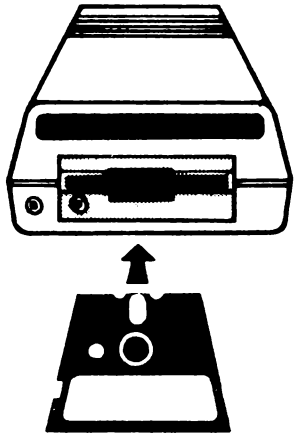
DISKETTER. Indlæsning af programmer fra Diskette

Disketter er hurtige og nemme at bruge. Vær forsigtig med behandlingen af dine disketter og diskettestation.

Fremgangsmåden er den samme for indlæsning af alle slags disketter.:

TRIN 1 Kontroller at diskettestationen er TÆNDT.

TRIN 2 Indsæt disketten i diskettestationen. Den side af disketten hvor etiketten er anbragt, skal vende opad. Indsæt disketten i åbningen, således at den ende, hvor etiketten sidder, vender udad. Find det lille hak på siden af disketten (det er muligvis dækket af en påklæbet mærkeseddel). Dette hak skal vende til venstre, når disketten isættes, hvis diskettestationen ses forfra. Kontroller at disketten er helt på plads.



TRIN 3 Luk beskyttelsesklappen på diskettestationen når du har isat disketten.

TRIN 4 Indtast:

DLOAD "programnavn"
Navnet på det program,
der skal indlæses

For at spare tid kan du trykke FUNKTIONSTAST 2 og indtaste programmets navn og det sidste anførselstegn.

TRIN 5 Tryk RETURN tasten. Disketten snurrer og skærmen viser

SEARCHING FOR PROGRAMNAVN

LOADING

READY

(SØGER EFTER PROGRAMNAVN
INDLÆSER
KLAR)

TRIN 6 Dit program er klar til brug. Indtast nu RUN og tryk RETURN tasten for start af programmet.

Hvis det røde lys på diskettestationen blinker efter at DLOAD er afsluttet, er der noget galt. Indtast:

?DS$ (Og tryk RETURN)
for at finde ud af, hvad der er i vejen.

Eksempler på DLOAD kommandoer:

DLOAD"*"	INDLÆSER det 1. program på disketten.
DLOAD"MINFIL"	INDLÆSER et disketteprogram.
DLOAD"SET*"	INDLÆSER det 1. program på disketten, der begynder med bogstaverne SET.

Formattering af en diskette

Formattering gør en ny TOM diskette klar til brug. Enhver tom diskette skal formatteres, før den kan bruges. Dette sker ved anvendelse af HEADER kommandoen.

VIGTIGT: BRUG ALDRIG HEADER KOMMANDO PÅ EN DISKETTE, DER INDEHOLDER NOGET, MEDMINDRE DU ØNSKER AT SLETTE HELE INDHOLDET. FORMATTERING SLETTER ALT, HVAD EN DISKETTE INDEHOLDER.

Formatet for HEADER kommandoen er:

HEADER "diskettenavn", Uenhedsnr., Iid., Ddrev

---

* Man skal bruge navnet på hele disketten. Man kan give disketten et hvilket som helst navn med indtil 16 karakterer.

* Enhedsnummer angiver den eksterne enhed, din computer er tilsluttet (diskettestation eller datasette), og er normalt nr. 8.

* Id. er bogstavet I og to vilkårlige bogstaver og/eller tal, f.eks. I21, IR 5 o.s.v. Man kan give disketten et hvilket som helst id., man ønsker, men for at undgå forveksling, bør man give disketterne forskellige id.

* Hvis man har en dobbelt diskettestation, tilføjes D0 eller D1 for valg af drev 0 eller drev 1.

ARE YOU SURE?

Så snart du trykker RETURN efter at have indtastet header kommandoen, spørger **C 16-plus/4** ARE YOU SURE? (er du sikker?). Dette sker for at give dig en sidste chance for at skifte mening.

Til formattering af disketten indtastes YES eller Y og RETURN tasten trykkes. Hvis man beslutter ikke at formattere disketten indtastes NO eller N og RETURN tasten trykkes.

Her følger nogle eksempler på HEADER kommandoer:

```
HEADER "BREVE",U8,I07, D0
HEADER "ØKONOMI",U8,IS3,D0
```

Nu da du ved, hvordan man formatterer en diskette, er du klar til at anvende disketten til konstruktion og lagring af programmer på din **C 16-plus/4**. Første del af **C 16-plus/4** encyklopædien indeholder yderligere oplysninger om HEADER kommandoen.

Lagring af programmer på diskette

Dersom du ønsker at genbruge et program, du har konstrueret, må du sørge for at LAGRE det, før du indlæser et andet program eller slukker for **C 16-plus/4**. Hvis du glemmer det, mister du programmet.

Hvis du ændrer på et lagret program, må du LAGRE det igen, hvis du vil beholde den nye version.

Når du lagrer et program igen, erstatter du den gamle version med en ny. Hvis du vil beholde både den gamle og nye version, skal du give den nye en anden titel, når du lagrer den.

Benyt følgende fremgangsmåde, når du lagrer et program på diskette:

TRIN 1 Indtast DSAVE "programnavn"

TRIN 2 Tryk RETURN tasten. Når programmet er lagret, viser computeren følgende:

SAVING "programnavn"
READY

Eksempel:

DSAVE "MITPROGRAM"	Programnavn kan indeholde indtil 16 karakterer.

Hvis det røde lys på diskettestationen blinker efter at DSAVE-funktionen er afsluttet, er der noget galt. Indtast:

?DS$ (tryk RETURN tasten)
for at finde ud af, hvad der er i vejen.

Directory kommandoen

Når du LAGRER programmer på diskette fører computeren en liste over alle de filer og programmer, der er lagret på disketten. Man kan få en visning i form af en indholdsfortegnelse, så man kan se, hvad der er på en diskette, ved hjælp af directory kommandoen:

Indtast: DIRECTORY, tryk derefter RETURN tasten (eller tryk FUNKTIONSTAST 3).

Så snart du trykker RETURN tasten, viser din **C 16-plus/4** alt, hvad der er på disketten. Man kan også nøjes med en visning af blot en del af indholdsfortegnelsen:

DIRECTORY "MI*" <RETURN>	Lister enhver fil på disketten, der begynder med bogstaverne MI.
DIRECTORY "*=prg"	Lister alle programfiler på en diskette.
DIRECTORY "*=seq"	Lister alle sekventielle filer på en diskette.
DIRECTORY "MI*=prg"	Lister alle programmer, der begynder med bogstaverne MI, fra en diskette.

DIRECTORY "MI*=seq"

Lister alle sekventielle filer, der begynder med bogstaverne MI, fra en diskette.

KAPITEL 4

Starten

* Tastaturfarver
* Farve og negativ skrift
* Nogle enkle programmer
* Rettelse af skrivefejl
* Introduktion til **C 16-plus/4** skærmen
* Mere om UDSKRIVNING på skærmen
* Skærmvinduer

Formålet med dette kapitel er at gøre dig bekendt med nogle af **C 16-plus/ 4**'s karakteristika og muligheder, samt at sætte dig i stand til at tage de første skridt til programmering med din computer.

Tastatur – farver

Man kan ændre på karakterernes farve på skærmen for at forbedre læseligheden eller for at finde en farvekombination, man kan lide. Dersom man vil se, hvordan de forskellige farver ser ud på skærmen, gør man følgende:

TRIN 1 Hold CONTROL tasten nede

TRIN 2 Tryk tasten 6 medens CONTROL holdes nede. Markøren bliver grøn.

TRIN 3 Slip CONTROL tasten og indtast nogle bogstaver. Alt hvad der indtastes vil nu fremstå grønt.

HOLD CONTROL SAMMEN MED FARVETAST	FARVE RESULTAT
1	SORT
2	HVID
3	RØD
4	CYAN (turkis)
5	VIOLET
6	GRØN
7	BLÅ
8	GUL

Brug af CONTROL tasten sammen med taltasterne 1 til og med 8 giver mulighed for at vælge de farver, der er vist øverst på hver farvetast.

Nu trykkes så <C=>-tasten. Ved brug af tasterne mellem 1 og 8 ændrer markøren en af de 8 farver, der er trykt nederst på hver farvetast. Alle 16 farver kan vises på skærmen samtidig.

HOLD <CTRL> SAMMEN MED FARVETAST	FARVE RESULTAT
1	ORANGE
2	BRUN
3	GULGRØN
4	LYSERØD
5	BLÅGRØN
6	LYSEBLÅ
7	MØRKEBLÅ
8	LYSEGRØN

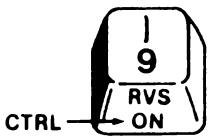
Farve og udskrivning i negativ

Din **C 16-plus/4** kan vise tal, bogstaver og grafiske symboler i 16 forskellige farver. Man kan også få disse karakterer vist i negativ med primærfarven (markør) og baggrundsfarven byttet om.

TRIN 1 Slet skærmbilledet ved tryk på SHIFT og CLR/HOME.

TRIN 2 Hold CONTROL tasten nede og tryk RVS ON tasten:

KEY



TRIN 3 Slip tasterne og hold mellemrumstasten nede (den lange tast nederst på tastaturet).

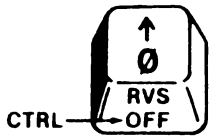
TRIN 4 Hold mellemrumstasten nede så længe, du ønsker. Medens du holder den nede, skulle en linie i samme farve som bogstaverne på skærmen forlænges. Hvis linien går til slutningen af rækken, fortsætter den på næste række.

TRIN 5 Slip mellemrumstangenten (men tryk ikke RETURN tasten).

TRIN 6 Hold CONTROL tasten nede og tryk en af farvetasterne (ikke en farve, der allerede er på skærmen). Så snart du gør dette, vil markøren antage den farve, der svarer til den nedtrykkede tast.

TRIN 7 Hold mellemrumstangenten nede igen. Nu tegner din **C 16-plus/4** en linie i den nye farve. Fortsæt med at ændre farverne med CONTROL eller <C=>-tasten og farvetasterne. Hold derefter mellemrumstasten nede for at lave forskelligt farvede linier.

TRIN 8 Afbryd udskrivning i negativ ved at holde CONTROL tasten nede og trykke RVS OFF tasten. Tryk på RETURN tasten afbryder også udskrivning i negativ.



Prøv at indtaste nogle bogstaver i negativ. Hold blot CONTROL og RVS ON tasten nede for at skifte til negativ og indtast så, hvad du har lyst til. Bogstaver i negativ egner sig glimrende til overskrifter. De kan også bruges til at fremhæve vigtige ord og tal. Prøv følgende:

```
          ┌TRYK CONTROL OG RVS ON
          ▽
PRINT "■COMMODORE C 16-PLUS/4■"
          └TRYK CONTROL OG RVS OFF
```

Prøv nu samme linie, idet reverse on og off erstattes af FLASH ON og OFF:

```
          ┌TRYK CONTROL OG FLASH ON
          ▽
PRINT "▮COMMODORE C 16-PLUS/4▬"
          └TRYK CONTROL OG FLASH OFF
```

Begge funktioner kan også bruges på **C 16-plus/4** som en del af en programinstruktion.

Nogle enkle plus/4 programmer

Indtast dette program præcis som det står her. Udelad ikke tallene ved begyndelsen af linien, da de er de linienumre, der angiver den rækkefølge, i hvilken din computer udfører programlinierne. Glem ikke at trykke RETURN tasten ved afslutningen af hver enkelt linie, du indtaster.

10 PRINT "PLUS/4"	Denne linie giver computeren ordre til at UDSKRIVE **plus/4** på skærmen.
20 GOTO 10	Denne linie giver din computer ordre til at vende tilbage til linie 10 og udskrive plus/4 igen.
RUN	Dette er en ordre til computeren om at udføre hvad de 2 linier giver den ordre til.

Tryk RUN/STOP tasten for at stoppe programmet.
Hvorfor skrev din **C 16-plus/4** sit navn så mange gange? GOTO giver computeren ordre til at gå tilbage til linie 10 og udskrive plus/4 igen og igen. Denne gentagelsesfunktion kaldes et loop (løkke).

Indtast nu følgende:

NEW

og tryk på

RETURN TASTEN	Her får computeren ordre til at glemme det sidste program og gøre klar til et nyt.
Computeren svarer READY	Du indtaster ikke dette ord, computeren viser det for at fortælle, at den er KLAR til et nyt program.
10 PRINT "PLUS/4"	Samme linie 10 som sidste gang. PRINT giver din **C 16-plus/4** ordre til at udskrive alt, hvad der står mellem anførselstegnene.

20 COLOR 0,12 — Denne ordre får computeren til at ændre farven på skærmen.

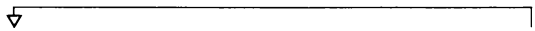
RUN

Denne gang er der ingen GOTO løkke i programmet, så dine ordrer udføres kun en gang.

Rettelse af indtastningsfejl

Hvis du laver fejl, når du taster noget ind, er der flere muligheder for at foretage ændringer.

1. Du kan når som helst TASTE EN LINIE OM, selv efter at du har kørt programmet. **C 16-plus/4** erstatter automatisk den gamle linie med den nye, når du trykker RETURN tasten, efter at du har indtastet den nye linie. Den gamle linie står stadig på skærmen, men **C 16-plus/4** ignorerer den. Hvis der er meddelelser med samme linienummer, bruger din **C 16-plus/4** kun det sidst indtastede. Hvis der f.eks. forekommer en fejl i et kort program med brug af COLOR kommandoen til ændring af baggrundsfarven på skærmen:



10 COKOR 0,3 — fejl
20 PRINT "PLUS/4"

Tryk RETURN tasten for at komme til en ny linie, hvorefter du indtaster linie 10 igen, men korrekt.

10 COLOR 0,3 — RETURN

Nu udskiftes den første linie 10 med den anden linie 10. Du kan kontrollere dette ved at indtaste LIST, hvorved dit program vises i en opstilling linie for linie, således som det er lagret i computerens hukommelse. Når du LISTER et program, fremkommer alle linier i rigtig rækkefølge og de gamle linier vises ikke:

LIST — RETURN

På skærmen står:

10 COLOR 0,3
20 PRINT "PLUS/4"

At udskifte linier i et program er også en god måde at eksperimentere med din computer på. Når du udskifter en linie behøver den nye ikke at have nogen som helst lighed med den gamle. I stedet for at rette stavningen i COLOR kan du f.eks. indtaste følgende:

10 PRINT "TI ER LIG MED SEKS" mellemrum RETURN

Kør nu programmet og se, hvad der sker.

2. DU KAN SLETTE EN UØNSKET LINIE blot ved at indtaste liniens nummer og trykke RETURN tasten. Computeren ignorerer linien, selv om denne stadig står på skærmen. Indtast LIST for at få programlistningen, så du kan sikre dig, at linien er forsvundet fra programmet.

```
10 PRINT "TI ER LIG MED SEKS"          RETURN
20 PRINT "PLUS/4"                      RETURN
10                                     RETURN
LIST                                   RETURN
20 PRINT "PLUS/4"
```

3. DU KAN REDIGERE EN LINIE. Brug markør tasterne for at komme til det sted på linien, hvor du vil foretage ændring. Indtast nu blot hen over det, du ønsker at ændre. Tryk RETURN når du er færdig. BEMÆRK: Når du arbejder med nummererede programlinier, behøver du ikke være ved slutningen af linien, før du trykker RETURN. Din **C 16-plus/4** husker hele linien, selv om du trykker RETURN midt på linien.

10 PRINT "KLOKKEN ER ET"

Hvis du ønsker at ændre tiden til TO, flyttes cursoren til E'et i ET.

10 PRINT "KLOKKEN ER ET"

Indtast nu blot TO over ET og tryk RETURN.

10 PRINT "KLOKKEN ER TO" RETURN

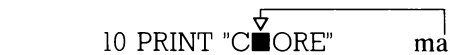
BEMÆRK: Når du indtaster et anførselstegn efter en PRINT instruks, er du i QUOTE MODE. I quote mode fungerer nogle af tasterne anderledes. Hvis du f.eks. trykker cursor-downpilen (markør ned), medens du er i quote mode, vil markøren ikke flytte sig og du vil se et Q i negativ skrift på skærmen. Når du kører PRINT instruksen, bliver Q'et i negativ skrift

ikke udskrevet; i stedet flytter markøren sig nedad. I quote mode formoder computeren at alt, hvad du indtaster, er noget du ønsker at vise eller foretage senere, når du kører PRINT instruksen.

4. DU KAN LAVE MELLEMRUM I ET ORD ELLER EN LINIE med INST tasten (du får denne indsætningstast ved at holde SHIFT-tasten nede, medens du trykker INST/DEL). Hold tasterne nede, indtil du har lavet de mellemrum, du behøver. (Bemærk at markøren bliver stående på samme sted, medens mellemrummene fremkommer til højre for den). Indtast derefter det, du ønsker at indsætte.

10 PRINT "CORE" RETURN

For at ændre dette til COMMODORE, flyttes markøren til O i CORE og SHIFT og INST tasterne trykkes, indtil der er tilstrækkeligt mellemrum. Det er ikke nødvendigt at tælle mellemrummene. Man kan blot skønne og så lave flere, hvis der ikke er nok.



Tilføj nu de øvrige bogstaver:

10 PRINT "COMMODORE" RETURN

5. MAN KAN SLETTE KARAKTERER OG LUKKE MELLEMRUM med DEL tasten (man får denne slettetast ved at trykke INST/DEL). Denne tast sletter karakterer eller mellemrum umiddelbart til VENSTRE for markøren.

10 PRINT "EFTERMIDDAGSPLAN" RETURN

Man kan ændre dette til UGEPLAN ved at bevæge markøren til A i EFTERMIDDAG trykke INST/DEL tasten ni gange og indtaste UGE.

10 PRINT "EFTERMIDDAGSPLAN" Indtast og erstat AGS
og tryk INST/DEL ni gange. med UGE og tryk RE-
10 PRINT " AGSPLAN" TURN

Et lidt længere program

Nu da du har eksperimenteret lidt med din **C 16-plus/4**, kommer vi til et program, du kan prøve, som det vil tage lidt længere tid at indtaste. (En-

hver øvet programmør kan fortælle dig, at om ikke andet er programmering en glimrende måde at opnå bedre rutine i maskinskrivning på.

Slet først det, der står på skærmen ved at holde SHIFT tasten nede, medens du trykker CLR/HOME tasten. Dette sletter, hvad der står på skærmen. Slet derefter gamle programmer fra **C 16-plus/4** hukommelse ved at indtaste NEW og trykke RETURN.

Indtast det følgende program nøjagtigt, som det står. Husk at indtaste linienumre og husk al tegnsætning. Gør brug af oplysningerne om fejlretning, hvis du indtaster noget forkert. Glem ikke at trykke RETURN efter hver enkelt linie.

BEMÆRK: Husk, du kan stoppe et program ved at trykke RUN/STOP tasten.

```
NEW

10 COLOR 1,8

20 PRINT "DER SKETE NOGET MÆRKELIGT ";   Sørg for at lave et mellemrum her

30 COLOR 1,3                              Sørg for at lave et mellemrum her

40 PRINT "PÅ VEJ TIL TASTATURET ";

50 COLOR 1,7                              man får hjerterne frem ved at holde SHIFT

60 PRINT "♥♥♥♥♥♥";                        nede, medens S-tasten trykkes 6 gange

70 GOTO 60

RUN
```

Efter at du har stoppet programmet (ved at trykke RUN/STOP tasten) prøver du nu at indtaste LIST. Når programmet kommer på skærmen, så husk oplysningerne om fejlretning og forsøg at ændre dette program, så det siger noget mere dybsindigt.

TIP: Kunne du tænke dig at sænke hastigheden på dette program uden at stoppe det? Så hold blot <CTRL>-tasten nede medens programmet kører.

C 16-Plus/4 tekst skærmen

Prøv at indtaste dette program. (Glem ikke at trykke RETURN efter indtastning af hver linie).

```
NEW

10 PRINT "♥";                    tryk SHIFT og S tasten

20 GOTO 10

RUN
```

Nu fyldes skærmen med hjerter. Når hele skærmen er dækket med hjerter, trykkes RUN/STOP tasten for at afslutte programmet. Dette program viser dig, hvor stor din **C 16-plus/4** skærm er.

Indtast nu dette program:

```
NEW

10 PRINT "♥";                    tryk SHIFT og CLR/
                                 HOME

20 FOR X = 1 TO 40

30 PRINT "♥";                    tryk SHIFT og S

40 NEXT X

RUN
```

Når du kører dette program, fyldes den første række på din skærm af hjerter. Der er 40 hjerter ialt. Eftersom rækken er fuld, kan du se, at der er 40 positioner i skærmens bredde. Hver position i skærmens bredde kaldes en kolonne.

Indtast nu dette program:

```
NEW
10 PRINT "♥"                     tryk SHIFT og CLR/
                                 HOME
```

```
20 FOR X = 1 TO 25

30 PRINT "♦"                          tryk SHIFT og Z

40 NEXT X

RUN
```

Når du kører dette program, fyldes den første kolonne på din skærm af ruder. Der er skrevet 25 ruder, men de første tre forsvinder øverst på skærmen, fordi ordet READY omgivet af to tomme linier altid kommer til syne ved slutningen af programmet. Der er så 25 rækker. En smule logik siger dig, at din **C 16-plus/4** har 40 kolonner og 25 rækker. **C 16-plus/4** har 1000 forskellige positioner på skærmen til bogstaver, tal, grafiske symboler o.s.v.

BEMÆRK: Somme tider indtaster du en særlig lang linie på din **C 16-plus/4** som f.eks. denne:

```
10 PRINT "JEG KAN GODT LIDE, AT DU RØRER VED MIT TASTATUR. KOMMER DU HER TIT?"
```

(Det er da en temmelig lang linie, ikke? – mere end 50 karakterer!)

Når du indtaster dette, vil du bemærke, at der ikke er plads i en række, men fortsæt bare med at indtaste; **C 16-plus/4** går automatisk videre til næste række og fortsætter med at udskrive der, indtil din linie er færdig. Du kan indtaste op til 80 karakterer på en programlinie (indtil to hele rækker).

Forsøg nu at KØRE dette en-linies program. Meddelelsen er indtastet på to rækker. Hvis din linie er længere end en række, lader **C 16-plus/4** den fortsætte til næste række. **C 16-plus/4** opfatter linien som færdig, når du trykker RETURN tasten, ikke når du indtaster frem til slutningen af rækken. Det vil du vænne dig til efterhånden, som du bruger din **C 16-plus/4**.

Indtast nu dette program:

```
NEW

10 PRINT " ♥ ";                       lav et mellemrum på
                                      hver side af hjertet tryk
                                      SHIFT og S tasten
```

```
20 GOTO 10
RUN
```

Når du kører dette program, vil du se, at det er muligt at meddele **C 16-plus/4** præcis, hvor den skal udskrive noget på skærmen.

Mere om udskrift på skærmen

Prøv at indtaste dette program:

```
NEW
10 PRINT "A", "B"
20 PRINT "A"; "B"
RUN
```

Skærmen vil se således ud:

```
A       B                                   linie 10 udskrev dette

AB                                          linie 20 udskrev dette
```

Når linie 10 og linie 20 er næsten identiske, hvorfor er der så en sådan forskel på, hvad de UDSKRIVER på skærmen? Forskellen ligger i den tegnsætning, der er brugt mellem de enkelte elementer, dette program udskriver.

Hvis man anvender komma til adskillelse af elementer i en PRINT instruks, udskrives elementerne med flere mellemrums indbyrdes afstand. Hvis man anvender semikolon, udskrives elementerne lige ved siden af hinanden.

Som du vil erindre, har **C 16-plus/4** skærmen 40 kolonner i bredden. Disse kolonner opdeles i fire afsnit hver med en bredde svarende til 10 mellemrum, der benævnes PRINTZONER. Hvis man anvender komma til adskillelse af udskrevne elementer, udskriver **C 16-plus/4** det første element i den første printzone, o.s.v. Kommaerne fungerer som tabulatorer på en skrivemaskine.

```
PRINTZONE   PRINTZONE   PRINTZONE   PRINTZONE
1           2           3           4
          1 1111111112  2222222223  3333333334
1234567890  1234567890  1234567890  1234567890
A           B
```

Hvis man forsøger at UDSKRIVE mere end fire elementer adskilt med kommaer, går **C 16-plus/4** automatisk til næste linie for at skrive. F.eks.:

```
PRINT "A", "B", "C", "D", "E", "F"
```

adskilles bogstaverne på skærmen således:

```
          1       11      21      31    KOLONNE
RÆKKE 1   A       B       C       D
RÆKKE 2   E       F
```

Hvis man anvender semikolon til adskillelse af de enkelte elementer i en PRINT instruks, ignorerer **C 16-plus/4** printzonerne og udskriver alle ele menterne efter hinanden:

```
PRINT "A"; "B"; "C"; "D"; "E"; "F"
```

Udskrives således:

```
ABCDEF
```

Dette er, hvad der sker, hvis det første element, du skriver, er 12 bogstaver langt og det andet element adskilles derfra ved hjælp af komma:

```
PRINT "ABCDEFGHIJKL", "M"
```

Udskrives således:

```
ABCDEFGHIJKL                M
printzone1      printzone 2     printzone 3
```

Slet nu skærmbilledet og indtast dette program:

```
NEW
10 PRINT 1,2
20 PRINT 1;2
RUN
1          2
1  2
```

Dette program viser dig to nye ting:

1. Tal behøver ikke at sættes i anførselstegn i en PRINT instruks.
2. Tal vises med et mellemrum på begge sider, så hvis man anvender semi-

kolon, udskrives tallene ikke lige ved siden af hinanden, således som det er tilfældet med bogstaver. Mellemrummet giver plads til et minustegn foran negative tal, når det er nødvendigt.

Skærmvinduer

Vinduer gør det muligt at afgrænse et bestemt areal på skærmen som arbejdsområde. Alt hvad man indtaster (linier, listning af programmer o.s.v.) vil, efter at man har sat et vindue, fremkomme inden for rammerne af vinduet uden at berøre den del af skærmen, der ligger uden for vinduet. Man kan opsætte et vindue hvor som helst på skærmen.

Man sætter et vindue på følgende måde:

1. Markøren flyttes til den position, hvor man ønsker øverste venstre hjørne af vinduet anbragt.
2. ESCape tasten og derefter bogstavet T, nedtrykkes.
3. Markøren flyttes til den position, hvor man ønsker nederste højre hjørne af vinduet anbragt.
4. ESCape trykkes efterfulgt af tasten B. Nu er vinduet sat.

Alle uddata på skærmen er begrænset til den "firkant", man har afgrænset. Hvis vinduet ønskes fjernet, trykkes home-tasten to gange. Dette sletter vinduet og markøren flyttes til en position i skærmens øverste venstre hjørne.

Kapitel 5

Tal
og beregninger

* Tal og grundoperatorer
* Om at udføre beregninger
* Brug af variabler
* Direkte tilstand
* Numeriske funktioner
* Tilfældige tal og andre funktioner

Tal og grundoperatorer

Man kan bruge sin **C 16-plus/4** som en almindelig regnemaskine. Foruden standardoperatorerne + og − bruger **C 16-plus/4** tegnet * for multiplikation og tegnet / for division og brøker (computere bruger tegnet * i stedet for × for multiplikation, fordi en computer ikke kan kende forskel på bogstavet x og det matematiske symbol ×). Man kan bruge disse operatorer og tal i direkte tilstand (ingen linienumre) eller i et program. Hverken tal eller operatorer bør være i anførselstegn, for at **C 16-plus/4** kan udføre matematiske funktioner.

MATEMATISKE GRUNDOPERATORER		RELATIONELLE GRUNDOPERATORER	
Addition	+	Større end	>
Subtraktion	−	Mindre end	<
Division og brøker	/	Lig med	=
Multiplikation	*	Større end eller lig med	=>
Eksponentiering	↑	Mindre end eller lig med	<=
		Ikke lig med <> eller	><

BEMÆRK: Din **C 16-plus/4** accepterer ikke komma som en del af et tal. Du skal f.eks. indtaste 109.401 i stedet for 109,401. Hvis du sætter komma i et tal, opfatter **C 16-plus/4** det som om du indtastede to tal (adskilt af kommaet), derfor ville din **C 16-plus/4** læse 109 og 401 i stedet for 109,409.

Brøker og decimaler

Man kan indtaste en brøk således: .5
eller således: 1/2

Din **C 16-plus/4** udfører faktisk divisionen.

Hvis man indtaster en brøk i en PRINT instruks, bliver svaret altid i form af en decimalbrøk eller et helt tal. F.eks.:

PRINT 139/493+5 RETURN
5.28194726

Her følger nu et eksempel med anvendelse af pi (3.14159256), der repræsenterer forholdet mellem en cirkels omkreds og dens diameter. Denne størrelse kan benyttes blot ved tryk på π-tasten:

PRINT π/374 RETURN
8.39998036E-03

Eksponentiel notation

Hvad mente **C 16-plus/4** med E-03, som var en del af ovenstående svar? Din **C 16-plus/4** viser decimaltal inden for området −999999999 til 999999999 i standardtal. Tal der ligger over dette område, (d.v.s. med mere end ni cifre), vises automatisk i eksponentiel notation. Du kan selv indtaste tal i denne form og din **C 16-plus/4** vil læse dem uden besvær (afgjort med mindre besvær, end du havde med at omskrive dem). Eksponentiel notation er tit nyttigt, da din **C 16-plus/4** vil derved bliver i stand til at vise store tal med færre cifre.

Tallet 198505478 indtastet i eksponentiel notation vil se således ud:

```
                    Der vises kun ET ciffer
                    til venstre for decimalkommaet
1.98505478E+8
            ^-------|Dette tal er det antal pladser
                    |kommaet flyttes.
```

Ved et tal mindre end en med flere decimaler, vil det andet tal blive et − i stedet for et +, hvilket angiver, at kommaet flyttes til højre. F.EKS.:

.0003359 = 3.359E−4

Andre eksempler:

20 = 2E+1	kommaet flyttes en plads til venstre
105000 = 1.05E+5	kommaet flyttes fem pladser til venstre
.0666 = 6.66E−2	kommaet flyttes 2 pladser til højre

Udførelse af beregninger

For at udføre en beregning indtastes PRINT efterfulgt af regnestykket. Husk ikke at sætte regnestykket i anførselstegn.

Skriv følgende program:

```
NEW
10 PRINT 1+2,2−1
20 PRINT 2*2,4/2                    |brug skråstregen på
           ^------------------------|?tasten
RUN
3        1
4        2
```

For første gang udskrev PRINT ikke nøjagtigt, hvad du indtastede i instruksen. Din **C 16-plus/4** løste i stedet regnestykkerne og UDSKREV facit. Det eneste du behøver at gøre for at bruge PRINT til beregninger, er at udelade anførselstegnene. Prøv nu følgende:

```
NEW
10 PRINT "2001/2010"
20 PRINT 2*3
RUN
2001/2010
 6
```

der er her et mellemrum til facits fortegn

Da regnestykket i linie 10 står i anførselstegn, udskriver din **C 16-plus/4** regnestykket, som om det var en almindelig tekst: nøjagtigt som det står mellem anførselstegnene. Stykket er ikke regnet og der er intet mellemrum til tallets fortegn.

Nu bevæger du markøren tilbage til linie 10 og ændrer linien således:

```
10 PRINT "2*3+1 = ";2*3+1
```

glem ikke semikolon

```
RUN
2*3+1 =  7
6
```

mellemrummet er til facits fortegn

facit for linie 20 forbliver det samme

Hvis du ønsker både at UDSKRIVE regnestykket OG at regne det, må du indtaste det to gange: en gang i anførselstegn og en gang uden, således:

```
10 PRINT "2+2 = ";2+2
```

Direkte tilstand

Man kan indføje enhver beregning i et program, eller man kan få facit omgående ved at indtaste PRINT samt opgaven uden linienummer og så trykke RETURN, således:

```
PRINT 3-6
-3
PRINT 24/(6+2)
3
```

Hvis du ikke har et linienummer foran en BASIC instruks, behøver du hverken ved tal, kommandoer eller tekst at indtaste RUN for at beordre computeren til at følge instruksen; du er i DIREKTE TILSTAND. Hvis der er et linienummer, betyder det, at instruksen er en del af et BASIC-program; du er i PROGRAM TILSTAND. Begge metoder kan bruges.

Du kan også i en enkelt PRINT instruks i direkte tilstand inkludere både en tekstsætning i anførselstegn og et matematisk problem, der skal løses.

denne pil står for eksponentialfunktion; denne fås ved at skrive SHIFT og 0

```
PRINT "2 OPLØF'TET TIL TREDJE POTENS ER LIG MED"; 2 ↑ 3

2 OPLØFTET TIL TREDJE POTENS ER LIG MED 8
```

meddelelsen udskrives og derefter udskrives facit

Rækkefølge ved beregning

Det andet eksempel i sidste afsnit viser, at man kan udføre mere end en beregning på en linie. Prøv at skrive følgende:

```
PRINT 200+50/5
```

Er resultatet det, du ventede? Prøv følgende:

```
PRINT (200+50)/5
```

Din **C 16-plus/4** udfører altid beregninger i en bestemt rækkefølge. Opgaverne løses fra venstre til højre; inden for den hovedregel løses visse typer beregninger først. Den rækkefølge, i hvilken din **C 16-plus/4** vurderer udtryk, kaldes operatorernes rangfølge.

FØRST: Kontrollerer din **C 16-plus/4**, om der er negative tal (ikke subtraktion, kun negative tal).

SÅ: Løser din **C 16-plus/4** eventuelle eksponenter.

DEREFTER: Løser din **C 16-plus/4** alle multiplikationer og divisioner, fra venstre mod højre.

ENDELIG: Løser din **C 16-plus/4** additioner og subtraktioner, fra venstre mod højre.

BEMÆRK: Din **C 16-plus/4** løser altid eventuelle parenteser først. Man kan endda sætte paranteser omkring parenteser: 36*(12+(A/3)). Indholdet af de inderste parenteser løses først.

Det kan somme tider være en god ide at sætte negative tal i parentes for klarhedens skyld. Hvis man f.eks. vil gange 45 med −5, skriver man følgende: 45*(−5). Din **C 16-plus/4** kan forstå opgaven med eller uden parenteser.

Brug af variabler

Eksemplet 36*(12+(A/3)) illustrerer en af computerens stærkeste sider. Da vi indsatte et bogstav i stedet for et tal i en matematisk opgave, brugte vi en VARIABEL. En variabel repræsenterer en værdi:

```
10 A = 3
20 PRINT "TOTAL:"; A*4
```

Hvis man kører dette program, bliver resultatet på skærmen følgende:

```
TOTAL: 12
```

Man kan anvende tre typer variabler:

TYPE	SYMBOL	BESKRIVELSE	PRØVE EKSEMPLER	VÆRDIER
Flydende komma		real(decimal) eller hele tal	X, AB, T4	23.5,12 1.3E+2
Helt tal	%	hele tal	X%, A1%	15, 102, 3
Tekststreng	$	bogstaver, tal og alle andre karakterer i anførselstegn	X$,MS$	"TOTAL:", "1.DAG", "CBM"

Hver gang man ønsker at en variabel skal være en integervariabel, (heltalsvariabel), indeholder symbolet for den variabel %-tegnet. En variabel der indeholder tekst SKAL slutte med et $-tegn som en del af variablen. Hvis den ikke indeholder dette symbol, opfatter **C 16-plus/4** den som et tal med flydende komma. En variabel, som ikke indeholder et af de to symboler (% eller $), læses som et tal med flydende komma (et "regulært" tal). Intergervariabler er tal uden decimaler (hele tal).

Du bør altid anvende den rette type variabel. Hvis du f.eks. forsøger at sætte en integervariabel = en tekststreng, vil dit program ikke fungere. Dette program viser dig, hvilken variabel du kan bruge eller ikke bruge i en given situation, og du kan finde ud af, hvad der sker, når du forsøger forskellige typer data:

```
10 REM DETTE PROGRAM KRÆVER NUMERISKE DATA
20 PRINT "INDTAST ET TAL"
30 INPUT X%                                    dette er inddatavaria-
40 PRINT "DET GÅR FINT; ES!"                   blen
50 PRINT "JEG LÆSER DIT TAL SOM"; X%
```

Prøv at indtaste følgende værdier og se, hvad der sker:

```
EN FEMTEDEL
.043
10
```

Numeriske funktioner

Dit **C 16-plus/4** BASIC 3,5 sprog indeholder numeriske funktioner, som ligner de avancerede beregninger, man har på de fleste videnskabelige regnemaskiner (som f.eks. sinus, kosinus, tangens o.s.v.).

Flertallet af funktionerne kan benyttes ved at indtaste navnet på funktionen efterfulgt af en parentes med det tal, der skal bearbejdes ved hjælp af formlen; således:

funktion(X)

Hvis man f.eks. vil finde sinus af en variabel indtaster man:

PRINT SIN(X)

Med X som et hvilket som helst tal, man ønsker at indtaste.

Man kan også indsætte en af funktionerne i en programlinie, som følgende eksempel viser:

```
10 FOR X = 1 TO 5
20 PRINT "KVADRATRODEN AF";X "ER"; SQR(X)
30 NEXT X
```

BASIC 3,5 Encyklopædien indeholder en komplet fortegnelse. Nogle af de mere komplekse funktioner forklares i det følgende.

Tilfældige tal og andre funktioner

At vælge et tilfældigt tal svarer til at tage 10 stykker papir, skrive et tal fra 1 til 10 på hvert og lægge de 10 stykker papir i en hat, hvorefter man udtrækker 1 stykke. Det udtrukne tal er et TILFÆLDIGT tal. Tallet lægges tilbage i hatten og et andet udtrækkes. Hver gang et tal udtrækkes, lægges det tilbage i hatten og et andet udtrækkes. Hver gang et tal udtrækkes, lægges det tilbage i hatten og således forbliver antallet af mulige tal det samme. Når et tal udvælges, er det umuligt at vide, hvilket tal det næste bliver, men man ved, at tallet vil ligge mellem 1 og 10. Dette er grundlaget for "TILFÆLDIGE TAL" FUNKTIONEN.

Tilfældige tal er overordentligt nyttige ved programmering, fordi de bidrager med chancefaktoren eller (naturligvis) tilfældigheden. Tilfældige tal repræsenterer almindeligvis en værdimængde, d.v.s., der er en øvre og en nedre grænse for de tal, man kan trække. I eksemplet med hatten er værdimængden fra 1 til 10. Den nedre grænse er "1" og den øvre grænse er "10", hvilket betyder, at ethvert tal mellem 1 og 10 tilfældigt kan dukke op, hver gang et nyt tal trækkes.

Lad os undersøge, hvordan din **C 16-plus/4** behandler tilfældige tal og nogle af de ting, man kan foretage med dem. Det følgende program frembringer fem fuldstændigt tilfældige tal:

```
10 FOR X = 1 TO 5: PRINT RND(X): NEXT X.
```

Disse tilfældige tal er alle temmeligt komplekse, med flere decimaler, men de fleste anvendelsesområder for tilfældige tal kræver hele tal. Man kan få sine tal frem som hele tal (uden decimaler) ved at anvende integer funktionen, som afskærer alle cifrene på højre side af kommaet. Det følgende er en formel til frembringelse af tilfældige tal inden for en hvilken som helst talrække, man måtte ønske. Man kan anvende denne formel næsten hvor som helst i sit program, man ville bruge en variabel eller et tal.

INT(talrække*RND(1))+nedre grænse

INT kommandoen får computeren til at afskære eventuelle decimaler og kun give hele tal som 1, 45 eller 320 i stedet for tal som 1.223, 45.6677 eller 320.59. Hele tal er lettere at arbejde med, når man anvender tilfældige tal.

Nedre grænse i formlen refererer til det laveste af de tal, man ønsker, computeren skal vælge blandt.

Talrækken er det antal tal, der er i hele gruppen.

Hvis man f.eks. vil vælge et tilfældigt tal mellem 1 og 5, er den nedre grænse 1 og talrækken er 5. Hvis man vil vælge et tilfældigt tal mellem 15 og 20 er den nedre grænse 15 og talrækken 6, fordi man vælger fra en pulje på 6 tal. Vælger man tal mellem 2 og 100 er den nedre grænse 2 og talrækken 99. Lad os nu prøve et program:

```
10 PRINT INT(5*RND(1))+1
```

Indtast RUN og tryk RETURN. KØR programmet nogle få gange. Hver gang man kører programmet, får man et tilfældigt tal mellem 1 og 5. Lad os nu indtaste 15 tilfældige tal med nedre grænse 1 og talrække 5 ... bemærk at alle de 15 valgte tal er udvalgt tilfældigt mellem 1 og 5:

```
10 FOR X = 1 TO 15
20 PRINT INT(5*RND(1))+1
30 NEXT X
```

sætter løkke for 15 gange

udvælger random (tilfældigt) tal

Indtast RUN og tryk RETURN.

En effektiv måde at anvende denne formel på er at gøre den til en bruger-defineret funktion. Bruger-definerede funktioner er særdeles vigtige i matematiske beregninger og overordentligt lette at iværksætte ved hjælp af **C 16-plus/4**. Bruger-definerede funktioner tillader programmering af en formel og lader så **C 16-plus/4** indsætte de værdier, der skal beregnes. Dette kan anvendes til mange forskellige formål.

Her er en instruks der udnytter den bruger-definerede funktion til frembringelse af tilfældige tal:

```
10 DEF FNR(X) = INT(X*RND(1))+1
```

Dette giver os tilfældige tal inden for talrækken fra 1 til X. FNR er navnet på den funktion, der er defineret ved denne instruks.

Eksemplet med anvendelse af en defineret funktion:

```
10 DEF FNR(X) = INT(X*RND(1))+1
20 DO
30 COLOR 1, FNR(16),5: REM VÆLG EN TILFÆLDIG FARVE FRA 1-16

40 PRINT "EFTERSØGNINGEN FORTSÆTTER"
50 LOOP
```

Brug af den definerede funktion sparer hukommelsesplads, når funktionen anvendes mere end én gang, og det gør dine programmer lettere at læse og forstå.

Kapitel 6

Basic for begyndere

* Introduktion
* Programmeringsmåder
* Indlæsningsinstrukser/udskrivningsinstrukser
* Styreinstrukser og løkker
* Betingede instrukser
* Underrutiner
* Bemærkninger

Introduktion

Indtil nu har du måske og måske ikke forstået præcis, hvad der foregik i de programmer, der introducerede dig til nogle af de muligheder, der er gemt i din **C 16-plus/4**. I dette kapitel forklares nogle af de BASIC kommandoer, der blev brugt i disse programmer. Dette kapitel koncentrerer sig om nogle af de hyppigst forekommende BASIC instrukser, som du vil få brug for til konstruktion af dine egne programmer. I slutningen af kapitlet berører vi nogle fremgangsmåder ved programmering. Dette kapitel giver en hurtig introduktion til programmering, men det er kun en introduktion. Hvis du ønsker at lære egentlig programmering, vil vi foreslå, at du vælger en god bog om BASIC (programmeringssprog, Beginner's All-Purpose Symbolic Instruction Code) hos din boghandler. Der er mange versioner afBASIC med små individuelle forskelle. Din **C 16-plus/4** er udstyret med en avanceret udgave af BASIC sproget, der hedder Commodore BASIC 3.5.

Programmeringsmåder

Din **C 16-plus/4** giver dig 2 muligheder for anvendelse af BASIC instrukser og kommandoer:
Direkte tilstand og indirekte tilstand. Direkte tilstand kaldes ofte umiddelbar tilstand og indirekte tilstand er også kendt som programtilstand. DIREKTE eller UMIDDELBAR TILSTAND udfører som betegnelsen antyder instrukser og kommandoer omgående (så snart man trykker RETURN efter at have indtastet en kommando). Man indtaster ikke linienumre, når man indtaster instrukser eller kommandoer i direkte tilstand. Man indtaster kun kommandoen eller instruksen og trykker RETURN tasten. Denne metode anvendes, hvis man ønsker, at computeren skal udføre beregninger og give resultatet straks. Kommandoer som LIST, SAVE, LOAD, VERIFY og RUN anvendes sædvanligvis i direkte tilstand. Størstedelen (men ikke alle) BASIC instrukser arbejder i direkte tilstand.

INDIREKTE TILSTAND eller PROGRAMTILSTAND giver mulighed for, at man kan samle et antal BASIC instrukser i et sæt, der udføres i den rækkefølge, man ønsker. Hver linie i programmet har et linienummer, som får computeren til at udføre linierne i en bestemt rækkefølge. Du har allerede set flere eksempler på programtilstand i Kapitel 4. Glem ikke, at når man arbejder i program tilstand, skal man trykke RETURN for hver enkelt linie i programmet, man indtaster i **C 16-plus/4**'s hukommelse. Hvis man ikke trykker RETURN og blot fortsætter til næste linie, kommer den linie, man indtastede, ikke med. Når først programmet er i hukommelsen,

sker der intet, før man indtaster RUN kommandoen. RUN kommandoen giver **C 16-plus/4** besked på at køre programmet, idet den starter med linien med det laveste nummer.

Linier nummereres som oftest i spring på 10, da man tit kommer ud for at skulle tilføje linier forskellige steder under udarbejdelsen af et program. Om nødvendigt kan man f.eks. tilføje ni nye linier mellem linie 10 og linie 20 i et program. Din **C 16-plus/4** indeholder en BASIC kommando RENUMBER, som gør det muligt at tilføje nye linier samt ændre eksisterende linienumre. Derved undgås en masse forvirring, der ofte opstår, når man ændrer og omarrangerer linier.

Indlæsnings/udskrivningsinstrukser

INDLÆSNINGS/UDSKRIVNINGS (I/U) instrukser benyttes i programmer til at kommunikere med den person, der kører programmet. Før programmet køres er der, hvis alle data til beregningerne foreligger, kun ringe brug for indlæsningsinstrukser. Det er ofte nyttigere, hvis computeren kan få data fra den person, der kører programmet (vi kan kalde ham eller hende for programbrugeren). Programmerne er langt mere smidige, hvis datamængden ikke "er hugget i sten", før kørselen. Udskrivningsinstrukser kan bruges af computeren til at meddele den, der kører programmet de resultater, computeren har beregnet. Udskrivningsinstrukser er helt indlysende yderst vigtige. Der ville ikke være megen fornuft i at køre et program, der ikke indeholdt udskrivningsinstrukser. (Kan vel sammenlignes med et træ, der falder i skoven, uden at der er nogen i nærheden, der kan høre det; frembringer det lyd? Betyder det noget?).

Avancerede programmører bruger også I/U instrukser til kommunikation med eksterne enheder i stedet for med programbrugeren. Det har du formodentlig selv gjort, men ikke i et program – da du brugte LOAD eller SAVE i forbindelse med din datasette eller diskettestation. LOAD er i grunden en indlæsningsinstruks, eftersom **C 16-plus/4** modtager data (dit program) fra din datasette eller diskette station, medens SAVE er en udskrivningsinstruks, idet **C 16-plus/4** sender data til disse eksterne enheder.

I denne introduktion til I/U instrukser vil vi begrænse os til nogle få af de vigtigste, som du straks får brug for. Det er følgende: PRINT, INPUT, GETKEY og READ/DATA. PRINT er en udskrivningsinstruks, medens de øvrige er indlæsningsinstrukser. (Husk at samtlige BASIC I/U instrukser findes i BASIC Encyklopædien bagest i denne bog).

Instruksnavn: PRINT

Format: PRINT "tekst i anførselstegn" eller variabler eller tal eller beregninger o.s.v.

Du har brugt PRINT instrukser tit i programmer i tidligere kapitler. Deraf, og af formateksemplet ovenfor, fremgår det, at PRINT er en overordentlig anvendelig instruks. Den kan bruges til at udskrive meddelelser, billeder fremstillet med grafiske karakterer, beregninger, der kan vise værdien af en variabel, o.s.v. Da PRINT instruksen anvendes så hyppigt, kan det betale sig at lære at bruge den godt.

Brug nr. 1 Tekstvisning
Lad os antage, at du i dit program ønsker at meddele brugeren, at hans eller hendes checkkonto udviser negativ saldo, eller at purpurøgler ikke har adgang til kontrolrummet. Det letteste ville være at UDSKRIVE instruksen som en tekststreng. Tekststrenge udskrives nøjagtigt, som de indtastes. De skal være i anførselstegn (" "). F.eks. ville instruksen:

```
100 PRINT "DU ER FALLIT!"
```

fortælle brugeren, at der ikke var flere penge tilbage, medens

```
150 PRINT "DU MÅ IKKE TAGE DIN VEN MED IND I KONTROLRUMMET"
```

kunne bruges i det andet eksempel.

Alt hvad der står i anførselstegn kaldes en litteral, fordi den UDSKRIVES præcis, som den fremkommer. Det er ligegyldigt, om det er ord, bogstaver, tal, skilletegn o.s.v.

Visse taster, som f.eks. markøren og farvetasterne fungerer anderledes, når de bruges i tekststrenge. I stedet for at ændre farven eller bevæge markøren, når den pågældende tast trykkes, skrives en karakter i strengen i negativ. Når programmet KØRES oversættes karakteren til det, du ønskede skrevet fra begyndelsen. Dette tillader sletning af skærmen, ændring af farven, du udskriver i, flytning af markøren, alt indlagt i programmet, prøv f.eks. følgende:

```
10 PRINT"<SHIFT><CLR/HOME><CONTROL>3 AFPRØVNING<CRSR NED><CONTROL>7 AFPRØVNING"
```

Husk at trykke de følgende taster samtidig, når du anvender SHIFT og CONTROL tasterne. De symboler, der fremstår i negativ, er de signaler,

der meddeler computeren, at den skal slette skærmen, ændre farven eller flytte markøren.

Brug nr. 2 ved udskrift af Tal og Beregninger
PRINT kan vise resultatet af en beregning foretaget i udskrivningsinstruksen. (SE TAL og BEREGNINGER). **C 16-plus/4** udfører de funktioner, der er nødvendige for at få resultatet, hvorefter den viser det på skærmen. F.eks. vil instruksen

```
100 PRINT 58*15,23,45+1000-45*(4-3)
```

udskriver:

```
870 23 1000
```

Dette bliver mere interessant, når der også anvendes variabler. Bruger inddata kan vises og tidligere beregninger, lagret i variabler, kan også udskrives eller endda bruges i yderligere beregninger.

Eksempler:

INDTAST:

```
10 R = 10*2:N = R-5
20 PRINT "R ER"; R "OG N ER"; N
30 PRINT "MEN R GANGE 2 ER"; R *2
40 PRINT "OG N MINUS 2 ER"; N-2
```

Efter hver PRINT instruks bevæger markøren sig normalt automatisk til begyndelsen af næste linie. Man kan undgå dette ved at indsætte et semikolon (;) efter PRINT instruksen som f.eks.:

```
200 PRINT "DISSE TO SÆTNINGSDELE BLIVER";
210 PRINT "SKREVET PÅ SAMME LINIE"
```

Sætningsnavn: INPUT

Format: INPUT "meddelelse"; variabel skal være input

INPUT instruksen tillader dig at få data fra programbrugeren via tastaturet og at bruge dem i programmet. Denne meddelelse tillader dig at meddele brugeren præcis, hvad du beder om; når INDLÆSNINGSinstruksen er udført, udskrives meddelelsen tillige med et spørgsmålstegn. Så venter

C 16-plus/4 på, at brugeren indtaster et svar og derefter trykker RETURN tasten. Inddata fra brugeren anbringes i en variabel. Man kan enten få en streng fra brugeren ved at anvende en strengvariabel (A$ f.eks.), eller et tal ved at bruge en numerisk variabel. INDLÆSNINGSinstruksen kan kun bruges i programtilstand.

Eksempler:

INDTAST:

```
10 PRINT "HVAD HEDDER DU";
20 INPUT A$
30 PRINT "DET GLÆDER MIG AT TRÆFFE DIG"; A$;"."
40 INPUT "HVOR GAMMEL ER DU"; AG
50 PRINT AG; "ER LIDT ÆLDRE END JEG ER."
RUN
```

Instruksnavn: GETKEY

Format: GETKEY variabel skal være Inddata

GETKEY er en anden måde, hvorpå man kan indtaste data, medens programmet KØRES. GETKEYinstruksen accepterer kun en tast ad gangen. Uanset hvilken tast der trykkes, tildeles den den strengvariabel, du angav i GETinstruksen (A$ f.eks.). GETKEY er nyttig, fordi den tillader dig at indtaste data med en karakter ad gangen uden at skulle trykke RETURN tasten efter hver karakter. GETKEY instruksen kan kun bruges i et program.

Eksempel på GETKEY i et program:

```
1000 PRINT "VÆR SÅ VENLIG AT VÆLGE A, B, C, D, E, eller F"
1010 GETKEY A$
```

Instruksnavn: READ/DATA

Format: READ variabler, der skal være inddata
DATA dataelementer, der skal læses

READ/DATA instrukserne bruges som en bekvem metode til at sætte værdier på variabler. Man kan betragte READ instruksen som en INPUTinstruks, der snarere beder **C 16-plus/4** end brugeren om data. Data indeholdes (naturligt nok) i DATAinstrukser. Når **C 16-plus/4** udfører en

READ instruks, ser den på det næste dataelement i DATAinstruksen og overfører det til variablen i READinstruksen.

READinstruksen bruges altid med en DATAinstruks. En DATAinstruks er blot en linie med data (ord eller tal) i et program. READinstruksen anvendes til at overføre disse værdier til variabler. (For hver variabel, der er anført i READinstruksen "læser" din **C 16-plus/4** en værdi fra DATAlinien til netop den variabel). En DATAinstruks kan forekomme hvor som helst i programmet. Det man skal huske om READinstruksen er, at den anvendte type variabel skal være den samme som den type data, der er indeholdt i DATAinstruksen (talvariabler til tal, tekstvariabler til tekst). Ellers opstår der TYPE MISMATCH ERROR (se fejlmeddelelser).

Eksempel:

```
10 READ A$, B$, C$, D$, E$
20 PRINT A$: PRINT B$: PRINT C$
30 PRINT D$: PRINT E$
40 DATA GROUCHO, HARPO, CHICO
50 DATA ZEPPO, GUMMO
```

Computeren svarer

```
GROUCHO
HARPO
CHICO
ZEPPO
GUMMO
```

Styreinstrukser og løkker

Det ville være ret så kedeligt, hvis din computer kun kunne udføre programlinier i rækkefølge. Computeren kunne kun starte ved begyndelsen og gennemgå hvert enkelt trin i rækkefølge indtil slutningen på programmet. Dette ville føre til meget lange programmer. Hvis du ønskede at gøre det samme to gange (som f.eks. PRINT "HALLO") måtte du dublere programlinierne. Når der er tale om et lille eksempel som PRINT "HALLO", gør det ikke den store forskel, men det kunne blive vanskeligt med store programmer. Dette er grunden til, at computere har styreinstrukser. Styreinstrukser giver computeren besked på at ignorere programliniernes normale rækkefølge og fortsætte til en anden linie uden hensyn til rækkefølgen. **C 16-plus/4** indeholder mange slags styreinstrukser: ubetingede (som GOTO), der altid overtager kontrollen; løkker (som FOR/NEXT),

der overtager kontrollen et angivet antal gange; og specielt til struktureret programmering: DO/LOOP.

Instruksnavn: GOTO

Format: GOTO linie

GOTO giver din computer besked på straks at gå fra den linie i programmet, den er på nu og til det linienummer, der er angivet i GOTO instruksen. Hvis der f.eks. i linie 20 står GOTO 40, vil din **C 16-plus/4** springe til linie 40, idet den springer over eventuelle instrukser mellem linierne 20 og 40.

Eksempel på brug af GOTO instruks i et program:

INDTAST:

```
10 PRINT "EN KRONE SPARET ER BEDRE END INTET"
20 GOTO 10
```

Computeren svarer med at skrive meddelelsen i linie 10 igen og igen, indtil du trykker STOPtasten, således:

```
EN KRONE SPARET ER BEDRE END INTET
EN KRONE SPARET ER BEDRE END INTET
EN KRONE SPARET ER BEDRE END INTET

BREAK IN 10                          Hvis du trykker STOP
READY                                tasten
```

Denne printinstruks vil fortsætte i det "uendelige". Hver gang din **C 16-plus/4** når til GOTO i linie 20, går den tilbage til linie 10. Dette kaldes en ENDELØS LØKKE i computerterminologi. Selv om man måtte ønske at lave en sådan, vil man i reglen kun ønske at få et vist antal gentagelser eller gentagelser, indtil noget sker. Det er derfor FOR/NEXT og DO/LOOP instrukserne findes i BASIC.

GOTO kan også anvendes i direkte tilstand. GOTO linie vil starte programmet på den linie, man angiver, samtidig med at variablerne bevares uforandret (i stedet for at slette dem, som RUN gør).

Instruksnavn: FOR/NEXT

Format: FOR variabel = startværdi TIL slutværdi
nogle BASIC instrukser
NEXT variabel

FOR/NEXT instrukserne gør det muligt at lave en løkke, som vil gentages et vist antal gange. Programinstrukserne mellem FOR instruksen og den dertil svarende NEXT instruks gentages i løkken. Variablen i FORinstruksen fungerer som en tæller. Den sættes til at begynde med på den startværdi, man tilfører. Så køres programlinierne efter FOR indtil computeren når til den tilsvarende NEXT instruks. NEXT giver **C 16-plus/4** besked på at lægge en til tælleren. Hvis tælleren er mindre end eller lig med slutværdien, vender computeren tilbage til programlinien efter FORinstruksen. Ellers fortsætter **C 16-plus/4** med den første instruks efter NEXT.

Eksempel på brug af en FOR/NEXT løkke:

```
10 PRINT "TÆL OP ..."
20 FOR J = 1 TO 10

30 PRINT "VI HAR"; J
35 T = J
40 NEXT J
50 PRINT "VI TALTE OP TIL"; T
```

Man kan også specificere en STEPværdi i en FORinstruks. I stedet for at tilføje 1 til tællervariablen, tilføjer din **C 16-plus/4** en STEPværdi. Hvis man bruger et STEP på 5 i instruksen FOR M = 10 TO 30, f.eks. vil tællerne tælle 10, 15, 20, 25, 30 efter hver løkke. STEP kommandoen tillader endog, at man tæller baglæns (ved brug af en negativ STEPværdi).

Et andet eksempel med et negativt STEP:

```
10 PRINT "NEDTÆLLING ..."
20 FOR J = 10 TO 0 STEP-1
30 PRINT "VI ER VED"; J
35 T = J
40 NEXT J
50 PRINT "RAKETSTART VED"; T
```

Instruksnavn: DO UNTIL/WHILE ... LOOP UNTIL/WHILE

Format: DO UNTIL (betingelse) WHILE (betingelse)
Nogle BASIC instrukser
(EXIT)
LOOP UNTIL (betingelse) WHILE (betingelse)

Instrukskombinationen DO/LOOP er endnu en måde at lave en løkke på. Denne instrukskombination er meget stærk og alsidig. At lave løkke efter DO/LOOP metoden er en udbredt teknik i struktureret programmeringssprog. I dette kapitel vil vi kun behandle nogle få anvendelsesmuligheder.

Hvis man vil lave en endeløs løkke, skal man blot starte et programlinieafsnit med DO og slutte det med en LOOP instruks; således:

```
100 DO: PRINT "STIGER"
110 LOOP
```

Tryk STOP tasten for at afslutte programmet.

En nyttigere form er at kombinere DO/LOOP med UNTIL instruksen. Løkken vil køre kontinuerligt, medmindre betingelsen for UNTIL indtræffer.

```
100 DO: INPUT "KAN DU LIDE DIN COMPUTER"; A$
110 LOOP UNTIL A$ = "JA"
120 PRINT "TAK"
```

Om de andre måder, man kan bruge DO/LOOP på, se BASIC Encyklopædien i slutningen af bogen.

Betingede instrukser

Betingende instrukser bruges til at træffe beslutninger. En af computerens stærkeste sider er dens evne til at træffe beslutninger baseret på, hvad der foregår. En af de betingede instrukser, der kan fås med **C 16-plus/4**, betegnes som IF/THEN instrukser.

Instruksnavn: IF/THEN

Format IF betingelse THEN gør dette (kun hvis betingelsen
er rigtig)

Forenklet kan man sige, at IF/THEN instruksen fungerer således:

HVIS (denne instruks er rigtig) SÅ (udfør denne instruks).
Egentlig har du altid vidst, hvordan betingede instrukser fungerer. Hvor mange gange har du hørt følgende berømte linie?:

HVIS du spiser alle dine grønsager, SÅ får du dessert. Det kan forekomme en smule ligegyldigt, men det er essensen af IF/THEN instruksen.

Hvis betingelsen i IF instruksen er rigtig, udføres alt efter THEN.

EKSEMPEL:

```
10 INPUT "HVAD ER DET TIENDE BOGSTAV I ALFABETET"; A$
20 IF A$ = "J" THEN PRINT "RIGTIGT": GOTO 100
30 INPUT "ER DETTE ET A"; X$
40 IF X$ = "A" THEN 60
50 PRINT "FORKERT; PRØV IGEN"; GOTO 30
60 PRINT "INDTAST ET B"
70 GETKEY A$: IF A$ = "B" THEN PRINT "RIGTIGT"
100 PRINT "DET ER NOK"
```

I linie 40 siger vi bare THEN 60. Det betyder i virkeligheden THEN GOTO 60, men da THEN GOTO kombinationen bruges så tit, tillader BASIC, at man undlader GOTO. En valgmulighed til IF/THEN instruksen er ELSE instruksen, der beordrer din computer til at foretage en specifik handling, hvis den oprindelige IF betingelse ikke blev opfyldt. Et eksempel på ELSE instruksen kunne være: IF B>5 THEN 40 ELSE GOTO 10. BASIC Encyklopædien uddyber forklaringen på IF/THEN/ELSE instruksen.

Underrutiner

Hvis der i dit program er noget, som skal gentages mere end et sted i programmet, har du to muligheder: Du kan få gentagelse af programstumper eller du kan lave en underrutine. En underrutine er et afsnit i dit program, der kan bruges fra et hvilket som helst andet sted i dit program. Når underrutinen er afsluttet, fortsætter programmet automatisk ved instruksen lige efter det sted, hvor underrutinen blev kaldt frem.

Instruksnavn: GOSUB/RETURN

Format: GOSUB linie

GOSUB instruksen bruges til at kalde en underrutine. Ligesom ved GOTO instruksen overtages kontrollen af det linienummer, der er anført i instruksen. Til forskel fra GOTO, HUSKER **C 16-plus/4**, hvor GOSUB er placeret. Næste gang RETURN mødes, vil kontrollen vende tilbage tilinstruksen umiddelbart efter GOSUB instruksen.

Eksempel:

```
5 T = 0: FOR J = 1 TO 99
10 PRINT "GIV MIG ET TAL FRA 1-10"
20 INPUT N
30 IF N<1 THEN GOSUB 100:GOTO 20
40 IF N> 10 LHEN GOSUB 100:GOTO 20
50 T = T+N
60 NEXT J
70 PRINT "TOTALEN ER" T
80 END
100 PRINT "DET TAL LIGGER UDEN FOR TALRÆKKEN"
105 PRINT "INDTAST VENLIGST ET TAL MELLEM 1 OG 10"
110 RETURN
```

Hvis en RETURN mødes, når der ikke er nogle aktive GOSUB, får man en RETURN WITHOUT GOSUB ERROR. Man bør passe omhyggeligt på, at computeren aldrig kommer ind i en af underrutinerne undtagen via GOSUB. En måde er at gruppere GOSUB og GOTO instrukserne sammen, beskyttet fra normal programudførelse ved END instruksen.

Instruksnavn: REM

Format: REM meddelelse

REM instruksen bruges til kommentering (eller REMark) af dine programmer. REM instruksen udføres ikke som en del af programmet; det er en meddelelse, der kun kan ses, når man gennemgår listningen af et program. Det sker tit, at hvis man ikke kommenterer, har man seks måneder efter, at programmet skrives, glemt, hvad en eller anden del foretager. Man kan bruge REM instrukser til indsættelse af påmindelser, så man lettere kan regne ud, hvad man virkelig havde til hensigt, eller give andre personer oplysninger med sine meddelelser.

Eksempel:

```
1560 E = R/I*9:REM DETTE UDREGNER EN KASTERS TID
100 INPUT A;B: REM A ER HØJDEN I TOMMER OG B ER VÆGTEN
```

Resumé

Som nævnt i introduktionen er dette ikke tænkt som et komplet kursus i BASIC. Vi har blot givet dig nogle af grundbegreberne. Enhver BASIC kommando i **C 16-plus/4** findes i BASIC Encyklopædien med format, beskrivelse og eksempler. Vær ikke bange for at eksperimentere.

Kapitel 7

Brug af grafik og farve

* Grafiske tegn
* Animering med tegn
* Kontrol af farver
* Højopløsningsgrafik
* Punkter, linier og etiketter
* Firkanter, cirkler, polygoner og farvelægning
* Flerfarve grafik

Grafiske tegn

Hver bogstavtast indeholder 2 forskellige grafiske tegn og det gør ,-, *, og tegnet for engelsk pund også. Hvis man vil udskrive grafiske tegn, skal man holde SHIFT eller <C=>-tasterne nede, medens man trykker tasten for det grafiske symbol, man ønsker.

Når din **C 16-plus/4** er indstillet på store bogstaver/grafik, skal du holde SHIFT og trykke en bogstavtast for at få det grafiske tegn, der ligger på højre side af denne bogstavtast, vist på skærmen. Disse tegn omfatter spillekortfarverne, en massiv og en hul kugle samt et sæt linier og forbindelsestegn, som tillader dig at tegne mange forskellige billeder på skærmen.

Her følger nogle eksempler, som kan hjælpe dig at blive fortrolig med disse tegn:

Øvelse 1: Stor cirkel

Trin 1: Tryk SHIFT LOCK tasten
2: Tryk bogstavet U og derefter bogstavet I.
3: Tryk RETURN tasten
4: Tryk bogstavet J, derefter bogstavet K.
5: Tryk RETURN tasten

Øvelse 2: Slange

Trin 1: Tryk SHIFT LOCK tasten
2: Tryk U, derefter I, så U, så I, så U, så I.
3: Tryk RETURN tasten
4: Tryk K, derefter J, så K, så J, så K, så J.
5: Tryk RETURN tasten

Øvelse 3: Krum linie

Trin 1: Tryk SHIFT LOCK tasten
2: Tryk E, derefter D, så C, så *, så F, så R.
3: Tryk RETURN tasten

Øvelse 4: To kors

Trin 1: Tryk SHIFT LOCK tasten
2: Tryk M, derefter MELLEMRUM, så N, så MLR, så -.
3: Tryk RETURN tasten
4: Tryk MLR, så V, så MLR, så *, så +, så *.
5: Tryk RETURN tasten
6: Tryk N, så MLR, så M, så MLR, så -.
7: Tryk RETURN tasten

Når du er færdig, trykkes SHIFT LOCK tasten atter, så den springer op.

Undrede du dig over, at computeren ikke sagde SYNTAX ERROR (syntaksfejl), da du trykkede RETURN? Du havde jo faktisk tegn på linien, der ikke var kommandoer, som computeren kan forstå.

Grunden er, at **C 16-plus/4** ikke interesserer sig for den linie, du indtaster, hvis du holder SHIFT nede, medens du trykker RETURN. Hvis du trykker RETURN uden SHIFT tasten, forsøger computeren at regne ud, hvad du mener, når du blot tegner billeder.

Hidtil har vi ikke beskæftiget os med de grafiske tegn på tasternes venstre side. Disse grafiske tegn fungerer præcis som dem på højre side, med den forskel, at man holder <C▪>-tasten nede i stedet for SHIFT. Der er ingen lås på <C▪>-tasten, så den må man selv holde nedtrykket. Man kan lave dette sæt grafik med computeren indstillet på store bogstaver/grafik eller på store/små bogstaver. De er altid til rådighed.

De grafiske tegn på venstre side omfatter linier og vinkler til brug ved tegning af diagrammer og tabeller. Det følgende eksempel viser, hvordan man kan understrege et ord:

Flyt først markøren til linien under det ord, du vil understrege. Hold derefter <C▪>-tasten og T tasten, som udskriver en understregning. Hold disse to taster nede, indtil ordet er understreget.

Øvelse 5: Halv barre

Trin 1: Hold <C▪>-tasten med den ene hånd under hele øvelsen
2: Tryk D, derefter I, så I, så F.
3: Tryk RETURN tasten

Øvelse 6: Kile

Trin 1: Hold <C=>-tasten nede og tryk T, derefter Y, så U.
2: Hold CONTROL tasten nede og tryk 9
3: Hold <C=>-tasten nede og tryk I, derefter O, så P, så @, så MELLEMRUM
4: Tryk RETURN tasten

Øvelse 7: Vindue

Trin 1: Hold <C=>-tasten nede indtil trin 4
2: Tryk A, derefter R, så S, så RETURN
3: Tryk Q
4: Slip <C=> (bare rolig, det er rigtigt)
5: Hold SHIFT nede og tryk +
6: Slip SHIFT og hold <C=>-nede igen. Slip den ikke mere
7: Tryk W, derefter RETURN
8: Tryk Z, derefter E, så X, så RETURN

Formålet med disse øvelser er at vise, hvordan de grafiske tegn i **C 16-plus/4** kan benyttes til at skabe forskellige former og figurer. Det er kun en håndfuld af de figurer og fremstillinger, du kan udvikle. Nu, da du har fået et indtryk af, hvordan man anvender de grafiske tegn til at bygge forskellige former, skulle du selv eksperimentere med dem og se, hvad der kan komme ud af det.

Animering med tegn

Spillefilm er i virkeligheden en serie fastbilleder. Hvert billede er lidt forskelligt fra det foregående. Forevisningsapparatet viser hvert enkelt billede et kort øjeblik og går så videre til det næste. Scenen bliver gjort levende.

Computer animering virker på samme måde. Først tegner computeren et billede, så ændrer den billedet en lille smule. **C 16-plus/4** er hurtig nok til at lade genstande bevæge sig frit over hele skærmen i dine spil og praktiske programmer.

Man kan ikke taste hurtigt nok til at skabe bevægelse. En spillefilm animeres med en hastighed af 30 billeder pr. sekund. Billedskiftene skal ske så hurtigt, at de narrer øjet. Derfor skal man bruge et program til at tegne et billede, vente en brøkdel af et sekund og så skifte til et nyt billede.

For at få programmet til at lave billeder bruger vi PRINT instruksen med de grafiske tegn. Den enkleste form for animering består i at skifte mellem to tegn for at opnå bevægelse.

Følgende program efterligner bevægelsen af en hoppende bold.

Skriv NEW og tryk RETURN før indtastning af hvert nyt program. Husk at trykke RETURN efter hver linie i alle disse programmer.

```
10 PRINT "<HOME> <SHIFT> Q"
20 FOR L = 1 TO 100
30 NEXT L
40 PRINT "<HOME> <SHIFT> W"
50 FOR L = 1 TO 100
60 NEXT L
70 GOTO 10
```

Tryk disse taster samtidig

Indtast RUN og tryk RETURN

For at få en mere interessant virkning, kan man konstruere et lille billede af flere grafiske tegn, og derefter ændre nogle få af tegnene, medens man lader andre blive stående på samme sted. Dette giver en virkning som om en del af en genstand bevæger sig og vises i det følgende program.

VIGTIGT: Hver gang SHIFT eller <C=> omtales, skal den pågældende tast trykkes SAMTIDIG med den efterfølgende tast, når programmet indlæses.

```
10 PRINT "<HOME> <SHIFT> M <SHIFT> W <SHIFT> N"
20 PRINT "<C=> +"
30 PRINT "<SHIFT> N<SHIFT> M"
40 FOR L = 1 TO 100: NEXT L
50 PRINT "<HOME><SHIFT> W"
60 PRINT " <C=> T <C=>+<C=> T"
70 PRINT"<C=> M <C=> G"
80 FOR L = 1 TO 100:NEXT L
90 GOTO 10
```

MELLEMRUM HER

Indtast RUN og tryk RETURN

I begge eksempler på animering har vi indtil nu kun arbejdet på ét sted på skærmen. Næste skridt bliver at flytte omkring med den animerede figur.

TAB funktionen hjælper, når man ønsker at flytte genstande fra venstre kant. Følgende program afbilder en slange, der kryber på skærmen.

Husk at SHIFT og den efterfølgende tast stadig trykkes samtidig.

```
5 FOR A = 0 TO 30
10 PRINT" <SHIFT> <CLR>"
20 PRINT TAB(A)" <SHIFT> U <SHIFT> I <SHIFT> U <SHIFT> I"
30 PRINT TAB(A)" <SHIFT> K <SHIFT> J <SHIFT> K <SHIFT> J"
40 FOR L = 1 TO 100:NEXT L
50 PRINT" <SHIFT> <CLR> "
60 PRINT TAB(A+1)" <SHIFT> I <SHIFT> U <SHIFT> I <SHIFT> U"
70 PRINT TAB(A+1)" <SHIFT> J <SHIFT> K <SHIFT> J <UHIFT> K"
80 FOR L = 1 TO 100:NEXT L
90 NEXT A
```

Ved brug af tegn som bolden (SHIFT Q), kan man spille videospil på skærmen. For at få en bold til at bevæge sig, skal man bare slette den og anbringe den et andet sted, som i det følgende program.

```
10 PRINT " <SHIFT> <CLR> "
20 PRINT" <SHIFT> Q <CURSOR VENSTRE> ";
30 FOR L = 1 TO 60:NEXT L
40 GOTO 20
```

Venstre markørpil

Lav et mellemrum her

Indtast RUN og tryk RETURN tasten. Tryk STOP tasten, når du ønsker at standse boldens bevægelser.

Styring af farver

Man kan indsætte forskellige farver på hvert afsnit af skærmen. Randen kan have én farve, baggrunden en anden og hvert tegn kan have sin egen farve. Du ved allerede, hvordan man sætter tegnfarver ved brug af tastaturet. COLOR kommandoen justerer farven på de øvrige afsnit af skærmen.

Gør randen af skærmen rød ved at indtaste kommandoen COLOR 4,3 og trykke RETURN tasten. Tallet 4 i kommandoen står for skærmranden og farvenummeret 3 er rød (samme nummer som på tasten mærket RED).

Skriv nu COLOR 0,7 og tryk RETURN. Skærmbaggrunden bliver blå. Tallet 0 står for baggrunden, medens 7 er blå (også det samme som på tastaturet).

Det første tal efter ordet COLOR står for det sted på skærmen, man vil ændre. Område 0 er baggrunden, 1 er tegnfarven, 4 er randen. Vi vil beskæftige os med områderne 2 og 3, når vi kommer til flerfarve grafik senere i kapitlet.

Skærmområdenumre

OMRÅDE NR.	OMRÅDEBETEGNELSE
0	Baggrund
1	Tegn
2	Flerfarve 1
3	Flerfarve 2
4	Rand

Hver farve har også et justerbart luminansniveau, som kaldes luminansen. Man kan tilføje et tal fra 0 (mørkest) til og med 7 (lysest) efter farvetallet for at ændre farven. Skriv COLOR 4,3,0 og tryk RETURN. Skærmranden bliver mørkerød. Skriv COLOR 4,3,7 og skærmranden skifter til klar rød.

Farvenumre

FARVE NR.	FARVE	FARVE NR.	FARVE
1	SORT	9	ORANGE
2	HVID	10	BRUN
3	RØD	11	GULGRØN
4	CYAN (turkis)	12	LYSERØD
5	VIOLET	13	BLÅGRØN
6	GRØN	14	LYSEBLÅ
7	BLÅ	15	MØRKEBLÅ
8	GUL	16	LYSEGRØN

Kort sagt ser COLOR kommandoen således ud:
COLOR område, farve, luminans.
Her følger et hurtigt program, der viser dig alle farverne i **C 16-plus/4**:

Skriv først NEW og tryk RETURN. Glem ikke at trykke RETURN efter hver programlinie.

```
10 COLOR 0,7,7
20 FOR M = 0 TO 7
30 FOR N = 1 TO 2
40 FOR L = 1 TO 16
50 PRINT" <CONTROL> <RVS ON> ";
60 READ A
70 COLOR 1,A,M
80 PRINT " ";
90 NEXT L
100 PRINT
110 RESTORE
120 NEXT N,M
130 COLOR 1,2,4
200 DATA 7,14,4,13,6,16,11,8,10,9,3,12,5,15,2,1
```

Indtast nu RUN og tryk RETURN for at få en skærm i klart blåt med hver af de 15 andre farver vist i hvert luminansniveau.

BEMÆRK: Som flertallet af BASIC's grafiske udtryk, der behandles i dette kapitel, omtales COLOR skiftevis som en instruks (statement) og som en kommando (command). Sondringen er uvæsentlig ved forklaringen af COLOR eller nogen anden af de grafiske kommandoer, da den er baseret på, om udtrykket hyppigst anvendes i direkte tilstand eller programtilstand.

Højopløsningsgrafik

Din **C 16-plus/4** skærm indeholder 25 rækker á 40 karakterer, med andre ord, totalt 1000 karakterer på skærmen. Hver karakter dannes af enkelte prikker, idet 8 rækker med hver 8 prikker udgør en hel karakter. Din skærm rummer ialt 320 prikker i hver række og ialt 200 rækker prikker eller totalt 64.000 prikker. Din **C 16-plus/4** kan styre hver enkelt prik.

Ved brug af almindelig grafik har man begrænset kontrol over de enkelte prikker. Man skal bruge de 256 karakterer i hvert karaktersæt, der er indbygget i **C 16-plus/4**, hvilket tillader konstruktion af mange billeder. Men forestil dig, hvor mange man kunne konstruere, hvis man kunne styre hver enkelt prik for sig.

C 16-plus/4's højopløsningsgrafik giver netop denne mulighed. Man kan give kommandoer, der tillader en at tegne eller slette prikker, linier, cirkler og andre figurer.

Der er én begrænsning i højopløsningsgrafik. **C 16-plus/4** kan kun bruge 2 farver i hver 8×8 karakters position. Det vil sige, at hvert 8×8 areal på skærmen, hvor der normalt er karakterer, er begrænset til to farver (forgrunds- og baggrundsfarve for den firkant). Man kan bruge forskellige farver for hver forskellig karakterposition, men kun to farver inden for den samme position. En anden grafisk metode, som vil blive behandlet senere i dette afsnit, flerfarve metoden, tillader op til fire forskellige farver pr. karakterposition på bekostning af den opløsning, der er mulig i metoden med høj opløsning.

Det følgende program anvender noget af **C 16-plus/4**'s højopløsningsgrafik, især GRAPHIC kommandoen. Begynd med at indtaste NEW og tryk RETURN. Tryk RETURN tasten efter at du har indtastet en linie. Efter at hele programmet er indtastet, taster du RUN og trykker RETURN som sædvanligt.

```
10 COLOR 0,1
20 GRAPHIC 1,1
30 FOR L = 2 TO 16
40 COLOR 1,L,4
50 DRAW 1,0,L* 12 TO 319,L*12
60 DRAW 1,L*18,0 TO L*18,199
70 NEXT L
80 FOR L = 1 TO 5000:NEXT
90 COLOR 1,2,3
100 GRAPHIC 0
```

Bemærk at farverne skifter i nærheden af skæringspunkterne.

Hvis man ønsker at skifte fra almindelig grafik (også kaldet tekst tilstand), til højopløsningsgrafik, skal man bare indtaste kommandoen GRAPHIC 2,1 og trykke RETURN. Hvad sker der? Skærmen bliver tom og markøren dukker op igen nær nederste kant på skærmen. **C 16-plus/4** opdelte skærmen i 2 separate afsnit: det øverste til grafik og de nederste fem linier til tekst. Hvis man ikke har brug for de nederste fem linier til tekst, kan man bruge kommandoen GRAPHIC 1,1, men man kan så ikke se de kommandoer, man indtaster.

Man kan skifte frem og tilbage mellem grafik og tekst ved brug af GRAP-

HIC kommandoen. Kommandoen GRAPHIC 0 skifter tilbage til tekst på skærmen, medens GRAPHIC 2 skifter tilbage til højopløsningsgrafik uden at slette skærmbilledet. Tilføjelse af ,1 efter kommandoen sletter skærmbilledet.

GRAPHIC kommandoen ser sædvanligvis ud som følger:

GRAPHIC skærmtilstand, slet◁————denne del er valgfri

Skærmtilstandsnumre	Virkning
0	Tekst
1	Højopløsn.grafik
2	Højopløsn.grafik+tekst
3	Flerfarve
4	Flerfarve+tekst

Slet-numre	Virkning
0	Slet ikke skærmbilledet
1	Slet skærmbilledet

Der er en anden måde, hvorpå man kan slette højopløsnings-skærmbilledet. Kommandoen SCNCLR sletter skærmbilledet uden at ændre den grafiske skærmtilstand.

Når man først benytter højopløsningsgrafik, reserverer computeren 10K af sin hukommelse for højopløsnings-skærmbilledet. Denne hukommelse tages fra BASIC programområdet. Når man er færdig med brugen af grafik, kan man få denne hukommelse tilbage ved at bruge kommandoen GRAPHIC CLR.

Punkter, linier og etiketter

Indtast kommandoerne GRAPHIC 2,1: DRAW 1,0,0 og tryk RETURN. Se nøje på skærmens øverste, venstre hjørne. **C 16-plus/4** tegnede en sort prik der.

I DRAW (tegn) kommandoen er det første tal enten 1 (tegn en prik) eller 0 (slet en prik). De to næste tal angiver række- og kolonnepositionerne for prikken. Hvis du derfor ville tegne en prik i række 17, kolonne 20, skal du

bare indtaste DRAW 1,17,20. For at slette denne prik indtaster du DRAW 0,17,20.

DRAW kommandoen kan også tegne en linie mellem to vilkårlige punkter. Tilføj bare ordet TO og koordinaterne for den anden ende, således: DRAW 1,1,1 TO 100,100.

Hvis du er vant til at tegne kurver i matematik, bliver du måske en smule forvirret til at begynde med, når du bruger computeren. Koordinatsystemet i **C 16-plus/4** er anderledes end det, du er vant til. I matematik ville 0,0 punktet enten være i centrum eller nederste venstre hjørne af skærmen, men på computeren er det i øverste, venstre hjørne. Du vænner dig til computerens system, når du øver dig.

Når man først har sat en prik eller linie på skærmen, kan man tegne en linie fra den til et hvilket som helst andet punkt på denne måde: DRAW 1 TO 150,50. Derved tegnes en linie fra det sidst tegnede punkt til række 150, kolonne 50. Hvis programmet indeholder mange DRAW TO kommandoer, kan man placere den første prik i en position på skærmen ved brug af LOCATE kommandoen som ved LOCATE 100,100.

DRAW kommandoen kan optræde i flere skikkelser, som f.eks.

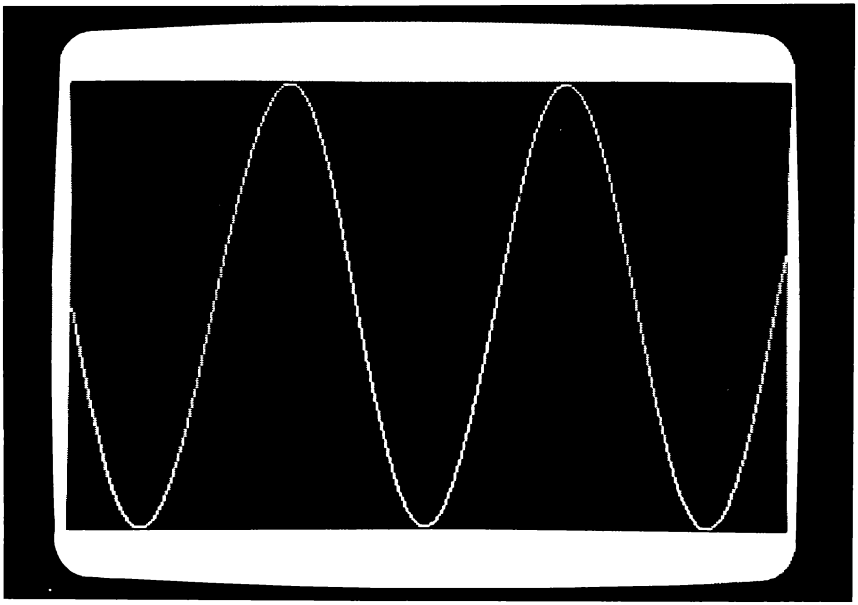
KOMMANDO	RESULTAT
Draw color source, row, column	PUNKT
DRAW color source, row, column TO row, column	LINIE
DRAW color source TO row, column	LINIE TEGNET FRA SIDSTE PUNKT

Color source (farvekilde) er 0 for baggrunden, 1 for forgrunden.

Til sletning af punkter eller linier på skærmen, bruger man DRAW kommandoen efterfulgt af tallet 0. Hvis man har lavet et punkt ved hjælp af DRAW 1,1,1, kan man slette det ved hjælp af DRAW 0,1,1. En linie, der er lavet ved hjælp af DRAW 1,1,1 TO 100,100, slettes ved hjælp af DRAW 0,1,1 TO 100,100.

Det følgende program tegner en kurve baseret på sinusfunktionen. Indtast NEW og tryk RETURN. Husk at trykke RETURN tasten efter hver linie. Indtast derefter RUN.

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 1,1
40 LOCATE 0,100
50 FOR X = 1 TO 319
60 Y = INT (100+99*SIN(X/20))
70 DRAW 1 TO X,Y
80 NEXT X
90 FOR L = 1 TO 5000
100 NEXT L
110 GRAPHIC 0
```



Indtast ikke NEW efter at du har kørt det sidste program. For anderledes udførelse af programmet, ændres linie 70 til:

```
70 DRAW 1,X,Y
```

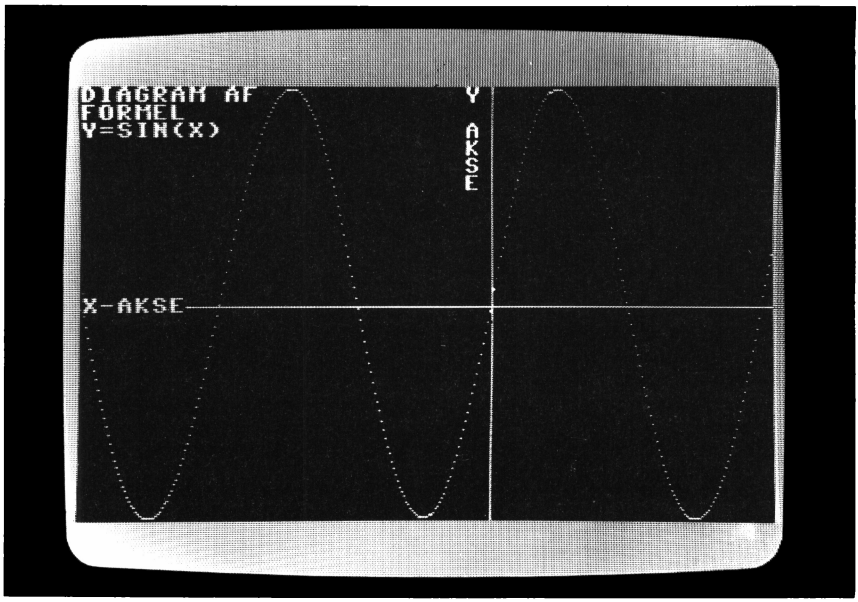
Dette program afsætter den samme kurve ved brug af prikker i stedet for linier.

CHAR kommandoen

Kurver bliver lettere at forstå og mere nyttige, hvis man sætter etiket på dem. Man kan bruge CHAR kommandoen til at indsætte tekst i en højopløsningstegning. Instruksen CHAR 1,0,5 "HALLO" indsætter ordet HALLO i femte række ved venstre kant af skærmen. Det første tal efter ordet CHAR er enten 1 (for karakter) eller 0 (for slet). De to næste tal angiver den kolonne og række, hvori teksten fremkommer.

Lad de to sidste programmer blive i computeren: indtast ikke NEW. Tilføj disse linier:

```
81 CHAR 1,0,0,"DIAGRAM AF":CHAR 1,0,1,"FORMEL"
82 CHAR 1,0,2,"Y = SIN(X)"
83 DRAW 1,0,100 TO 319,100,189,0 TO 189,199
84 CHAR 1,0,12,"X-AKSE":CHAR 1,22,0,"Y"
85 CHAR 1,22,2,"A": CHAR 1,22,3,"X"
86 CHAR 1,22,4,"I" CHAR 1,22,5,"S"
```



Firkanter, cirkler; polygoner og farvelægning

Ved brug af DRAW kommandoen kan man tegne billeder ved at afsætte mange prikker eller linier. Til tegning af et kvadrat kan man bruge kom-

mandoen DRAW 1,0,0 TO 100,0 TO 100,100 TO 0,100 TO 0,0 (idet hvert endepunkt af kvadratet afsættes), eller man kan nøjes med at bruge BOX kommandoen.

Tegning af rektangler

BOX KOMMANDOEN

Din **C 16-plus/4** indeholder en kommando, der gør det lettere at tegne kvadrater og andre firkantede figurer. BOX kommandoen tillader dig at vælge 2 modsatte hjørner af firkanten. Til duplikering af firkanten i ovenstående eksempel bruger du blot BOX 1,0,0,100,100. Tallet 1 betyder atter, at du vil tegne, ikke slette. De fire næste tal er koordinaterne for firkantens modsatte hjørner, (0,0) i øverste, venstre hjørne og (100,100) ved skærmens midte.

BOX kommandoen kan danne en hvilken som helst firkant, blot ved at man ændrer hjørnerne. Man kan endda dreje firkanten ved at angive en vinkel (i grader) efter den sidste koordinat. Således: BOX 1,50,50,100,100,45. Firkanten drejes 45 grader med uret.

Hvis man nu vil tegne en massiv (udfyldt) firkant i stedet for bare omridset, behøver man blot tilføje et komma 1 efter vinklen. En massiv firkant ved centrum af skærmen vises som BOX 1,100,50,220,150,,1. Bemærk at man behøver et komma til at tage vinklens plads, også selv om man ikke vil have firkanten drejet.

Følgende er eksempler på typiske BOX kommandoer:

KOMMANDO	VIRKNING
BOX tegn, række, kolonne 1, række 2, kolonne 2	Omrids
BOX tegn, r1, k1, r2, k2, vinkel	Drejning
BOX tegn, r1, k1, r2, k2,, farvelæg	Massiv firkant
BOX slet, r1, k1, r2, k2, vinkel	Slet område på skærm

Her følger et par programmer, der anvender BOX kommandoen:

Glem ikke at indtaste NEW og derefter trykke RETURN, før indtastning af hvert program og tryk RETURN efter hver linie, der indtastes.

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 2,1
40 A = RND(1)*20+10
50 FOR L = 0 TO 359 STEP A
60 BOX 1,100,30,220,130,L
70 NEXT L
80 FOR L = 1 TO 2000:NEXT L
90 GRAPHIC 0,1
```

```
5 TRAP 60
10 GRAPHIC 2,1
20 DEF FNA(X) = INT(RND(1)*X)
30 COLOR 1, FNA(15)+1
40 BOX 1, FNA(320),FNA(160),FNA(320),FNA(160),,1
50 GOTO 30
60 COLOR 1,2,3: GRAPHIC 0
```

Tryk RETURN og indtast RUN efter endt indtastning af hvert program. Hold STOP tasten for at slutte programmet.

Det andet program tegner forskelligt farvede firkanter over hele skærmen. Man kan se nogle områder af skærmen ændres, når andre områder i nærheden forandrer sig. Grunden til dette var vi inde på tidligere i dette kapitel.

Tegning af cirkler

Din **C 16-plus/4** har også kommandoer til tegning af cirkler. Som med BOX kommandoen, kan vi også variere formen på cirklen, så der dannes en oval (også kaldet en ellipse) og vi kan dreje ovalen. Vi kan også nøjes med at tegne en del af figuren (kaldet en bue).

Denne kommando tegner en cirkel i skærmens centrum: CIRCLE 1,160,100,50. Den giver **C 16-plus/4** besked på at tegne en cirkel med centrum ved række 160 og kolonne 100 med en radius på 50. Der fremkommer i virkeligheden en oval, da prikkerne på skærmen er højere end de er brede. For at ændre dette til en egentlig cirkel, skal man tilføje et separat tal for at fortælle, at højden er forskellig fra bredden, således: CIRCLE 1,160,100,50,42.

C 16-plus/4 kan også tegne en firkant, trekant eller anden polygon ved brug af CIRCLE kommandoen. Fortæl bare computeren, hvor mange grader der skal være mellem punkter på cirkelbuen, således: CIRCLE 1,160,100,50,42,,,,120. Denne kommando tegner en trekant, da hver side

er 120 grader. (Udeladelse af talværdier medens man includerer kommaer i en grafisk kommando får computeren til at læse standard (default)værdier i stedet for det manglende tal). En simpel formel for at få vinklen i en polygon med N sider er 360/N.

Her er et hurtigt program til tegning af polygoner:

```
10 GRAPHIC 2,1
20 INPUT "HVOR MANGE SIDER";A
30 IF A<2 OR A>100 THEN PRINT "VÆR IKKE FJOLLET":
GOTO 20
40 CIRCLE 1,160,80,40,33,,,,360/A
50 GOTO 20
```

Du kan vælge kun at tegne en bue i stedet for en hel cirkel. CIRCLE kommandoen accepterer start- og slutvinklerne i grader lige efter højdetallet. Kommandoen CIRCLE 1,160,100,50,42,90,180 viser kun den nederste, højre del af cirklen.

Til drejning af en oval tilføjer man vinklen for drejning med uret efter kommandoen, som i dette eksempel:
CIRCLE 1,160,100,100,20,,,30.

Det følgende er eksempler på de almindeligste CIRCLE kommandoer:

KOMMANDO	VIRKNING
CIRCLE på, midterrække, midterkolonne, radius	oval
CIRCLE på, midterrække, midterkolonne, bredde, højde	cirkel/oval
CIRCLE på, midterrække, midterkolonne, bredde, højde, start, slut	bue
CIRCLE på, midterrække, midterkolonne, bredde, højde ,,, vinkel	drejer oval
CIRCLE på, midterrække, midterkolonne, bredde, højde,,,, punkt vinkel	polygon

BEMÆRK: Når der er kommaer uden tal i en kommando som f.eks. CIRCLE eller BOX, tolker **C 16-plus/4** automatisk komma som en indlæsning af den udeladte værdi for den parameter i kommandoen. For eksempel, CIRCLE, 160,40,100,100 læses af computeren som CIRCLE 1,160 ..., idet den læser den udeladte værdi 1 for farvekilden.

Det følgende program bruges CIRCLE kommandoen til en interessant effekt. Indtast NEW og tryk RETURN for at slette det sidste program fra hukommelsen, før dette program indtastes.

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 1,1
40 A = RND(1)*20+10
50 FOR L = 0 TO 359 STEP A
60 CIRCLE 1,160,100,90,40,,,L
70 NEXT L
80 FOR L = 1 TO 2000: NEXT L
90 GRAPHIC 0,1
```

Her følger et program, du kan prøve, som bruger CIRCLE kommandoen til et velkendt design:

```
NEW
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 2,1
40 FOR L = 1 TO 4
50 Y = 50
60 IF L = 2 OR L = 4 THEN Y = 100
70 X = L*35+50
80 CIRCLE 1,X,Y,50,42
90 NEXT L
100 PRINT "PLUS/4 CIRKLER"
```

PAINT kommandoen

PAINT kommandoen udfylder et hvilket som helst afmærket område på skærmen ud til grænserne, der udgøres af eventuelle linier tegnet på skærmen. Hvis der ikke er tegnet nogen linier, udfyldes skærmen helt til randen. BOX kommandoen kan benyttes til at udfylde kasser og rektangler med farve. PAINT kommandoen kan farvelægge uregelmæssige figurer og andre uensartede områder på skærmen, der ikke kan udfyldes ved hjælp af andre kommandoer.

I den sidste øvelse konstruerede vi cirkler. Ved at tilføje nogle PAINT kommandoer til programmet kan vi farvelægge det område, der ligger mellem ringene.

Tilføj følgende linier til det sidste program:

```
110 FOR L = 0 TO 1
120 PAINT 1,120+35*L,75
130 NEXT L
```

Flerfarve grafik

C 16-plus/4's højopløsningsgrafik giver dig kontrol over hver eneste prik på skærmen, men som du har set, er evnen til at sætte farver tæt sammen begrænset. De fleste højopløsningsprogrammer kan kun anvende en eller to farver.

Til medtagelse af flere forskellige farver har din **C 16-plus/4** en særlig „ind imellem„ grafik tilstand, der kaldes flerfarve grafik. Ved brug af flerfarve grafik kontrollerer du halvt så mange prikker i hver række som i højopløsning, fordi hver prik er dobbelt så bred. Du får 160 prikker på hver række, men stadig 200 rækker. Man må betale for brugen af flere farver og prisen er en lidt lavere opløsning.

Når du begynder at bruge flerfarve grafik, så repetér først GRAPHIC kommandoen tidligere i kapitlet. Du kan se, at flerfarve skærmen uden tekst er GRAPHIC 3 og flerfarve skærmen med 5 linier tekst er GRAPHIC 4.

Se nu på tabellen over COLOR kommandoen. Der er to områder, vi endnu ikke har brugt, områderne 2 og 3. Disse områder indeholder to ekstra farver. Man kan anvende enhver af de tre farver (1, tekstfarven; 2, en ekstra farve; og 3, endnu en ekstra farve). Disse farver blander sig ikke

med hinanden på skærmen således som højopløsningsfarverne i nogle programmer tidligere i kapitlet.

Disse to programmer gør brug af flerfarve grafik, det første med ringene og det andet giver en „neon skilt„ effekt.

```
10 COLOR 0,1
20 GRAPHIC 4,1
30 FOR L = 1 TO 4
40 Q = L: IF Q>3 THEN Q = Q-3
50 COLOR Q,L+1
60 Y = 50
70 IF L = 2 OR L = 4 THEN Y = 100
80 X = L*18+25
90 CIRCLE Q,X,Y,25,42
100 NEXT L
```

Farveområde 3, det andet af flerfarve områderne, har en særlig egenskab, som ingen af de andre har. Når først man har tegnet på skærmen og brugt område 3, kan man, ved brug af COLOR kommandoen, ændre den farve overalt, hvor den dukker op på skærmen. Hvis man sætter farven med COLOR 3,5 og tegner med den farve, fremkommer grafikken i purpur. Hvis man så ændrer farven med COLOR 3,6 ændres alle purpurfarvede områder til grønt. Dette fungerer ikke i noget andet område.

Indtast NEW og tryk RETURN. Glem ikke at trykke RETURN tasten efter hver linie. Indtast RUN og tryk RETURN til sidst.

```
10 COLOR 0,1
20 GRAPHIC 3,1
30 COLOR 3,1
40 TRAP 200
50 DRAW 3,10,10 TO 10,100: DRAW 3,10,55 TO 30,55
60 DRAW 3,30,10 TO 30,100: DRAW 3,50,10 TO 80,10
70 DRAW 3,65,10 TO 65,100: DRAW 3,50,100 TO 80,100
80 FOR L = 0 TO 7
90 COLOR 3,2,L
100 FOR M = 1 TO 100: NEXT M
110 NEXT L
120 COLOR 3,1
130 FOR M = 1 TO 100: NEXT M
140 GOTO 80
200 GRAPHIC 0: COLOR 1,2,7
```

Kapitel 8

Frembringelse af lyd og musik med C16-plus/4

* Introduktion
* Volume kommando
* Sound kommando
* Frembringelse af lydeffekter
* Frembringelse af musik
* Musikmaskinen C-16-plus/4

Introduktion

Her følger et kort program til frembringelse af musik med din C 16-plus/4. Indtast programmet nøjagtigt, som det står og husk at trykke RETURN. VED SLUTNINGEN AF HVER LINIE. Når programmet beder dig om at indtaste et tal, indtaster du et hvilket som helst tal fra 0 til 1023 og trykker RETURN. Til standsning af programmet indtastes et nul.

```
10 VOL 8
20 DO
30 INPUT X
35 IF X>1023 OR X<0 THEN PRINT "0 TIL 1023, TAK": GOTO 30
40 SOUND 1, X, 10
50 LOOP UNTIL X = 0
```

Tryk RUN/STOP tasten for at standse programmet.

Sådan gør du, hvis du vil spille en enkelt tone på din **C 16-plus/4**. Indtast NEW og tryk RETURN for at slette det, der står i **C 16-plus/4**'s hukommelse.

Først: Indtaster du NEW og trykker RETURN
Indtast VOL 8 og tryk RETURN

Dernæst: Indtaster du SOUND 1,266,60 og trykker RETURN

Du skulle høre en tone spillet i ca. et sekund og derefter standse. Hvis du ikke hører noget, skruer du op for lyden på dit TVapparat eller monitor og prøver igen.

Disse to trin er de eneste kommandoer, du behøver for at kunne spille musik på din **C 16-plus/4**. Lad os se, hvad disse to kommandoer gør.

Volume kommandoen

VOL kommandoen styrer lydstyrken af de toner, **C 16-plus/4** spiller. Tænk på de første tre bogstaver i ordet „volumen„ for at huske VOL kommandoen. Tallet der følger efter VOL er styrkeindstillingen. Prøv at forestille dig VOL kommandoen som en styrkereguleringsknap på **C 16-plus/4**. Når knappen står på nul, er styrken skruet helt ned, og du hører intet. Når knappen er indstillet på 8, er der skruet helt op for lyden, og din **C 16-plus/4**spiller så højt den kan.

Prøv det første eksempel igen og brug et andet tal efter VOL kommandoen. Jo mindre tal, jo sagtere spilles tonen.

Sound kommandoen

SOUND kommandoen fortæller din **C 16-plus/4** alt, hvad den behøver at vide om den lyd, du vil spille. SOUND kommandoen efterfølges af tre tal, der beskriver tonen:

SOUND stemme, toneværdi, varighed.

Det første tal i sound kommandoen refererer til stemme. Tallet for stemme kan være 1,2 eller 3. **C 16-plus/4** lyden frembringes af to forskellige „stemmer", 1 for den første stemme og 2 for den anden stemme. Den tredje mulighed vedrører stemme 2s evne til at frembringe enten en tone eller støj.

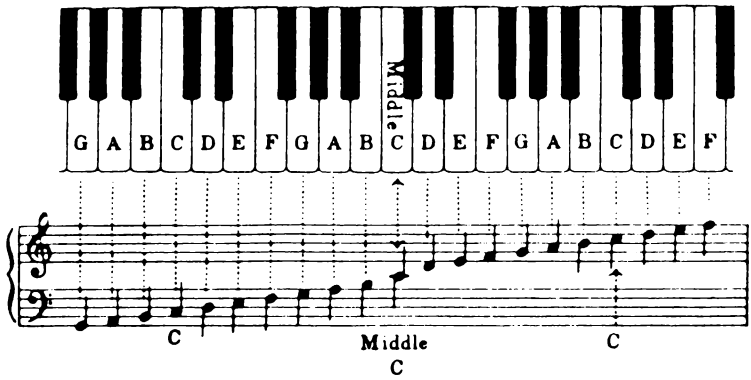
Stemme 1 – Denne stemme spiller kun toner. Vælg denne stemme med tallet 1 efter SOUND kommandoen.

Stemme 2 – Denne stemme er som stemme 1, men kan bruges til lydfrembringelse i form af toner eller støj. Indtast 2 i kommandoen, hvis denne stemme skal frembringe toner eller 3, hvis den skal bruges til støj til frembringelse af lydeffekter som torden og regn.

Det andet tal efter ordet SOUND er toneværdien (frekvensen). Det kan være et hvilket som helst tal mellem 0 og 1023. Det fortæller din **C 16-plus/4** hvor lav- eller højfrekvent en tone, den skal spille. Jo større tal, jo højere tone. De højeste værdier (i nærheden af 1023) kan ikke opfanges af det menneskelige øre.

BEMÆRK: Støj er kun „hvid„ i området 600-940 med stemme 3. Man kan anvende registerværdier uden for dette område til frembringelse af interessante lydeffekter.

Det følgende er en tabel, der viser alle tonerne i en skala tilligemed de toneværdier, der skal bruges. Der er en komplet tonetabel til **C 16-plus/4** i afsnit 11 i **C 16-plus/4** encyklopædien.



A	B	C	D	E	F	G
770	798	810	834	854	864	881
440.4	494.8	522.7	588.7	658	699	782.2

Prøv følgende program på din **C 16-plus/4**:

```
NEW

10 VOL 7                          indstiller lydstyrken
20 X = 0
30 DO
40 SOUND 1,X,5                    spiller tonen
50 X = X+5
60 LOOP UNTIL X = 1020
70 VOL 0                          afbryder lyden
```

Indtast RUN og tryk RETURN. Dette program giver din **C 16-plus/4** lejlighed til at vigte sig med noget af sin musikalske formåen.

Det tredje tal efter ordet SOUND bestemmer varigheden (længden) af tonen. Det fortæller **C 16-plus/4**, hvor længe den skal spille tonen. Tallet kan være et hvilket som helst fra 0 til 65535. Dette tal sætter en tidtager, der tager tid i tresindstyvendedele af et sekund. En varighed på 60 holder tonen i et sekund. En tommelfingerregel for varigheden er at jo højere tal, jo længere varer tonen. Faktisk varer tonen mere end 16 minutter, hvis du bruger 65535. Til at afbryde en lyd bruges en varighed på 0, som ikke tillader frembringelse af lyden.

En musikalsk lydeffekt

Lydeffekter kan man skabe enten ved brug af musikalske toner eller støj. Ved at kombinere enkle BASIC programmer og sound kommandoer kan man frembringe usædvanlige og underholdende effekter. FOR ... NEXT ... STEP løkken kan f.eks. anvendes kreativt i lydeffekter. Dette program anvender en FOR ... NEXT løkke med et negativt step til at tælle ned fra et højt tal til et lavere.

```
10 VOL 8                          indstiller lydstyrken på
20 FOR S = 1000 TO 700 STEP-25    8 laver løkke med ned-
                                  adgående steps
30 SOUND 1,S,1
40 NEXT S
```

Indtast RUN og tryk RETURN for at høre lydeffekten. Tonearten er linie 20, som vælger en talrække fra 1000 til 700, der går ned ad skalaen i spring på 25 tal ad gangen. Til sidst instruerer linie 30 din **C 16-plus/4** om at spille hver enkelt tone et kort øjeblik ved at indstille VARIGHEDEN på 1, som er 1/60 sekund. Eksperimenter med forskellige tal og varighedsværdier kan frembringe meget interessante effekter.

Frembringelse af støjlydeffekt

Brug af en værdi på 3 ved valg af stemme i en SOUND kommando bestemmer støjarten. Det bruges til frembringelse af lydeffekter med støj snarere end toner. Det følgende program bruger stemme 3 til at frembringe lyden af en storm.

```
10 VOL 2                          sætter lydniveauet væl-
20 R = INT(RND(0)*10)+1           ger tilfældigt tal fra 1 til
                                  10
30 FOR X = 1 TO R
40 SOUND 3,600+30*X,10
50 NEXT X
60 FOR X = R TO 1 STEP-1
70 SOUND 3,600+30*X,10
80 NEXT X
90 T = INT(RND(0)*100)+30
100 SOUND 3,600,T
110 GOTO 20
```

Linierne 30 og 60 sætter løkken for lydens toneværdi (frekvensen), en stigende og en faldende, baseret på det tilfældige tal fra linie 20. Det er

vigtigt, at der er varierende tonehøjde, da der er forskel på styrken af vindstødene i en storm. Linierne 40 og 70 er de SOUND kommandoer, der frembringer støjen. Linierne 90 og 100 indsætter tilfældige pauser for at genskabe et stormvejrs ujævne forløb med stormens tuden afbrudt af stille perioder. Programmet vælger et tilfældigt tal som varighedsværdi for en anden SOUND kommando. Tonehøjden for denne SOUND kommando er konstant, og kommandoen sørger for en ensartet baggrundsstøj, der tjener som et kontrapunkt til vindstødene.

Det at skabe lydeffekter ved hjælp af støj er yderst fængslende i forsøget på at fange de helt rigtige bestanddele af den lyd, man ønsker at frembringe. Det er nødvendigt at eksperimentere, hvis man vil frembringe gode lydeffekter.

Hvordan man laver musik

Nu da vi har beskæftiget os med nogle måder at skabe lydeffekter med **C 16-plus/4** på, går vi over til at lave musik, her følger et par programmer, man kan prøve:

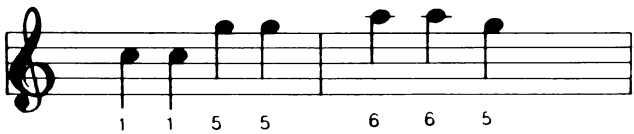
Det første program forvandler tasterne 1 til og med 8 til et klaver. Indtast programmet og indtast derefter RUN

```
5 SCNCLR                                        sletter skærmbilledet
10 FOR X = 1 TO 8: READ N(X):NEXT X
20 VOL 7
30 DO
40 GET A$: IF A$ = "" THEN 40
50 A = ASC(A$): IF A<49 OR A>56 THEN 90
60 N = A-48
70 SOUND 1,N(N),5
80 COLOR 0,N,3
90 LOOP UNTIL A = 32
100 VOL 0: COLOR 4,2,7
110 DATA 169,262,345,383,453,516,571,596
```

Man kan spille toner ved at nedtrykke tasterne 1 til og med 8. Skærmranden skifter endda farve i takt med de forskellige toner. Når man er færdig med at spille, trykkes på mellemrumstasten for at standse programmet.

Her følger de tal, man skal trykke for at spille en velkendt sang.

Twinkle, Twinkle Little Star
1 1 5 5 6 6 5
4 4 3 3 2 2 1
5 5 4 4 3 3 2
5 5 4 4 3 3 2
1 1 5 5 6 6 5
4 4 3 3 2 2 1



Dette program spiller en sang ved at læse en række DATA instrukser. Data instrukserne er ordnet parvis. Det første tal er toneværdien for SOUND kommandoen og det andet tal er kommandoens varighedsværdi.

ROW BOAT

```
10 VOL 8
20 DO
30 READ X,Y
40 SOUND 1,X,Y
45 FOR D = 1 TO 550:NEXT

50 LOOP UNTIL X = 0
60 END
100 DATA 169,45,169,45,169,30
110 DATA 262,15,345,45,345,30
120 DATA 262,15,345,30,383,15
130 DATA 453,60,596,45,453,45
140 DATA 345,45,169,45,453,30
150 DATA 383,15,345,30,262,15
160 DATA 169,60
200 DATA 0,0
```

Denne løkke skaber en kort pause mellem noderne

Dette program spiller noder op og ned ad skalaerne med forskellige hastigheder og viser samtidig nogle farvestriber.

```
10 VOL 8
20 DO
30 D = INT(RND(0)*5)+2: REM VARIGHED
40 S = INT(RND(0)*300)+700: REM START
50 R = INT(RND(0)*(1020-S)): REM OMRÅDE
60 P = INT(RND(0)*30)+5: REM SPRING
70 T = SGN(RND(1)-.5): IF T = 0 THEN 70
80 FOR Z = S TO S+T*R STEP P*T
90 SOUND 1,Z,D
100 Y = (Z AND 15)+1: FOR X = 1 TO D
110 PRINT CHR$(18);: COLOR 1,Y: PRINT" ";
120 NEXT X,Z
130 LOOP
```

Disse remark instrukser hjælper dig med at holde styr på, hvilken linie, der gør hvad.

sæt et mellemrum her

Den store Plus/4 musikmaskine

Det sidste program er lidt længere. Det er den „STORE PLUS/4 MUSIKMASKINE„. Når du trykker en tast fra 1 til og med 9 spilles noden og en node fremkommer på den rigtige linie på nodeplanen.

```
5 GOSUB 1000
6 FOR X = 1 TO 9: READ N(X): NEXT X
8 CHAR 1,8,1,"*DEN STORE MUSIKMASKINE*"
10 VOL 7
20 DO
30 GET A$: IF A$ = "" THEN 30
35 A = ASC(A$): IF A<49 OR A>57 THEN 50
36 N = A-48
40 SOUND 1,N(N),4
45 GSHAPE N$,150,8*(6+(9-N)),4
46 FOR Z = 1 TO 50: NEXT Z
47 GSHAPE N$,150,8*(6+(9-N)),4
50 LOOP UNTIL A = 32
55 VOL 0: GRAPHIC 0: SCNCLR
60 END
100 DATA 345,383,453,516,571,596,643,685,704
1000 GRAPHIC 1,1
1010 FOR Y = 60 TO 124 STEP 16
1020 DRAW 1,100,Y TO 200,Y
1030 NEXT Y
```

intet mellemrum

```
1040 A$ = "FEDCBAGFE"
1050 FOR X = 1 TO 9: C = 13
1060 IF INT(X/2)=X/2 THEN C = 14
1070 CHAR 1, C, X+6, MID$ (A$,X,1),0
1075 CHAR 1, C+10, X+6, RIGHT$ (STR$(10-X),1)
1080 NEXT X
1090 FOR X = 1 TO 8: FOR Y = 11 TO 16: DRAW 1,X,Y: NEXT Y,X
1100 Y = 1: X = 8: DRAW 1,8,16 TO X,Y
1110 SSHAPE N$,1,1,8,16
1120 GSHAPE N$,1,1,4
1130 RETURN
```

Som du ser, er det ikke svært at lave sine egne lydprogrammer. De, der er vist i dette kapitel, er blot en forsmag på, hvad din **C 16-plus/4** evner musikalsk. Du skal ikke være bange for at prøve ny lyd og støj og skabe dit eget mesterstykke.

Commodore C 16-plus/4 Encyklopædi

C 16-plus/4 Encyklopædien indeholder oplysninger, der er nyttige både for begynderen og eksperten på computerområdet. Nogle afsnit er absolut nødvendige for begynderen, f.eks. BASIC 3.5 Encyklopædien, som indeholder og forklarer alle BASIC kommandoer, instrukser og terminologi. Andre afsnit vil være nyttige for den computerbruger, hvis kundskaber går ud over BASIC. TEDMON, **C 16-plus/4**'s maskinsprogs-monitor, giver vejledning i maskinsprogsprogrammering med **C 16-plus/4**. Atter andre afsnit kan være til hjælp for alle ejere af **C 16-plus/4** ... tabeller over fejlmeddelelser, prøveprogrammer, nodetabellen med mere. **C 16-plus/4** Encyklopædien har noget for enhver **C 16-plus/4** programmør, hvad enten det er en begynder eller en øvet programmør.

Basic 3.5 Encyklopædien

Denne håndbog har givet dig en introduktion til BASIC sproget, for at give dig et indtryk af computerprogrammering og den dertil hørende terminologi. Denne Encyklopædi giver en fuldstændig liste over reglerne (SYNTAX) i BASIC 3.5 sproget tillige med en præcis beskrivelse af hver. Prøv at eksperimentere med disse kommandoer og husk, at du ikke kan beskadige din **C 16-plus/4** ved at indtaste programmer og at den bedste måde at lære at bruge en computer på er at øve sig.

Encyklopædien indeholder formater, korte forklaringer og eksempler på BASIC 3.5 kommandoer og instrukser. Det er ikke hensigten, at den skal give undervisning i BASIC.

Kommandoer og instrukser er opført i hvert sit afsnit. Inden for afsnittene er kommandoer og instrukser opført i alfabetisk orden. Kommandoer bruges mest i direkte tilstand, medens instrukser oftest anvendes i programmer. I de fleste tilfælde kan kommandoer bruges som instrukser i et program, hvis man sætter et linienummer foran. Det er muligt at bruge mange instrukser som kommandoer ved at udstede dem i direkte tilstand (d.v.s uden linienummer).

De forskellige typer operationer i BASIC findes opført i afsnit efter følgende kriterier:

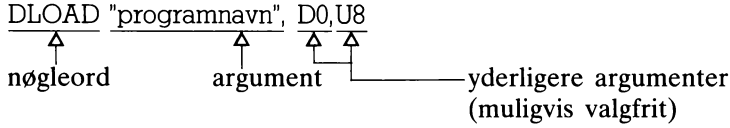
- **KOMMANDOER:** kommandoer der bruges til at arbejde med programmerne, redigere, lagre og slette dem.
- **INSTRUKSER:** BASIC programinstrukserne, der bruges i nummererede programlinier.
- **FUNKTIONER:** strenge, numeriske og udskriftfunktioner.
- **VARIABLER OG OPERATORER:** de forskellige typer variabler, lovlige variabelnavne og aritmetiske og logiske operatorer.

Kommando og instruks format

De kommandoer og instrukser, der præsenteres i dette afsnit af encyklopædien styres af ensartede formatbeskrivelser konstrueret med henblik på at gøre dem så klare som muligt. I fleste tilfælde er der angivet flere praktiske eksempler for at illustrere, hvordan den faktiske kommando ser ud.

Følgende eksempel viser nogle af de formatbeskrivelser, der anvendes i BASIC kommandoer og instrukser:

EKSEMPEL:



○ **NØGLEORD**. også kaldet RESERVEREDE ORD fremstår med store bogstaver. MAN SKAL INDTASTE DISSE NØGLEORD NØJAGTIGT, SOM DE STÅR. Mange nøgleord har imidlertid en forkortet form, man også kan bruge (se afsnit om BASIC forkortelser).

Nøgleord er ord, der udgør en del af det BASIC sprog, din computer forstår. Nøgleord er den centrale del af en kommando eller instruks. De meddeler computeren, hvilken slags handling, du ønsker udført af den. Disse ord kan ikke anvendes som variabel navne.

○ **ARGUMENTER** (også benævnt parametre) fremstår med små bogstaver. Argumenter er de dele af en kommando eller instruks, man vælger; de supplerer nøgleordene ved at give specifik oplysning om kommandoen eller instruksen. Et nøgleord meddeler f.eks. computeren, at den skal indlæse et program, medens et argument meddeler computeren hvilket program, den skal indlæse og et andet argument specificerer det drev, der indeholder disketten med programmet. Argumenter omfatter filnavne, variabler, linienumre o.s.v.

○ **SKARPE PARENTESER** [ ] angiver valgfri argumenter. Man kan vælge et hvilket som helst eller intet af de anførte, alt efter hvilket behov man har.

○ **VINKELPARENTESER** <> angiver, at man SKAL vælge et af de anførte argumenter.

○ **LODRET STREG** | adskiller elementer i en liste over argumenter, når valgmulighederne er begrænset til de anførte argumenter og man ikke kan bruge andre argumenter. Når den lodrette streg forekommer i en liste i SKARPE PARENTESER, er valgmulighederne begrænset til de elementer i listen, men der er stadig den mulighed ikke at anvende nogen argumenter.

○ **ELLIPSER** ..., en sekvens på tre prikker, betyder, at en valgmulighed eller et argument kan gentages mere end én gang.

○ **ANFØRSELSTEGN** " " omgiver karakterstrenge, filnavne og andre udtryk. Når argumenter står i anførselstegn i et format, skal man indsætte anførselstegnene i sin kommando eller instruks. Anførselstegn bruges ikke til at beskrive formater; de er nødvendige bestanddele af en kommando eller instruks.

○ **PARENTESER** ( ) Når argumenter står i parentes i et format, skal man medtage parenteserne i sin kommando eller instruks. Parenteser bruges ikke til at beskrive formater; de er nødvendige bestanddele i en kommando eller instruks.

○ **VARIABEL** betegner ethvert gyldigt BASIC variabelnavn, f.eks. X, A$ eller T%.

○ **UDTRYK** betyder ethvert gyldigt BASIC udtryk som f.eks. A+B+2 eller .5*(X+3).

Basic kommandoer

AUTO

AUTO [linie]

Tilslutter den automatiske linienummereringsfunktion, som letter arbejdet med at indtaste programmer, idet den indtaster linienummeret for dig. Når du indtaster en programlinie og trykker RETURN udskrives næste linienummer på skærmen med markøren i position klar til at påbegynde indtastning af den linie. Linie betegner tilvæksten mellem linienumrene. AUTO uden tilføjelse af ARGUMENT afbryder automatisk linienummerering og det gør RUN også. Denne instruks kan kun udføres i direkte tilstand.

EKSEMPLER:

AUTO 10	nummererer automatisk linierne i spring på ti
AUTO 50	nummererer automatisk linierne i spring på halvtreds

AUTO	AFBRYDER den automatiske linienummerering

BACKUP

BACKUP Ddrev TO Ddrev [, ON Uenhed]

Denne kommando kopierer alle filer på en diskette over på en anden diskette på en dobbelt diskettestation. Man kan kopiere over på en ny diskette uden først at bruge HEADER kommandoen til at formattere den nye diskette, fordi BACKUP kommandoen kopierer al information på disketten, inkl. formatet. Man bør altid BACKUP (kopiere) vigtige disketter for det tilfælde, at originalen går tabt eller bliver beskadiget. Da BACKUP kommandoen også formatterer disketter, ødelægger den eventuel information på den diskette, på hvilken man kopierer information. Hvis man derfor kopierer over på en tidligere brugt diskette, skal man sikre sig, at den ikke indeholder programmer, man ønsker at gemme. Se også COPY kommandoen.

BEMÆRK: Denne kommando kan kun anvendes med en dobbelt diskettestation.

EKSEMPLAR:

BACKUP D0 TO D1	Kopierer alle filer fra disketten i drev 0 til disketten i drev 1
BACKUP D0 TO D1, ON U9	Kopierer alle filer fra drev 0 til drev 1 i diskettestation 9

COLLECT

COLLECT [Ddrev] [,ON Uenhed]

Brug denne kommando til frigørelse af plads tildelt fejlagtigt lukkede filer og sletning af henvisninger til disse filer fra adresselisten.

EKSEMPEL:

COLLECT D0

CONT (Continue)

Denne kommando bruges til pånyat starte udførelsen af et program, der har været standset, enten ved brug af STOPTASTEN, en STOP-instruks eller en END-instruks i programmet. Programmet vil fortsætte fra det sted, hvor det slap. CONT vil ikke fungere, hvis man har ændret eller tilføjet linier i programmet (eller endda bare flyttet markøren til en programlinie og trykket RETURN uden at ændre noget), hvis programmet stoppede på grund af en fejl, eller hvis man har begået en fejl, før man forsøger at starte programmet pånyt. Fejlmeddelelsen vil i så tilfælde være CAN'T CONTINUE ERROR.

COPY

COPY [Ddrev,] "kildefil" TO [Ddrev,] "anden fil" [,ON Uenhed]

Kopierer en fil fra disketten i et drev (kildefilen) til disketten i det andet, men kun i en anden dobbelt diskettestation, eller laver en kopi af en fil i samme drev (med et andet filmnavn).

EKSEMPLER:

COPY D0, "PRØVE" TO D1, "PRØVE PROG"	Kopierer "PRØVE" fra drev 0 til drev 1 og giver den nyt navn "PRØVE PROG" i drev 1.
COPY D0 "MATERIALE" TO D1, "MATERIALE"	Kopierer "MATERIALE" fra drev 0 til drev 1.
COPY D0 TO D1	Kopierer alle filer fra drev 0 til drev 1.
COPY "ARBEJDSSPROG" TO "KOPIER"	Kopierer "ARBEJDSPROG" som et program betegnet "KOPIER" i samme drev.

DELETE [fra linie] [−[til linie]]

Sletter BASIC tekstlinier. Denne kommando kan kun udføres i direkte tilstand.

EKSEMPLER:

DELETE 75	Sletter linie 75.
DELETE 10–50	Sletter linierne 10 til og med 50.
DELETE –50	Sletter alle linier fra begyndelsen af programmet til og med linie 50.
DELETE 75-	Sletter alle linier fra 75 til programmets slutning.

DIRECTORY

DIRECTORY [Ddrev] [,Uenhed] [,"filnavn [=<PRG | SEQ | REL>]"]

Viser en diskettes indholdsfortegnelse på skærmen til **C 16-plus/4**. Brug CONTROL-S for at gøre pause i visningen (enhver anden tast genoptager visningen efter en pause). Brug <C=>-tasten (Commodore tasten) til at sænke hastigheden. DIRECTORY kommandoen kan ikke bruges til udskrift. Man skal indlæse diskette indholdsfortegnelsen (programmet der befinder sig i hukommelsen ødelægges) for at få en udskrift.

EKSEMPLER:

DIRECTORY	Lister alle filer på disketten.
DIRECTORY D1, U9, "arbejde"	Lister filen i disketteenhed 9 (8 er defaultværdien, drev 1, betegnet "arbejde"
DIRECTORY "AB*"	Lister alle filer, der begynder med bogstaverne "AB", f.eks. ABOVE, ABOARD, o.s.v.
DIRECTORY D0, "FIL ?.BAK"	Spørgsmålstegnet er et søgetegn, der passer til enhver enkeltkarakter, på den position: fil 1.BAK, FIL 2.BAK, FIL 3.BAK alle passer til strengen.
DIRECTORY "*=prg"	Lister alle programfiler på en diskette.

DIRECTORY "*=seq"	Lister alle sekventielle filer på en diskette.
DIRECTORY "MI*=prg"	Lister alle programmer, der begynder med bogstaverne MI, fra en diskette.
DIRECTORY "MI*=seq"	Lister alle sekventielle filer, der begynder med bogstaverne MI, fra en diskette.

BEMÆRK: For at udskrive indholdsfortegnelsen for drev 0 enhed 8 bruges følgende:

```
LOAD "$0",8
OPEN4,4:CMD4:LIST
PRINT#4:CLOSE4
```

Dload

DLOAD "filnavn" [,Ddrev] [,Uenhed]

Denne kommando indlæser et program fra diskette til arbejdshukommelsen. (Brug LOAD til indlæsning af programmer fra bånd). Man skal tilføje et programnavn.

EKSEMPLER:

DLOAD "BUDGET"	Søger på disketten efter "BUDGET" og indlæser det.
DLOAD (A$)	Indlæser et program fra diskette, hvis navn ligger i variablen A$. Man får en fejl, hvis A$ er tom.

DLOAD kommandoen kan anvendes i et BASIC program for at finde og KØRE et andet program på diskette. Dette kaldes sammenkædning af programmer.

DSAVE

DSAVE "filnavn" [,Ddrev] [,Uenhed]

Denne kommando lagrer et program på diskette. (Brug SAVE til lagring af programmer på bånd). Man skal tilføje et programnavn.

EKSEMPLER:

DSAVE "BUDGET"	Lagrer programmet "BUDGET" på diskette.
DSAVE (A$)	Lagrer program, hvis navn ligger i variablen A$.
DSAVE "PROG 3", D0,U9	Lagrer programmet "PROG 3" på diskettedrev med et enhedsnummer 9.

HEADER

HEADER "diskettenavn" [,Iid] [,Ddrev] [,ON Uenhed]

Før en ny diskette kan tages i brug første gang, skal den formatteres ved hjælp af HEADER kommandoen. Hvis du vil slette en hel diskette for at bruge den igen, kan du bruge HEADER kommandoen. Denne kommando deler disketten i afsnit kaldet blokke og laver en indholdsfortegnelse (directory). Diskettenavnet kan være et hvilket som helst navn med op til 16 karakterer. I.d.nummeret er 2 vilkårlige karakterer. Giv hver diskette et selvstændigt i.d.nummer. Vær omhyggelig når du FORMATTERER en diskette, for HEADER kommandoen sletter alle lagrede data. Hvis du ikke bruger i.d.nummer, kan du lave en hurtig formattering. Det gamle i.d.nummer bruges. Du kan kun bruge den hurtige formatteringsmetode, hvis disketten har været formatteret tidligere, da hurtig formattering kun sletter indholdsfortegnelsen og ikke formatterer disketten.

EKSEMPLER:

HEADER "MINDISKETTE", D0
HEADER "POSTER", I45, D1, U8

HELP

HELP

HELP kommandoen bruges, efter at man har fået en fejl i sit program. Når man indtaster HELP listes den linie, hvori fejlen forekommer, med den del, hvor fejlen forekommer vist med blinkende karakterer.

KEY

KEY [funktionstastnr, streng]

Der står otte (8) funktionstaster til brugerens disposition på **C 16-plus/4** computeren: fire som bruges uden hjælp af skiftetasten og fire som bruges i kombination med skiftetasten. **C 16-plus/4** lader dig bestemme, hvad hver enkelt tast udfører, når den trykkes.
KEY uden angiven parameter giver en listning, der viser alle eksisterende funktionstast-funktioner. De data, du tildeler en tast, udskrives, når denne funktionstast trykkes. Maksimumlængden på alle definitionerne tilsammen er 128 karakterer. Hele kommandoer (eller en serie kommandoer) kan tildeles en tast. For eksempel får:

```
KEY 7, "GRAPHIC0"+CHR$(13)+"LIST"+CHR$(13)
```

computeren til at vælge teksttilstand og liste dit program, når som helst "F7" tasten trykkes (i direkte tilstand). CHR$(13) er ASCII karakteren for RETURN. Brug CHR$(34) til indsætning af anførselstegn i en KEY-streng.

Tasterne kan omdefineres i et program. F.eks. definerer:

```
10 KEY 2, "AFPRØVNING"+CHR$(34): KEY 3, "NEJ"
```

```
10 FOR I = 1 TO 8:KEY I, CHR$(I+132):NEXT
```

funktionstasterne således, som de er defineret på Commodore 64 og VIC 20.

For at bringe alle funktionstasterne tilbage til deres defaultværdier omstilles **C 16-plus/4** ved at tænde og slukke for den eller ved at trykke RESET knappen.

LIST

LIST [fra linie] [- [til linie]]

LIST kommandoen lader dig se BASIC programlinier, der er blevet indtastet eller indlæst i **C 16-plus/4**'s hukommelse. Når den anvendes alene (uden påfølgende numre) giver **C 16-plus/4** en fuldstændig listning af programmet på skærmen. Hastigheden kan nedsættes ved at holde C= tasten nede, listningen kan afbrydes ved hjælp af CONTROL-S (igangsættes

igen ved tryk på en hvilken som helst anden tast), eller standses ved tryk på RUN/STOP tasten. Hvis ordet LIST efterfølges af et linienummer, viser **C 16-plus/4** kun det linienummer. Hvis du indtaster LIST med to numre adskilt af et minus, viser **C 16-plus/4** alle linierne mellem det første og det andet linienummer. Hvis man indtaster LIST efterfulgt af et nummer og et minus, viser den alle linierne fra det nummer til slutningen af prggrammet. Og hvis du indtaster LIST, et minus og et nummer, får du alle linierne fra begyndelsen af programmet og frem til dette linienummer. Ved brug åf disse variationer kan du undersøge enhver del af et program og let fremkalde linierne på skærmen for at foretage ændringer.

EKSEMPLER:

LIST	Viser hele programmet.
LIST 100-	Viser fra linie 100 til slutningen af programmet.
LIST 10	Viser kun linie 10.
LIST -100	Viser linierne fra begyndelsen til linie 100.
LIST 10-200	Viser linierne fra 10 til og med 200

LOAD

LOAD "filnavn" [,enhed] [,relocate flag]

Dette er den kommando, der skal bruges, når man vil indlæse et program, der er lagret på kassettebånd eller diskette. Hvis man bare indtaster LOAD og trykker RETURN tasten, bliver skærmen på **C 16-plus/4** tom. Tryk play og **C 16-plus/4** begynder at lede efter et program på båndet. Når den finder et, udskriver **C 16-plus/4** FOUND "filnavnet". Man kan trykke C= tasten for indlæsning eller mellemrumstasten for fortsat søgning på bånden. Når først programmet er indlæst, kan man køre, liste eller ændre det.

Man kan også indtaste ordet LOAD efterfulgt af et programnavn, som oftest er et navn i anførselstegn ("programnavn"). Navnet kan følges af et komma (uden for anførselstegnene) og et nummer eller (numerisk variabel), der fungerer som nummer på den eksterne enhed for at angive, hvor programmet er lagret (diskette eller bånd). Hvis der ikke er noget nummer, formoder **C 16-plus/4**, at der er tale om ekstern enhed nr. 1, som er

kassettebåndoptageren. Den anden eksterne enhed, der almindeligvis bruges med LOAD kommandoen, er oftest diskettestationen, som er ekstern enhed nr. 8.

EKSEMPLER:

LOAD	Indlæser det næste program på båndet.
LOAD "HALLO"	Søger på båndet efter et program kaldet "HALLO" og indlæser det, hvis det findes.
LOAD A$	Leder efter et program hvis navn ligger i variablen kaldet A$.
LOAD "HALLO",8	Søger efter programmet, der kaldes HALLO i diskettestationen.

LOAD kommandoen kan anvendes i et BASIC program for at finde og køre det næste program på et bånd. Dette kaldes sammenkædning.

RELOCATE FLAG bestemmer, hvor i hukommelsen et program indlæses. Relocate flag 0 instruerer **C 16-plus/4** om at indlæse programmet i begyndelsen af BASIC programområdet, og et flag på 1 instruerer den om at indlæse fra det sted, hvor det blev lagret. Defaultværdien af relocate flag er 0. Dette bruges i almindelighed kun, når der indlæses programmer i maskinsprog.

NEW

NEW

Denne kommando sletter hele programmet i hukommelsen og fjerner eventuelle variabler, der måtte være brugt. Medmindre programmet var lagret et sted, er det nu gået tabt, indtil det atter indtastes. Pas derfor på, når denne kommando anvendes.

NEW kommandoen kan også bruges som en instruks i et BASIC program. Når **C 16-plus/4** når til denne linie, slettes programmet og alting standser. Dette er under normale omstændigheder ikke særligt nyttigt.

RENAME

RENAME "gammelt navn" TO "nyt navn" ,Ddrev ,Uenhed

Bruges til at give en fil på en diskette nyt navn.

EKSEMPEL:

RENAME "PRØVE" TO "AFSLUTTENDE PRØVE", D0 Ændrer navnet på filen fra "PRØVE" til "AFSLUTTENDE PRØVE".

RENUMBER

RENUMBER [ny startlinienr. [,tilvækst [,gammelt startlinienr.]]]

Den nye startlinie er nummeret på den første linie i programmet efter omnummerering. Default er 10.

Tilvæksten er springet mellem linienumrene, d.v.s. 10, 20, 30 o.s.v. Default er 10.

Det gamle startlinienummer er det linienummer i programmet, hvor omnummereringen skal begynde. Dette muliggør omnummerering af en del af dit program. Default er første linie af dit program.

Denne kommando kan kun udføres i direkte tilstand.

EKSEMPLER:

RENUMBER 20, 20, 1	Omnummererer programmet med start ved linie 1. Linie 1 bliver til linie 20 og de andre linier nummereres i spring på 20.
RENUMBER, ,65	Med start ved linie 65 omnummereres i spring på 10. Linie 65 bliver til linie 10.

RUN

RUN [linie]

Når først et program er indtastet i hukommelsen eller indlæst, får RUN kommandoen det til at begynde at fungere. RUN kommandoen nulstiller alle variabler i programmet, før den starter udførelsen af programmet. Hvis der ikke følger noget nummer efter kommandoen RUN, starter computeren med den programlinie, der har det laveste nummer. Hvis der føl-

ger et nummer efter RUN kommandoen, starter udførelsen ved den linie. RUN kommandoen kan anvendes i et program.

EKSEMPLER:

RUN	Starter programmet fra linien med det laveste nummer.
RUN 1000	Starter programmet ved linie 100.

SAVE

SAVE ["filnavn" [,enhed[,EOT flag]]]

Denne kommando lagrer et program, der befinder sig i arbejdshukommelsen over på et kassettebånd eller en diskette. Hvis du blot indtaster ordet SAVE og trykker RETURN, forsøger **C 16-plus/4** at lagre programmet på kassettebånd. Den har ingen mulighed for at kontrollere, om der allerede befinder sig et program på det sted på båndet, så vær forsigtig med dine bånd. Hvis du indtaster SAVE kommandoen efterfulgt af et navn i anførselstegn eller en strengvariabel, giver **C 16-plus/4** programmet dette navn, således at det lettere kan findes og fremkaldes senere. Hvis du vil specificere et nummer på en ekstern enhed til LAGRING, skal du sætte et komma efter navnet (efter anførselstegnene) samt et nummer eller en numerisk variabel. Den eksterne enhed med nr. 1 er båndoptageren og nr. 8 er diskettestationen. Efter nummeret på en båndkommando, kan der være et komma og endnu et nummer, som enten er 0 eller 1. Hvis det andet nummer er 1, sætter **C 16-plus/4** et END-OF-TAPE (slut på bånd) mærke (EOT flag) efter programmet. Hvis du prøver at INDLÆSE et program og **C 16-plus/4** finder et af disse mærker i stedet for det program, du prøver at INDLÆSE får du en FILE NOT FOUND ERROR (filen ikke fundet fejl).

EKSEMPLER:

SAVE	Lagrer programmet på bånd uden navn.
SAVE "HALLO"	Lagrer på bånd under navnet HALLO
SAVE A$	Lagrer på bånd under navnet i variabel A$
SAVE "HALLO",8	Lagrer på diskette under navnet HALLO

SAVE "HALLO",1,1	Lagrer på bånd under navnet HALLO og sætter et END-OF-TAPE mærke efter programmet.

SCRATCH

SCRATCH "filnavn", [Ddrev] [,Uenhed]

Sletter en fil fra diskette indholdsfortegnelse. For en sikkerheds skyld spørger **C 16-plus/4** "Are you sure" (er du sikker), før den udfører funktionen. Indtast et Y til gennemførelse af SCRATCH kommandoen eller indtast et N, der annullerer funktionen. Brug denne kommando til sletning af uønskede filer, for at gøre bedre plads på disketten.

EKSEMPEL:

SCRATCH "MINFIL", D$	Sletter filen MINFIL fra disketten i drev 1

VERIFY

VERIFY "filnavn" [,enhed] [,relocate flag]

Denne kommando får **C 16-plus/4** til at sammenholde programmet på bånd eller diskette med det, der er i hukommelsen. Dette er bevis på, at det program, du lige lagrede, virkelig er lagret, dersom der er noget i vejen med dit bånd, eller hvis der er noget, der ikke fungerer. Denne kommando er også nyttig til mærkning af et bånd, således at **C 16-plus/4** skriver efter det sidste program på båndet. Du behøver bare at bede **C 16-plus/4**VERIFY (verificere) navnet på det sidste program på båndet. Det vil den gøre og så meddele dig, at programmerne ikke svarer til hinanden (det vidste du i forvejen). Nu er båndet, hvor du vil have, det skal være og du kan lagre det næste program uden at være bange for at komme til at slette et gammelt. VERIFY uden tilføjelser efter kommandoen får **C 16-plus/4**til at sammenholde det næste program på båndet, uanset navnet, med det program, der befinder sig i hukommelsen. VERIFY efterfulgt af et programnavn (i anførselstegn) eller en strengvariabel søger på båndet efter dette program og sammenholder så. VERIFY efterfulgt af et navn og et komma og et nummer kontrollerer programmet i den eksterne enhed med det pågældende nummer (1 for bånd, 8 for diskette). Relocate flag er det samme som i LOAD kommandoen.

EKSEMPLER:

VERIFY	Kontrollerer det næste program på båndet.
VERIFY "HALLO"	Søger efter HALLO på båndet og sammenholder med hukommelsen.
VERIFY "HALLO",8,1	Søger efter HALLO på disketten, og kontrollerer derefter.

Basic instrukser

BOX

BOX [farvekilde,] a1, b1, [a2, b2,] [,vinkel] [,farvelæg]

farvekilde	Farvekilde (0-3); default er 1 (forgrundsfarve)
a1, b1	Hjørne koordinat (skaleret)
a2, b2	Hjørne modsat a1, b1 (skaleret); default er PC
vinkel.....	Rotation i grader med uret; default er 0 grader
farvelæg.....	Farvelæg figuren (0:off, 1:on); default er 0

Denne kommando lader dig tegne et rektangel af en hvilken som helst størrelse på skærmen. For at få default værdien, skal man indsætte et komma uden at indsætte en værdi. Rotation er baseret på rektanglets centrum. Punkt Cursoren (PC) står ved a2, b2 efter at BOX instruksen er udført.

EKSEMPLER:

BOX 1, 10, 10, 60, 60	Tegner omridset af en rektangel
BOX , 10, 10, 60, 60, 45, 1	Tegner en fyldt, drejet firkant (en rhombe)
BOX , 30, 90, , 45, 1	Tegner en fyldt, drejet polygon

CHAR

CHAR [farvekilde], x, y [,streng] [,reverse]

farvekilde	Farvekilde (0-3)
x.....	Karakterkolonne (0-39)
y.....	Karakterrække (0-24)
STRING.....	Streng til udskrivning
reverse.....	Negativ skrift (0 = off, 1 = on)

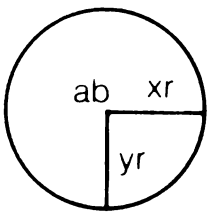
Tekst (alfanumeriske strenge) kan vises på et givet sted på enhver skærm ved hjælp af CHAR kommandoen. Karakterdata læses fra **C 16-plus/4**'s karakter ROM (read-only memory) område. Man tilføjer x og y koordinaterne for startpositionen og den tekststreng, man vil have frem. Farvekilde og negativ visning er valgfri.

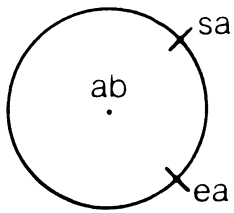
Strengen fortsættes på næste linie, hvis den strækker sig forbi højre skærmrand. Når man er i teksttilstand, fungerer strengen der udskrives af CHAR kommandoen præcis som en PRINT streng inklusive negativ skrift flash on/off o.s.v. Disse styrefunktioner i strengen fungerer ikke, når CHAR kommandoen bruges til visning af tekst i GRAFIK tilstand.
BEMÆRK: Når man i flerfarvetilstand skal vise en karakter i flerfarve 2, sættes farvekilden til 0 og negativ flag til 1. Til visning af en karakter i flerfarve 1, sættes farvekilden til 0 og negativ flag til 0.

CIRCLE

CIRCLE [fk] , [a,b] ,xr [, [yr] [, [sa] [, [ea] [, [vinkel] [,grader]]]]]

cs	Farvekilde (0-3)
a,b	Centrumskoordinat (skaleret) (default er Punkt Cursoren PC)
xr	X radius (skaleret)
yr	Y radius (default er xr)
sa	Start på buevinkel (default 0)
ea	Slut på buevinkel (default 360)
vinkel.....	Rotation i grader med uret (default er 0 grader)
grader	Grader mellem cirkelafsnit (default er 2 grader)





Ved hjælp af CIRCLE kommandoen kan man tegne en cirkel, ellipse, bue, trekant eller ottekant. Den sidste koordinat er på cirklens omkreds ved den afsluttende buevinkel. Eventuel rotation sker omkring centrum. Sætter man Y radius lig med X radius tegnes ikke en cirkel, da X og Y koordinaterne er forskelligt skaleret. Buen tegnes fra startvinklen med uret til slutvinklen. Afsnittilvæksten styrer figurens udseende, idet lavere inc. (tilvækst) værdien giver rundere figurer.

EKSEMPLER:

CIRCLE, 160,100,65,10	Tegner en ellipse.
CIRCLE, 160,100,65,50	Tegner en cirkel.
CIRCLE, 60,40,20,18,,,,45	Tegner en ottekant.
CIRCLE, 260,40,20,,,,,90	Tegner en rhombe.
CIRCLE, 60,140,20,18,,,,120	Tegner en trekant.

CLOSE

CLOSE filnummer

Denne kommando færdiggør og lukker eventuelle filer, brugt med OPEN instrukser. Det nummer der følger efter ordet CLOSE er nummeret på den fil, der skal lukkes.

EKSEMPEL:

CLOSE 2	Logisk fil 2 lukkes.

CLR

CLR

Denne kommando sletter eventuelle variabler i hukommelsen, men lader selve programmet intakt. Denne kommando udføres automatisk, når en RUN eller NEW kommando gives eller ved udførelse af redigering.

CMD

CMD filnummer [,liste af variabler]

CMD sender uddata, som normalt ville komme til skærmen (d.v.s. Printinstruks, LISTs, men ikke POKEs ind i skærmen), til en anden enhed i stedet. Det kunne være en skriver eller en datafil på bånd eller diskette. Denne enhed eller fil må først åbnes. CMD kommandoen skal følges af et nummer eller en numerisk variabel, der refererer til filen.

EKSEMPLER:

OPEN 1,4	Åbner enhed 4, som er skriveren.
CMD 1	Alle normale uddata går nu til skriveren.

LIST	Listningen går til skriveren, ikke skærmen – selv ordet READY
PRINT #1	Sætter uddata tilbage til skærmen.
CLOSE 1	Lukker filen.

COLOR

COLOR farvekilde, farve [,lysstyrke]

Tildeler en af de 5 farvekilder en farve:

Nummer	Kilde
0	baggrund
1	forgrund
2	flerfarve 1
3	flerfarve 2
4	rand

De farver du kan bruge ligger i området 1-16 (SORT, HVID ...). Som valgmulighed kan man indføje lysstyrker 0-7, med 0 som det laveste og 7 som det højeste. Lysstyrker default er 7. Lysstyrker gør det muligt for dig at vælge 8 lysstyrker for enhver farve undtagen sort.

DATA

DATA liste af konstanter adskilt af kommaer

Denne instruks følges af en liste af faktorer, der skal anvendes ved READ instrukser. Faktorerne kan være tal eller ord og de er adskilt af kommaer. Ord behøver ikke at stå i anførselstegn, medmindre de indeholder en eller flere af følgende karakterer: MELLEMRUM, kolon eller komma. Hvis der mellem to kommaer ikke står noget, vil værdien blive læst som et nul for et tal eller en tom streng. Se også RESTORE instruksen, som tillader **C 16-plus/4** at omlæse data.

EKSEMPLER:

DATA 100, 200, FRED, "HALLOMOR", , 3, 14, ABC123

DEF FN

DEF FN navn (variabel) = udtryk

Denne kommando gør det muligt for dig at definere en kompliceret beregning som en funktion. Hvor der er tale om en lang formel, der anvendes flere gange i et program, kan dette spare en masse plads. Det navn man giver funktionen begynder med bogstaverne FN efterfulgt af et hvilket som helst lovligt numerisk variabelnavn. Først skal du definere funktionen ved brug af instruksen DEF efterfulgt af det navn, du har givet funktionen. Efter navnet følger et sæt parenteser () med en numerisk variabel (i dette tilfælde X) indeni. Så følger et lighedstegn efterfulgt af den formel, du vil definere. Du kan "kalde" formlen ved at indsætte et vilkårligt nummer i stedet for X og bruge formatet, der er vist i linie 20 i nedstående eksempel:

EKSEMPEL:

10 DEF FNA(X) = 12*(34.75−X/.3)+X	Tallet 7 indsættes hvert sted.
20 PRINT FNA(7)	X findes i formlen, der er opgivet i DEF instruksen.

DIM

DIM variabel (elementnummer) [,variabel (elementnummer) ...]

Før du kan bruge et sæt variabler, skal programmet udføre en DIM instruks for at bestemme sættets dimensioner (medmindre der er 11 eller færre elementer i sættet). Instruksen DIM følges af navnet på sættet, som kan være et hvilket som helst lovligt variabelnavn. Så indsætter du en parentes med antallet (eller den numeriske variabel) på elementerne i hver dimension. Et sæt med mere end en dimension kaldes en matrix. Du kan anvende et hvilket som helst antal dimensioner, men husk at hele den liste af variabler, du laver, fylder i hukommelsen, og du kan nemt bruge hele hukommelsen, hvis du ikke passer på. For at udregne antallet af variabler, der laves med hver DIM, ganger du de samlede tal for hver enkelt af sættets dimensioners indhold af elementer med hinanden. (Hvert sæt begynder med elementet 0).

BEMÆRK: Integer (heltal) sæt 2 bytes per element. Decimaltal (flydende komma) sæt optager 5 bytes per element.

EKSEMPEL:

```
10 DIM A$(40),B7(15),CC%(4,4,4)
       ↑       ↑        ↑
41 elementer   16 elementer   125 elementer
```

Du kan dimensionere mere end et sæt i en DIM instruks ved at adskille instrukserne med kommaer. Hvis programmet udfører en DIM instruks for et sæt mere end én gang, får du en re'DIMed (gendimensioneret) sæt fejlmelding. Det er god programmeringspraksis at sætte DIM instrukser i begyndelsen af programmet.

DO | LOOP | WHILE | UNTIL | EXIT

DO [UNTIL betingelse | WHILE betingelse] [instrukser] [EXIT]

LOOP [UNTIL betingelse | WHILE betingelse]

Udfører instrukserne mellem DO instruksen og LOOP instruksen. Hvis der ikke er noget UNTIL eller WHILE til at ændre enten DO eller LOOP instruksen, fortsætter udførelsen af de mellemliggende instrukser i det uendelige. Hvis der dukker en EXIT instruks op i en DO løkke, overføres udførelsen til den første instruks efter LOOP instruksen. DO løkker kan parentesstruktureres i overensstemmelse med reglerne fastsat for FOR-NEXT løkker. Hvis UNTIL parameteret anvendes, fortsætter programmet med løkken, indtil betingelsen opfyldes. WHILE parameteren er i grunden det modsatte af UNTIL parameteren: programmet fortsætter med løkken, så længe betingelsen opfyldes. Eksempler på betingelser er A = 1, eller C> = 65.

EKSEMPEL:

```
DO UNTIL X = 0 OR X = 1
    :
    LOOP

DO WHILE A$ = " ":GET A$:LOOP
```

DRAW

DRAW [farvekilde,] [a1, b1] [TO a2, b2 ...]

Med denne kommando kan du tegne individuelle prikker, streger og figurer. Du indsætter farvekilde (0-3), start (a1, b1) og endepunkter (a2, b2).

EKSEMPLER:

en prik:	DRAW 1, 100, 50	intet endepunkt anført, default er a1, b1 værdi for a2, b2 for at lave en prik
streger:	DRAW , 10,10 TO 100,60	
	DRAW TO 25,30	
en figur:	DRAW , 10,10 TO 10,60 TO 100,60 TO 10,10	

END

Når programmet udfører en END instruks, holder programmet straks op med at køre. Man kan bruge CONT kommandoen til fornyet start af programmet fra den instruks, der følger efter END instruksen.

FOR ... TO ... STEP

FOR variabel = startværdi TO slutværdi [STEP tilvækst]

Denne instruks bevirker sammen med NEXT instruksen, at der sættes et afsnit af programmet, der gentages et fastsat antal gange. FOR-TO-STEP kan bruges, hvis **C 16-plus/4** skal tælle op til et højt tal, således at programmet hviler nogle få sekunder, hvis du vil have noget talt, eller hvis der skal udføres noget et vist antal gange (som f.eks. udskrivning).
Loop variablen er den variabel der lægges til eller trækkes fra i løbet af FOR/NEXT løkken. Startværdien og slutværdien er første og sidste nummer for løkke variablen.
FOR instruksens logik er følgende. Først sættes løkkevariablen på startværdien. Når programmet når en linie med kommandoen NEXT, føjer det STEP tilvæksten (default = 1) til værdien af løkkevariablen og kontrollerer, om den er højere end værdien ved afslutningen af løkken. Hvis den ikke er højere, bliver den næste linie der udføres instruksen der følger umiddelbart efter FOR instruksen. Hvis løkkevariablen er større end tallet ved afslutningen af løkken, bliver den næste instruks, der udføres, den der følger efter NEXT instruksen. Se også NEXT instruksen.

EKSEMPEL:

```
10 FOR L = 1 TO 10
20 PRINT L
30 NEXT
40 PRINT "JEG ER FÆRDIG! L = "L
```

Dette program udskriver tallene fra et til ti på skærmen, fulgt af meddelelsen JEG ER FÆRDIG! L = 11.

Slutværdien for løkken følges måske af ordet STEP og et andet tal eller en anden variabel. Hvis det er tilfældet, tillægges den værdi, der følger efter STEP, hver gang i stedet for én. Dette gør det muligt for dig at tælle baglæns, i brøker eller på en hvilken som helst måde, det er nødvendigt.

Man kan sætte løkker inden i hinanden. Dette kaldes parentesstrukturering af løkker. Man må være forsigtig og parentesstrukturere løkkerne således, at den sidste løkke der begynder, er den første der slutter.

EKSEMPEL PÅ PARENTESSTRUKTUREREDE LØKKER:

```
10 FOR L = 1 TO 100

20 FOR A = 5 TO 11 STEP 2      Denne FOR ... NEXT løkke "parentes-
                               struktureres" inden i den større

30 NEXT A
40 NEXT L
```

GET

GET strengvariabel

GET instruksen er en måde at få data fra tastaturet på, en karakter ad gangen. Når GET udføres, modtages den karakter, der blev indtastet. Hvis der ikke blev indtastet nogen karakter, får man en nulkarakter (tom) og programmet fortsætter uden at vente på en karakter. Det er ikke nødvendigt at trykke RETURN tasten, og faktisk kan RETURN tasten fås med en GET instruks.
Ordet GET følges af et variabelnavn, sædvanligvis en strengvariabel. Hvis en numerisk variabel blev anvendt og en hvilken som helst anden tast end et tal blev trykket, ville programmet standse med en fejlmeddelelse. GET instruksen kan også anbringes i en løkke, til kontrol for tomt resultat, som venter med at fortsætte indtil en tast trykkes. GETKEY instruksen kan også anvendes i dette tilkælde. Denne kommando kan kun udføres i et program.

EKSEMPEL:

```
10 GET A$:IF A$ <> "A" THEN 10  Denne linie venter på, at A tasten
                                trykkes, før den fortsætter.
```

GETKEY

GETKEY strengvariabel

GETKEY instruksen er meget lig GET instruksen. Til forskel fra GET instruksen, venter GETKEY på, at brugeren taster en karakter på tastaturet. Dette gør, at den let kan bruges til at vente på, at enkeltkarakterer tastes.
Denne kommando kan kun udføres i et program.

EKSEMPEL:

10 GETKEY A$	Denne linie venter på, at en tast trykkes. Et hvilket som helst anslag vil fortsætte programmet.

GET

GET# filnummer, liste af variabler

Anvendes med en tidligere åbnet enhed eller fil til indlæsning af én karakter ad gangen. Ellers fungerer den som GET instruksen.
Denne kommando kan kun udføres i et program.

EKSEMPEL:

```
GET#1, A$
```

GOSUB

GOSUB linie

Denne instruks ligner GOTO instruksen, med den forskel, at **C 16-plus/4** husker, hvor den kom fra. Når der forekommer en linie med en RETURN instruks, springer programmet straks tilbage til den instruks, der fulgte umiddelbart efter GOSUB instruksen. Målet for en GOSUB instruks kaldes en underrutine. En underrutine er nyttig, hvis der er en rutine i dit program, der kan bruges af flere forskellige dele af programmet. I stedet for at duplikere det programafsnit igen og igen, kan du sætte det op som en underrutine og kalde det fra de forskellige dele af programmet med GOSUB. Se også RETURN instruksen.

EKSEMPEL:

```
20 GOSUB 800
      :
      :
800 PRINT "GO DAW":RETURN
```

betyder gå til underrutinen, der begynder ved linie 800 og udfør den

GOTO eller GO TO

GOTO line

Efter en GOTO instruks er udført, vil næste linie, der skal udføres være den med det nummer, der følger efter ordet GOTO. Når den anvendes i direkte tilstand muliggør GOTO line instruksen, at du kan starte udførelsen af programmet ved et givet linienummer uden at slette variablerne.

EKSEMPEL:

```
10 PRINT"COMMODORE"
20 GOTO 10
```

GOTO instruksen i linie 20 gør, at linie 10 gentages uafbrudt indtil du trykker RUN/STOP

GRAPHIC

GRAPHIC <tilstand [,slet] | CLR>

Denne instruks sætter **C 16-plus/4** i en af dens 5 grafiske tilstande:

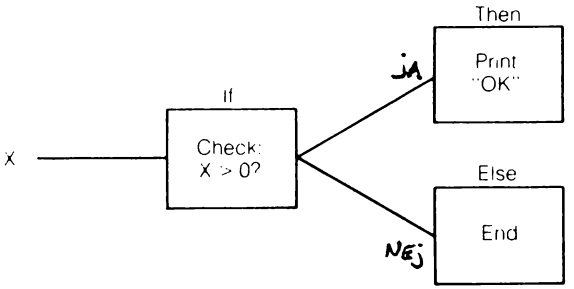
tilstand	beskrivelse
0	normal tekst
1	højopløsningsgrafik
2	højopløsningsgrafik, delt skærm
3	flerfarvegrafik
4	flerfarvegrafik, delt skærm

Når de udføres, reserverer GRAFIK tilstandene 1-4 et 10K bit-mapped område og starten på BASIC tekst området flyttes oven over højopløsningsområdet. Dette område forbliver reserveret, selv om brugeren går tilbage til TEKST tilstand (GRAPHIC 0). Hvis 1 anføres i GRAPHIC instruksen som det andet argument, slettes skærmbilledet også. Udførelse af en GRAPHIC CLR kommando fjerner reservationen af det 10K bit-mapped område og gør det igen tilgængeligt for BASIC tekst og variabler.

IF ... THEN [..:ELSE]

IF udtryk THEN instrukser [:ELSE instrukser]

IF ... THEN lader computeren analysere et BASIC udtryk, der står umiddelbart efter IF, og vælge et af to mulige handlingsforløb. Hvis udtrykket er korrekt, udføres instruksen, der følger efter THEN. Dette udtryk kan være en hvilken som helst BASIC instruks. Hvis udtrykket er forkert, går programmet straks til næste linie, medmindre der er en ELSE sætning. Det udtryk der vurderes kan være en variabel eller en formel, i hvilket tilfælde det betragtes som korrekt, hvis det ikke er nul og forkert, hvis det er nul. I de fleste tilfælde er der et udtryk, der indeholder relationelle operatorer (=, <, >, <=, >=, <>, AND, OR, NOT).
Hvis der er en ELSE sætning, skal den være på samme linie som IF - THEN delen. Når der er en ELSE sætning, udføres den når THEN sætningen ikke udføres. Med andre ord, ELSE sætningen udføres, når IF udtrykket er forkert.



EKSEMPEL:

50 IF X>0 THEN PRINT "OK":ELSE END

Undersøger værdien af X. Hvis X er større end 0, udføres THEN instruksen, og ELSE instruksen udføres ikke. Hvis X er mindre end 0, udføres ELSE instruksen og THEN instruksen udføres ikke.

INPUT ["meddelelse";] liste af variabler

INPUT instruksen lader computeren bede om data fra den person, der kører programmet og anbringe dem i en eller flere variabler. Programmet standser, udskriver et spørgsmålstegn (?) på skærmen og afventer, at brugeren indtaster svaret og trykker RETURN tasten.
Ordet INPUT følges af et variabelnavn eller en liste variabelnavne adskilt af kommaer. Der kan være en meddelelse i anførselstegn før listen af variabler, der skal indlæses. Hvis denne meddelelse forekommer, skal der være et semikolon (;) efter det anførselstegn, der afslutter meddelelsen. Når mere end en variabel skal INDLÆSES, bør de være adskilt af kommaer, når de indtastes. Hvis de ikke er det, spørger computeren efter de resterende værdier ved at udskrive to spørgsmålstegn (??). Hvis du trykker RETURN tasten uden at indlæse en værdi, bibeholder INDLÆSNINGS variablen den værdi, der tidligere blev indlæst for denne variabel. Denne instruks kan kun anvendes i et program.

EKSEMPEL:

```
10 INPUT "VÆR SÅ VENLIG AT INDTASTE ET NUMMER";A

20 INPUT "OG DIT NAVN";A$

30 INPUT B$

40 PRINT "JEG TØR VÆDDE PÅ, DU IKKE VIDSTE HVAD JEG VILLE!"
```

INPUT

INPUT# fil nummer, liste af variabler

Fungerer på samme måde som INPUT, men tager data fra en tidligere åbnet fil eller enhed. Der tillades ingen meddelelsesstreng. Denne kommando kan kun bruges i programtilstand.

EKSEMPEL:

```
INPUT#2, A$, C, D$
```

LET

[LET] variabel = udtryk

Ordet LET bruges næppe nogen sinde i programmer, da det ikke er nødvendigt, men selve instruksen er kernen i alle BASIC programmer. Når

som helst en variabel defineres eller gives en værdi, er LET altid underforstået. Det variabelnavn som skal blive resultatet af en beregning er på venstre side af lighedstegnet og nummeret eller formlen er på højre side.

EKSEMPEL:

```
10 LET A = 5

20 B = 6

30 C = A*B+3

40 D = "HALLO"
```

LOCATE

LOCATE x-koordinat, y-koordinat

LOCATE kommandoen lader dig anbringe punktcursoren (PC) (punktmarkøren) hvor som helst på skærmen. PC er den øjeblikkelige position af startpunktet for næste tegning. Til forskel fra den almindelige markør kan man ikke se PC, men man kan flytte den ved hjælp af LOCATE kommandoen. For eksempel:

```
LOCATE 160, 100
```

flytter PC til centrum af højopløsningsskærmen. Du vil ikke se noget, før du faktisk tegner noget. Du kan til enhver tid finde ud af, hvor PC befinder sig ved brug af RDOT(0) FUNKTIONEN TIL AT FÅ X-koordinaten og RDOT(1) til at få Y-koordinaten. Farvekilden til prikken ved PC kan findes ved at udskrive RDOT(2). (I alle tegnekommandoer, hvori der er mulighed for farvevalg, kan du vælge en værdi fra 0 til 3 svarende til baggrunden, forgrunden, flerfarve 1 eller flerfarve 2 som farvekilde).

MONITOR

MONITOR

Denne kommando fører dig ud af BASIC og ind i den indbyggede maskinsprogs monitor. Monitoren bruges til at udvikle, fjerne fejl fra og udføre programmer i maskinsprog lettere end fra BASIC. Se afsnittet om monitor kommandoer for yderligere oplysninger. (Når du arbejder i monitoren, skal du taste X og trykke RETURN for at komme tilbage til BASIC).

NEXT

NEXT [variabel, ...,variabel]

NEXT instruksen bruges sammen med FOR instruksen. Når computeren møder en NEXT instruks, går den tilbage til den tilsvarende FOR instruks og kontrollerer løkkevariablen. (Se FOR instruksen for nærmere enkeltheder). Hvis løkken er afsluttet, fortsætter udførelsen med instruksen efter NEXT instruksen. Ordet NEXT kan efterfølges af et variabelnavn, en liste af variabelnavne adskilt af kommaer, eller af slet ingen variabelnavne. Hvis der ikke er listet nogen navne, er den sidst påbegyndte løkke den, der fuldendes. Hvis variablerne gives, færdiggøres de i rækkefølge fra venstre til højre.

EKSEMPEL:

```
10 FOR L = 1 TO 10:NEXT
20 FOR L = 1 TO 10:NEXT L
30 FOR L = 1 TO 10:FOR M = 1 TO 10:NEXT M, L
```

ON

ON udtryk <GOTO | GOSUB> line nr. 1, [, line nr. 2,...]

Denne kommando kan ændre GOTO og GOSUB instrukser til særlige udgaver af IF instruksen. Ordet ON følges af en formel, derefter enten GOTO eller GOSUB og en liste med linienumre adskilt af kommaer. Hvis resultatet af beregningen af formlen er 1, udføres første linie i listen. Hvis resultatet er 2, udføres det andet linienummer o.s.v. Hvis resultatet er 0 eller større end de i listen anførte linienumre, er den næste linie, der udføres, den instruks der følger efter ON instrukserne. Hvis nummeret er negativt, bliver resultatet en ILLEGAL QUANTITY ERROR.

EKSEMPEL:

```
10 INPUT X:IF X<0 THEN 10
20 ON X GOTO 50,30, 30, 70
```

Når X = 0 fortsætter programmet til linie 20. Når X = 2, fortsætter programmet til linie 30 o.s.v.

```
25 PRINT "FALDT IGENNEM":GOTO 10
30 PRINT "FOR HØJT":GOTO 10
50 PRINT "FOR LAVT":GOTO 10
70 END
```

OPEN

OPEN filnummer, enhed, [,sekundær adresse] [,"filnavn, type, tilstand"]

OPEN instruksen giver **C 16-plus/4** adgang til eksterne enheder som f.eks. kassettebåndoptager og diskette til data, en skriver eller endog **C 16-plus/4**'s skærm. Ordet OPEN efterfølges af et logisk filnummer, som er det nummer, hvortil alle andre BASIC instrukser refererer. Dette nummer ligger mellem 1 og 255. Der er altid et andet nummer efter det første, der kaldes enhedsnummer. Enhedsnummer 0 er **C 16-plus/4**'s tastatur, 3 er **C 16-plus/4**'s skærm, 1 er kassettebåndoptageren, 4 er skriveren, 8 er oftest en diskette. Et nul (0) kan indsættes foran enhedsnummerets ciffer (f.eks. 08 for 8). Det er tit en god ide at bruge samme nummer på filen og enheden, fordi det gør det let at huske, hvad der er hvad. Efter det andet nummer kan følge et tredje kaldet den sekundære adresse. Hvor der er tale om kassetten, kan det tredje nummer være 0 for læs, 1 for skriv og 2 for skriv med slut-på-bånd mærke tilsidst. Hvor der er tale om en diskette, refererer nummeret til kanalnummeret. I skriveren bruges de sekundære adresser til at sætte skriverens tilstand. Se håndbogen for hver enkelt ekstern enhed for nærmere oplysning om sekundære adresser. Der kan også være en streng efter det tredje nummer; det kunne være en kommando til diskettestationen eller navnet på filen på båndet eller disketten. Type og tilstand refererer kun til diskettefiler. (Filtyper er prg, seq, rel, og usr; tilstande er read og write).

EKSEMPLER:

10 OPEN 3,3	Åbner skærmen som enhed.
10 OPEN 1,0	Åbner tastaturet som enhed.
20 OPEN 1,1,0,"DOT"	Åbner kassetten for læsning, den fil der skal søges efter hedder DOT.
OPEN 4,4	Åbner en kanal for brug af skriver.
OPEN 15,8,15	Åbner kommandokanalen på disketten.
5 OPEN 8,8,12,"TESTFIL,SEQ,WRITE"	laver en sekventiel diskettefil til skrivning.

Se også: CLOSE, CMD, GET INPUT og PRINT instrukserne, og system variablerne ST, DS, og DS$.

PAINT

PAINT [farvekilde] [, [a,b] [,tilstand]]

Farvekilde.....	(0-3); (default er 1 forgrundsfarve)
a,b, ...	startkoordinat, skaleret (default er PC)
tilstand	0 = farvelæg et område, defineret af den valgte farvekilde.
	1 farvelæg et område defineret af en hvilken som helst anden kilde end baggrundskilden.

PAINT kommandoen gør det muligt for dig at udfylde et område med farve. Den udfylder området omkring det angivne punkt, indtil en grænse af samme farve (eller en hvilken som helst anden blot ikke baggrundsfarven, afhængigt af hvilken tilstand, du har valgt) dukker op. Slutpositionen for PC vil være ved startstedet (a,b).
BEMÆRK: Hvis startstedet allerede har samme farve som farvekilden, du nævner (eller en hvilken som helst, der ikke er baggrundsfarve, når tilstand 1 består), er der ingen ændring.

EKSEMPEL:

10 CIRCLE , 160,100,65,50	tegner omridset af en cirkel
20 PAINT , 160,100	udfylder cirklen med farve

POKE

POKE adresse, værdi

POKE kommandoen giver dig mulighed for at ændre enhver værdi i **C 16-plus/4**'s RAM hukommelse (Random Access Memory) og tillader dig at ædre mange af **C 16-plus/4**'s inddata/uddata registre. POKE efterfølges altid af to tal (eller ligninger). Det første tal er en plads i **C 16-plus/4**'s hukommelse. Det kunne have en hvilken som helst værdi fra 0 til 65535. Det andet tal er en værdi mellem 0 og 255, som er anbragt på pladsen og som erstatter enhver værdi, der tidligere befandt sig der.

EKSEMPEL:

10 POKE 28000,8	sætter stedet 28000 til 8
20 POKE 28*1000,27	sætter stedet 28000 til 27

BEMÆRK: PEEK, en kommando beslægtet med POKE, er listet under FUNKTIONER

PRINT

PRINT printliste

PRINT instruksen er den vigtigste udlæsningsinstruks i BASIC. Selvom PRINT instruksen er den første BASIC instruks, de fleste lærer at bruge, er der alligevel også her mange finesser, man kan komme til at mestre. Ordet PRINT kan efterfølges af enhver af de her nævnte:

Karakterer i anførselstegn	("tekstlinier")
Variabelnavne	(A, B, A$, X$)
Funktioner	(SIN(23), ABS(33))
Skilletegn	(; ,)

Karaktererne i anførselstegn benævnes ofte litteraler, fordi de udskrives nøjagtigt, som de fremkommer. Variabelnavne får den værdi, de indeholder (enten et tal eller en streng), udskrevet. Funtioner får også deres talværdier udskrevet. Skilletegn bruges som en hjælp til at formattere data ordentligt på skærmen. Kommaet deler skærmen i 4 kolonner til data, medens semikolon ikke tilføjer nogen mellemrum. Begge tegn kan bruges som sidste symbol i instruksen. Dette har til resultat, at den næste PRINT instruks "opfører sig", som om den fortsætter den foregående PRINT instruks.

EKSEMPEL:

	RESULTAT
10 PRINT "HALLO"	HALLO
20 A$ = "DER":PRINT "HALLO",A$	HALLO, DER
30 A = 4:B = 2:PRINT A+B	6
50 J = 41:PRINT J;:PRINT J−1	41 40
60 C = A+B:D = A−B:PRINT A;B;C,D;	4 2 6 2

Se også: POS(), SPC(), and TAB() FUNKTIONERNE.

PRINT#

PRINT# filnummer, print liste

Der er et par forskelle på denne instruks og PRINT instruksen. For det første efterfølges ordet PRINT# af et tal, der refererer til den eksterne enhed eller en tidligere åbnet fil. Tallet følges af et komma og en liste, der skal udskrives. Komma og semikolon fungerer på samme måde for så vidt angår mellemrum, som de gør i PRINT instruksen. Visse eksterne enheder kan ikke fungere med instrukserne TAB og SPC.

EKSEMPEL:

```
100 PRINT# 1,"HALLO DER!",A$,B$,
```

PRINT USING

PRINT [#filnummer,] USING format; print liste;

Ved hjælp af disse instrukser kan du meddele skærmen, skriveren eller en anden ekstern enhed formatet på streng- og numeriske elementer, du vil udskrive. Sæt det format du ønsker i anførselstegn. Det er formatlisten. Tilføj derefter et semikolon og en liste over, hvad du vil have udskrevet i formatet for udskrivningslisten. Listen kan være variabler eller de faktiske værdier, du vil have udskrevet.
For eksempel:

```
5 X = 32: Y = 100.23: A$ = "KAT"
10 PRINT USING "$##.##";13.25,X,Y
20 PRINT USING "####>#";"CBM",A$
```

Når du kører dette, udskriver linie 10:

$13,25 $32.00 $*****	udskriver ***** i stedet for Y værdien, fordi Y har 5 cifre, hvilket ikke passer med formatlisten (som forklaret nedenfor)

Linie 20 udskriver dette:

CBM CAT	lader tre mellemrum stå for "CBM" udskrives, som defineret i formatlisten

KARAKTER	NUMERISK	STRENG
Nummertegn (#)	X	X
Plus (+)	X	
Minus (−)	X	
Decimalkomma (.)	X	
Komma (,)	X	
Dollartegn ($)	X	
Fire indskudstegn (↑ ↑ ↑ ↑)	X	
Lighedstegn (=)		X
Større-end tegn (>)		X

Nummertegn (#) reserverer plads til en enkelt karakter i udlæsefeltet. Hvis dataelementet indeholder flere karakterer, end du har # i dit formatfelt, sker der følgende:

For et numerisk element fyldes hele feltet med asterisker (*). Der udskrives ingen tal.

For eksempel:

```
10 PRINT USING "####",X
```

For disse X-værdier viser dette format følgende på skærmen:

A = 12.34	12
A = 567.89	568
A = 123456	****

For et STRENG element bliver strengens data afskåret ved feltets grænser. Der udskrives kun så mange karakterer, som der er nummertegn (#) i formatelementet. Afskæring forekommer på højre side.

Plus (+) og minus (−) tegnene kan bruges enten i første eller sidste position af et formatfelt, men ikke i begge. Plustegnet udskrives, hvis tallet er positivt. Minustegnet udskrives, hvis tallet er negativt.

Hvis du bruger et minustegn i forbindelse med et positivt tal, udskrives et blanktegn i den karakterposition, der angives ved minustegnet.

Hvis du hverken bruger et plus- eller minustegn i dit formatfelt for et numerisk dataelement, udskrives et minustegn før det første ciffer eller

dollarsymbol, hvis tallet er negativt, medens der ikke udskrives noget tegn, hvis tallet er positivt. Dette betyder, at du kan udskrive en karakter mere, hvis tallet er positivt. Hvis der er for mange cifre til, at det kan passe med det felt, der er angivet ved # og +/− tegnene, opstår der et overløb og feltet fyldes med asterisker (*).

Et decimalkomma (.) symbol betegner kommaets position i tallet. Man kan kun have et decimalkomma i et formatfelt. Hvis man ikke specificerer et decimalkomma i sit formatfelt, afrundes værdien til nærmeste hele tal og udskrives uden decimaler.

Når man specificerer et decimalkomma, må det antal cifre, der står foran kommaet (inkl. minustegnet, hvis værdien er negativ), ikke overstige antallet af # før decimalkommaet. Hvis der er for mange cifre opstår der et overløb og feltet fyldes med asterikser (*).

Et komma (,) tillader dig at placere kommaer i numeriske felter. Kommaets positition i formatlisten angiver, hvor kommaet dukker op i et udskrevet tal. Kun kommaer inden i et tal udskrives. Ubrugte kommaer til venstre for første ciffer vises som fyldkarakteren. Mindst et # skal stå før det første komma i et felt.

Hvis man angiver kommaer i et felt og tallet er negativt, udskrives et minustegn som første karakter, selvom karakterpositionen er angivet som et komma.

EKSEMPLER:

FELT	UDTRYK	RESULTAT	KOMMENTAR
##.#+	−.01	0.01−	Nul tilføjet foran komma.
##.#−	1	1.0	Nul tilføjet efter komma.
####	−100.5	−101	Afrundet til tal uden decimaler.
####	−1000	****	Overløb fordi fire cifre og minustegn ikke passer i feltet.
###.	10	10.	Decimalkomma tilføjet.
#$##	1	$1	Dollartegn foran.

Et dollartegn ($) symbol viser, at et dollartegn vil blive udskrevet i tallet. Hvis du ønsker at dollartegnet skal flyde (altid blive sat foran tallet), skal du angive mindst et # foran dollartegnet. Hvis du angiver et dollartegn uden først at sætte et #, udskrives dollartegnet i den position, der er vist i formatfeltet.

Hvis du angiver kommaer og/eller et plus- eller minustegn i et formatfelt med et dollartegn, udskriver dit program et komma eller tegn før dollartegnet.
Symbolet med de fire pile der peger opad (↑ ↑ ↑ ↑), også kaldet indskudstegn, bruges til at angive, at tallet skal udskrives i E+format. Du skal foruden ↑ ↑ ↑ ↑ bruge # for at angive feltets bredde. ↑ ↑ ↑ ↑ kan forekomme enten før eller efter # i formatfeltet.
Du skal angive fire indskudstegn (↑ ↑ ↑ ↑), når du vil udskrive et tal i E−format (eksponentiel notation). Hvis du angiver mere end fire indskudstegn, bruges kun de første fire. Det femte indskudstegn tolkes som et ingen-tekst-symbol.
Et lighedstegn (=) bruges til at centrere en streng i feltet. Du angiver feltbredden ved antallet af karakterer (# og =) i formatfeltet. Hvis strengen indeholder færre karakterer og således fylder mindre end feltbredden, bliver strengen centreret i feltet. Hvis strengen indeholder flere karakterer, end der kan rummes i feltet, bliver karaktererne yderst til højre afskåret og strengen fylder hele feltet.
Et større end tegn (>) bruges til at højrestille en streng i et felt. Du angiver feltbredden ved antallet af karakterer (# og =) i formatfeltet. Hvis strengen indeholder færre karakterer, end der skal til for at fylde feltet, bliver strengen højrestillet i feltet. Hvis strengen indeholder flere karakterer, end feltet kan rumme, bliver karaktererne yderst til højre afskåret og strengen fylder hele feltet.

PUDEF

PUDEF "1 til 4 karakterer"

PUDEF gør det muligt for dig at redefinere indtil 4 symboler i PRINT USING instruksen. Du kan ændre blanktegn, kommaer, decimalkommaer og dollartegn til andre karakterer ved at anbringe den nye karakter på den korrekte position i PUDEF styrestrengen.

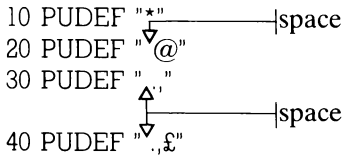
Position 1 er fyldkarakteren. Default er et blanktegn. Sæt en ny karakter her, når du vil have en anden karakter frem på blanktegnenes plads.

Position 2 er tusindskilletegn. Default er et komma.

Position er decimalkommaet.

Position 4 er dollartegnet.

EKSEMPLER:



Udskriver * på blanktegnenes plads.
Udskriver @ i stedet for kommaer.
Udskriver decimalkommaer i stedet for kommaer og kommaer i stedet for decimalpunktum.
Udskriver tegnet for engelske pund i stedet for $, decimalkommaer i stedet for kommaer og kommaer i stedet for decimalpunktum. Andre tegn er default værdierne.

READ

READ liste af variabler

Denne instruks bruges til at få oplysninger fra DATA instrukser ind i variabler, hvor disse data kan bruges. READ variable list instruksen kan indeholde både strenge og numre. Man må passe på at undgå at læse strenge, hvor READ instruksen forventer et tal, hvilket forårsager en ERROR melding.

EKSEMPEL:

READ A$, G$, 45

REM

REM bemærkning

REM er blot en kommentar til den person, der læser programlisten. Den forklarer måske et afsnit af programmet, eller den giver oplysning om forfatteren o.s.v. REM instrukser påvirker på ingen måde afviklingen af programmet bortset fra at forlænge dette (og derved gøre det langsommere). Ordet REM kan efterfølges af en hvilken som helst tekst, skønt brug af grafikkarakterer giver mærkelige resultater.

EKSEMPEL:

10 NEXT X: REM DENNE LINIE ER UNØDVENDIG

RESTORE

RESTORE [line]

Når den udføres i et program, omstilles pegepinden, der viser det element i en DATA instruks, der skal læses næste gang, til det første element i listen. Dette giver dig mulighed for at omlæse oplysningerne. Hvis en line efterfølger RESTORE instruksen, indstilles pegepinden på den linie. Ellers omstilles pegepinden til den første DATA instruks i programmet.

EKSEMPEL:

RESTORE 200

RESUME

RESUME [line | NEXT]

Bruges til at gå tilbage til udførelse efter at have fundet en fejl med TRAP instruksen. Hvis der ikke er argumenter, forsøger RESUME igen at udføre den linie i hvilken fejlen var. RESUME NEXT genoptager udførelsen ved næste instruks efter den instruks, der indeholder fejlen; RESUME line vil GOTO den angivne linie og begynde udførelsen der.

RETURN

RETURN

Denne instruks bruges altid sammen med GOSUB instruksen. Når programmet møder en RETURN instruks, går den til den instruks, der følger umiddelbart efter den sidst udførte GOSUB kommando. Hvis der ikke tidligere var udstedt nogen GOSUB instruks, afgives der en RETURN WITHOUT GOSUB ERROR meddelelse og programudførelsen stopper.

SCALE

SCALE <1 | 0>

Skaleringen af bit maps i flerfarve- og højopløsningstilstand kan ændres ved hjælp af SCALE kommandoen. Indtastning:

SCALE 1

åbner for skalering. Koordinaterne kan så skaleres fra 0 til 1023 både i X og Y i stedet for de normale skaleringsværdier som er:

flerfarvetilstand.....	X = 0 til 159 Y = 0 til 199
højopløsningstilstand	X = 0 til 319 Y = 0 til 199

Skaleringen kan slås fra ved indtastning af 'SCALE 0'.

SCNCLR

SCNCLR

Sletter skærmbilledet hvad enten det er grafik, tekst eller begge dele (delt skærm).

SOUND

SOUND stemme nummer, frekvens, varighed

Denne instruks frembringer en lyd ved brug af en af tre stemmer med en frekvensstyring i området 1-1023 med en varighed fra 0-65535 tresindstyvendedele af et sekund.

V	STEMME
1	stemme 1 (tone)
2	stemme 2 (tone)
3	stemme 2 (hvid støj)

Hvis man forlanger en SOUND til stemme N og den foregående SOUND til samme N stadig spiller, venter BASIC på, at den foregående SOUND skal spille færdigt. SOUND med en varighed på 0 er et særtilfælde. Det får BASIC til omgående at afbryde den nuværende SOUND til den stemme, uden hensyn til det tidsrum, der resterer for den foregående SOUND. Se MUSIK NODE TABELLEN for de frekvensstyringsværdier, der svarer til virkelige noder.

EKSEMPEL:

SOUND 2, 800, 360	Spiller en node ved brug af stemme 2 med frekvensen indstillet på 800 i et minut.

SSHAPE/GSHAPE

SSHAPE og GSHAPE bruges til at gemme og placere rektangulære områder af flerfarve- eller højopløsningsskærmbilleder ved brug af BASIC strengvariabler. Kommandoen for lagring af et område er:

SSHAPE strengvariabel, a1, b1 [,a2,b2]

strengvariabel.....	Strengnavn til at lagre data i
a1,b1	Hjørnekoordinat (skaleret)
a2,b2	Hjørnekoordinat modsat (a1,b1) (default er PC)

Da BASIC begrænser længden af strenge til 255 karakterer, er størrelsen af det område, du kan lagre, også begrænset. Den strengstørrelse der kræves kan beregnes ved brug af en af de følgende (uskalerede) formler:

L(flerfarve) = INT ((ABS(a1−a2)+1) / 4+.99)*(ABS(b1−b2)+1)+4
L(højopl.) = INT ((ABS(a1−a2)+1) / 8+.99)*(aBS(b1−b2)+1)+4

Figuren lagres række for række. De sidste fire bytes i strengen indeholder kolonne- og rækkelængder minus én (d.v.s.: ABS(A1−a2)) i lavt/højt byte format (hvis skaleret divideres længderne med 3.2 (X) og 5.12 (Y).

Kommandoen til visning af en lagret figur på et hvilket som helst areal af skærmen:

GSHAPE strengvariabel [, [a,b] [,måde]]

streng	Indeholder den figur, der skal tegnes
a,b, ...	Øverste venstre koordinat viser, hvor figuren skal tegnes (skaleret – default er PC)
måde.....	Placeringsmåde: 0: placer tegning som den er (default) 1: placer negativ tegning 2: OR tegning med arealet 3: AND tegning med arealet 4: XOR tegning med arealet

STOP

STOP

Denne instruks standser programmet. En meddelelse, BREAK IN LINE, hvor er det linienummer der indeholder STOP. Programmet kan startes

igen ved den instruks, der følger efter STOP, hvis du bruger CONT kommandoen. STOP instruksen anvendes almindeligvis medens der rettes fejl i et program.

SYS

SYS adresse

Ordet SYS efterfølges af et decimalt talord eller en numerisk variabel i området 0 til 65535. Programmet begynder at udføre maskinsprogsprogrammet fra den hukommelsesadresse, der svarer til det pågældende tal. Funktionen minder om USR funktionen, men overfører ikke parametre.

TRAP

TRAP [line]

Når TRON sættes, opfanger TRAP instruksen alle fejl (inklusive STOP KEY) undtagen "UNDEF'D STATEMENT ERROR". I tilfælde af eventuel udførelsesfejl sættes fejlmærket og udførelsen overføres til det linienummer, der er nævnt i TRAP instruksen. Det linienummer hvori fejlen opstod kan findes ved hjælp af systemvariablen EL. Den specifikke fejl indeholdes i systemvariablen ER. Strengfunktionen ERR$ (ER) giver den fejlmeddelelse, der svarer til en eventuel fejl ER.
BEMÆRK: En fejl i en TRAP instruks kan ikke fanges. RESUME instruksen kan bruges til genoptagelse af udførelsen. TRAP instruksen uden angivelse af linie argument afbryder fejlopfangning.

TRON

TRON

TRON bruges ved rettelse i fejl i programmer. Denne instruks indleder søgetilstand. Når du arbejder i søgetilstand, udskrives linienumrene på instrukserne efterhånden som disse udføres.

TROFF

TROFF

Denne instruks afbryder søgetilstanden.

VOL

VOL lydstyrke

Sætter den øjeblikkelige lydstyrke for SOUND kommandoen. Lydstyrken kan sættes fra 0 til 8, hvor 8 er største lydstyrke og 0 betegner, at der er "skruet helt ned". VOL berører begge stemmer.

WAIT

WAIT adresse, værdi 1 [,værdi 2]

WAIT instruksen bruges til at standse programmet, indtil indholdet af et område i hukommelsen ændres på en bestemt måde. Adressen skal være i området fra 0 til 65535. Værdi 1 og værdi 2 skal være i området fra 0 til 255. Indholdet af området i hukommelsen bliver først exclusive-or'ed med værdi 2 (hvis der er en sådan) og derefter logisk and'ed med værdi 1. Hvis resultatet er nul, undersøger programmet området i hukommelsen igen. Hvis resultatet ikke er nul, fortsætter programmet med den næste instruks.

Yderligere oplysninger om grafik instruksen

Der er et par begreber, der vedrører alle bit map grafikinstrukser. Først er der begrebet Punkt Cursoren (PC). PC ligner cursoren (markøren) i tekst-tilstand; det er den position, hvori næste prik skal tegnes. Til forskel fra tekstmarkøren er PC usynlig. Alle tegnekommandoer bruger PC. Desuden tillader locate kommandoen, at man flytter PC uden at tegne noget. Uanset hvor du måtte ønske at bruge X,Y koordinaterne i en tegnekommando, kan du i stedet bruge RELATIVE koordinater. RELATIVE koordinater er baseret på PC's øjeblikkelige værdi. Hvis du vil bruge relative koordinater, sætter du bare et + eller − foran dine koordinater. Et plustegn foran X værdien bevæger PC til højre. Et minustegn foran X værdien bevæger PC til venstre. På lignende måde bevæger et minustegn foran Y koordinaten PC opad, medens et plustegn bevæger PC nedad. For eksempel:

LOCATE+100,−25	bevæger PC 100 punkter til højre og 25 punkter opad
DRAW,+10,+10to100,100	tegner en linie 10 punkter til højre for og 10 punkter under den øjeblikkelige værdi af PC til det absolutte punkt 100,100.

Man kan også angive en afstand og en vinkel i forhold til den øjeblikkelige PC ved at adskille de to parametre med et semikolon.

For eksempel:

LOCATE 50;45	bevæger PC fra dens øjeblikkelige position i en afstand af 50 prikker og i en vinkel på 45 grader.

Funktioner
Numeriske funktioner

Numeriske funktioner klassificeres som sådanne, fordi de returnerer tal. De funktioner, de udfører, spænder fra at beregne matematiske funktioner til at angive en plads på skærmen. Numeriske funktioner følger formen:

funktion (argument)

hvor argumentet kan være en numerisk værdi, variabel eller streng.

ABS(X) (absolut værdi)

Den absolutte værdifunktion giver som resultat den positive værdi af argumentet X.

ASC(X$)

Denne funktion giver som resultat ASCII kode (nummeret) for den første karakter af X$.

ATN(X) (bue tangens)

Giver som resultat den vinkel målt i radianer, hvis tangent er X.

COS(X) (kosinus)

Giver som resultat kosinus af X, hvor X er en vinkel målt i radianer.

DEC (hexadecimalstreng)
Giver som resultat decimalværdien af hexadecimalstrengen (0<hexadecimalstrengen<FFFF)

EKSEMPEL:

N = DEC("F4")

EXP(X) (eksponentialfunktionen)

Giver som resultat den matematiske konstant e (2.71828183) opløftet til X'te potens.

FNxx(x)

Giver som resultat værdien af den brugerdefinerede funktion xx defineret ved hjælp af DEF FNxx instruks.

INSTR (streng 1, streng 2 [,startposition])

Giver som resultat streng 2's position i streng 1 i eller efter startpositionen. Startpositionens default er begyndelsen af streng 2. Hvis der ikke findes nogen lighed, gives værdien 0 som resultat.

EKSEMPEL:

PRINT INSTR("KATTEN I HATTEN", "HAT")

resultatet er 10, fordi HAT begynder ved den tiende karakter i streng 1.

INT(X) (integer = heltal)

Giver som resultat heltalsdelen af X med alle decimaler på højre side af kommaet fjernet. Resultatet er altid mindre end eller lig med X. Eventuelle negative tal med decimaler bliver således heltallet mindre end deres øjeblikkelige værdi (f.eks. INT(−4,5) = −5).

Hvis INT funktionen skal bruges til at runde op eller ned er formlen INT(X+/−.5).

EKSEMPEL:

X = INT(X*100+.5)/100) Runder af til næsthøjeste øre.

JOY (n)

Når n = 1 ..... joystick nr. 1's position
n = 2 ..... joystick nr. 2's position

Enhver værdi på 128 eller mere betyder at fireknappen også nedtrykkes. Retningen angives således:

		OP		
fire = 128+		1		
	8		2	
VENSTRE 7		0		3 HØJRE
	6		4	
		5		
		NED		

EKSEMPEL:

JOY(2) = 135 joystick 2 fyrer til venstre

LOG(X) (logaritmefunktionen)

Giver som resultat den naturlige logaritme til X. Den naturlige logaritme er logaritmen til e-logaritme (se EXP(X)). For konvertering til 10-talslogaritme skal resultatet divideres med LOG(10).

PEEK(X)

Denne funktion bruges til at finde indholdet i adressen X, hvor X ligger i området mellem 0 og 65535 og resultatet mellem 0 og 255. Denne funktion bruges også i forbindelse med POKE instruksen.

RCLR(N)

Giver som resultat den farve, der i øjeblikket er tildelt kilde N (0<N<4) (0 = baggrund, 1 = forgrund, 2 = flerfarve 1,3 = flerfarve 2,4 = skærmranden)

RDOT(N)

Giver som resultat oplysning om den øjeblikkelige position af punkt cursoren (PC) ved XPOS/YPOS.

N – 0 for XPOS (x-koordinat)
1 for YPOS (y-koordinat)
2 farvekilde

RGR(X)

Giver som resultat den øjeblikkelige grafiktilstand (X er et pseudoargument).

RLUM(N)

Giver som resultat det luminansniveau, der er tildelt kilde N.

RND(X) (tilfældigt tal)

Denne funktion giver som resultat et tilfældigt tal mellem 0 og 1. Dette er nyttigt i spil, til simulering af terningkast og andre chancebetonede elementer og finder også anvendelse i statistik. Det første tilfældige tal bør frembringes ved hjælp af formlen RND(−TI) for at begynde forskelligt hver gang. Derefter bør tallet i X være et 1 eller et hvilket som helst positivt tal. (X repræsenterer det tal, hvorpå det tilfældige tal er baseret). Hvis X er nul, baseres det tilfældige tal på hardware clock, hver gang RND bruges. En negativ værdi for X vil "blande kortene" ved brug af X og giver en rækkefølge for de tilfældige tal. Brug af samme negative tal for X som basis, vil resultere i samme rækkefølge af tilfældige tal. En positiv værdi giver tilfældige tal baseret på den foregående basis. Til simulering af et terningkast bruges formlen (INT(RND(1)*6+1). Først multipliceres det tilfældige tal fra 0-1 med 6, hvilket udvider området til 0-6 (faktisk større end nul og mindre end seks). Så tillægges 1, hvilket giver et område fra 1 til 7. INT funktionen afskærer alle decimaler og giver et resultat i form af et ciffer mellem 1 og 6.
For at simulere 2 terninger adderes to af de tal, som blev resultatet ved brug af ovennævnte formel.

EKSEMPEL:

```
100 X = INT(RND(1)*6)+INT(RND(1)*6)+2     Simulerer 2 terninger.
100 X = INT(RND(1)*1000)+1                Tal mellem 1 og 1000.
100 X = INT(RND(1)*150)+100               Tal mellem 100 og 249.
```

SGN(X)(fortegn)

Denne funktion viser om X er positiv, negativ eller nul. Resultatet bliver +1, hvis X er positiv, 0 hvis det er nul og −1, hvis X er negativ.

SIN(X) (sinusfunktion)

Dette er den trigonometriske sinusfunktion. Resultatet er sinus af X, hvor X er en vinkel målt i radianer.

SQR(X) (kvadratrod)

Denne funktion giver som resultat kvadratroden af X, hvor X er et positivt tal eller 0. Hvis X er negativt, bliver resultatet en ILLEGAL QUANTITY ERROR.

TAN(X) (tangens)

Giver som resultat tangens til X, hvor X er en vinkel målt i radianer.

USR(X)

Når denne funktion bruges, springer programmet til et maskinsprogsprogram, hvis startadresse ligger i hukommelsesadresserne 1281 og 1282. Parameteren X overgives til maskinsprogsprogrammet i akkumulatoren med flydende komma. Et andet tal gives tilbage til BASIC programmet via kaldevariablen. Med andre ord, dette giver dig mulighed for at udveksle en varibel mellem maskinkode og BASIC.

VAL(X$)

Denne funktion konverterer strengen X$ til et tal og er i alt væsentligt det omvendte af STR$ funktionen. Strengen undersøges fra venstre mod højre, så længe der er karakterer i genkendeligt, numerisk format. Hvis **C 16-plus/4** finder ulovlige karakterer, bliver kun den del af strengen, der ligger foran en sådan, konverteret.

EKSEMPEL:

```
10 X = VAL("123.456")        X = 123.456

10 X = VAL("3E03")           X = 3000

10 X = VAL("12A13B")         X = 12

10 X = VAL("RIU017*")        X = 0

10 X = VAL("-1.23.23.23")    X = 1.23
```

Strengfunktioner

Strengfunktioner adskiller sig fra numeriske funktioner ved, at de som resultat giver karakterer, grafik eller tal fra en streng (defineret ved hjælp af anførselstegn) i stedet for et tal.

CHR$(X)

Denne funktion giver en strengkarakter, hvis ASCII kode er X.

ERR$(N)

Giver en streng, der beskriver fejltilstanden N (se TRAP).

HEX$(N)

Giver en streng på 4 karakterer indeholdende den hexadecimale repræsentation af værdien N (0<N<65535).

LEFT$(X$,Z)

Giver en streng, der indeholder de yderste venstre X karakterer i X$.

LEN(X$)

Giver antallet af karakterer (inkl. mellemrum og andre symboler) i strengen X$.

MID$(X$,S,X)

Giver en streng med X karakterer, begyndende med karakter nr. S i X$. MID$ kan også bruges på venstre side af en tildelingsinstruks som en pseudovariabel såvel som en funktion. MID$ (strengvariabel, startposition, længde) = kildestreng.

Denne funktion tildeler påny positionsværdierne (startposition til og med startposition+længde af kildestrengen) til karaktererne i strengvariabel på tilsvarende adresser. Længdens default er længden af strengvariablerne og en fejl bliver resultatet, hvis (startposition+længde) er større end længden af kildestrengen.

EKSEMPEL:

```
10 A$ = "HUNDEN I HATTEN":
20 PRINT A$
30 MID$(A$,10,3) = "KAT"
40 PRINT A$
```

RIGHT$(X$,X)

Giver de yderste højre X karakter i X$.

STR$(X)

Giver en streng, som er identisk med den udskrevne version X.
EKSEMPEL:

```
A$ = STR$(X)
```

Andre funktioner

FRE(X)

Denne funktion giver antallet af ubrugte bytes, der er tilbage i hukommelsen. X er et pseudoargument.

POS(X)

Giver til resultat nummeret på den kolonne (0-39), hvor næste PRINT instruks begynder på skærmen. X er et pseudoargument.

SPC(X)

Denne bruges i PRINT instruksen til at springe over X mellemrum. X kan have en værdi fra 0-255.

TAB(X)

Denne bruges i PRINT instruksen. Det næste element, der skal udskrives, er i kolonne X. X kan have en værdi fra 0-255.

π (PI)

π symbolet har, når det bruges i en ligning, værdien 3,14159265.

Variabler

C 16-plus/4 bruger tre typer variabler i BASIC. Disse er: normale numeriske, integer numeriske og streng (alfanumeriske) variabler.

Normale NUMERISKE VARIABLER, også kaldet variabler med flydende komma, kan have en hvilken som helst værdi fra 10^{-38} til 10^{+38} med op til ni cifres nøjagtighed. Hvis et tal bliver større, end det kan skrives med ni cifre som i 10^{10} eller 10^{-10}, viser computeren det i eksponentiel notation med tallet normalisert til et ciffer og otte decimaler fulgt af bogstavet E og tiende potens, hvormed tallet ganges. Tallet 12345678901 vises for eksempel således: 1.23456789E+10.

INTEGER VARIABLER kan bruges, når tallet er mellem +32767 og −32768 og uden decimaler. En integer variabel er et tal som f.eks. 5, 10 eller −100. Integer variabler fylder mindre end variabler med flydende komma, især når de bruges i en total.

STRENGVARIABLER er sådanne som bruges til karakterdata, der kan indeholde tal, bogstaver og en hvilken som helst anden karakter, som **C 16-plus/4** kan lave. Et eksempel på en strengvariabel er "plus/4".

Variabelnavne kan bestå af et enkelt bogstav, et bogstav efterfulgt af et tal eller to bogstaver. Variabelnavne kan være længere end 2 karakterer, men kun de to første er vigtige.

En integer variabel angives ved at bruge procenttegnet (%) efter variabelnavnet. Navnet på strengvariabler efterfølges af dollartegnet ($).

EKSEMPLER:

Numeriske Variabelnavne:	A, A5, BZ
Integer Variabelnavne:	A%, A5%, BZ%
Strengvariabelnavne:	A$, A5$, BZ$

TABELLER er lister med variabler med samme navn, der bringer et ekstra tal (eller flere) til at angive et element i sættet. Tabellerne defineres ved brug af DIM instruksen og de kan være flydende komma-, integer- eller strengvariabeltabeller. Tabelvariabelnavnet efterfølges af et sæt parenteser (), der indeholder nummeret på variablen i listen.

EKSEMPLER: A(7),BZ%(11),A$(87)

Tabeller kan have mere end en dimension. En todimensional tabel kan betragtes som indeholdende rækker og kolonner med det første tal til identifikation af rækken og det andet tal i parenteserne til identifikation af kolonnen (som om det specificerede et bestemt net på et kort).

EKSEMPLER: A(7,2),BZ%(2,3,4),Z$(3,2)

Reserverede variabelnavne

Der er syv variabelnavne, som er reserveret til **C 16-plus/4**'s brug og som ikke må bruges til andre formål. Det drejer sig om variablerne DS, DS$, ER, EL, ST, TI, og TI$. Man kan heller ikke bruge nøgleord som TO og IF eller navne som indeholder nøgleord som SRUN, RNEW eller XLOAD som variabelnavne.

ST er en statusvariabel for indlæsning og udlæsning (undtagen normale skærm/tastatur operationer). Værdien af ST afhænger af resultaterne af den seneste indlæsnings/udlæsningsoperation, men i almindelighed var operationen vellykket, hvis værdien af ST er 0.

TI og TI$ er variabler, der har forbindelse med det realtidsur, der er indbygget i **C 16-plus/4**. Systemuret opdateres for hver tresindstyvendedel sekund. Det starter ved 0, når der tændes for **C 16-plus/4** og det slettes kun ved at ændre værdien af TI$. Variablen TI giver urets øjeblikkelige stilling i tresindstyvendedele af et sekund.
TI$ er en streng, der læser værdien af realtidsuret som et 24 timers ur. De to første karakterer af TI$ viser timen. tredje og fjerde karakter angiver minutterne og femte og sjette karakter er sekunderne. Denne variabel kan sættes til en hvilken som helst værdi (så længe alle karakterer er tal), og bliver automatisk opdateret som et 24 timers ur.

EKSEMPEL: TI$ = "101530" sætter uret til 10:15 og 30 sekunder

Urets værdi mistes, når der slukkes for **C 16-plus/4**. Det starter ved nul, når der tændes for **C 16-plus/4** og sluttes til nul, når værdien overstiger 235959 (23 timer, 59 minutter og 59 sekunder).

Variablen DS læser diskettedrevets kommandokanal og tilbagesender drevets øjeblikkelige status. For at få disse oplysninger i ord PRINT DS$. Disse statusvariabler bruges efter en disketteoperation som DLOAD eller

DSAVE for at finde ud af, hvorfor det røde fejlvisningslys på diskettedrevet blinker.

ER, EL og ERR$ er variabler, der bruges i fejlfindingsrutiner. De er normalt kun nyttige i et program. ER returnerer den sidste fejl, den mødte fra programmet blev KØRT. EL er den linie, i hvilken fejlen opstod. ERR$ er en funktion, der gør det muligt for programmet at udskrive en af BASIC's fejlmeddelelserne. PRINT ERR$(ER) udskriver den egentlige fejlmeddelelse.

Basic operatorer

De ARITMETISKE operatorer omfatter følgende tegn:

+	addition
−	subtraktion
*	multiplikation
/	division
↑	opløftning til potens (eksponentiering)

På en linie, der indeholder mere end en operator er der en fast orden, hvori operationerne altid forekommer, Hvis flere operatorer bruges sammen, prioriterer computeren dem som følger: først eksponentiering, så multiplikation og division og til sidst addition og subtraktion. Hvis to operationer har samme prioritet, udføres beregningerne i rækkefølge fra venstre mod højre. Hvis man ønsker, at disse operationer skal komme i en anden rækkefølge, tillader **C 16-plus/4** BASIC, at man giver en beregning en højere prioritet ved at sætte den i parenteser. Operationen der står i parentes bliver udført før nogen anden operation. Man må sikre sig, at ligningerne har samme antal venstre og højre parenteser, ellers får man en SYNTAX ERROR meddelelse, når programmet kører.

Der er også operatorer for ligheder og uligheder der benævnes RELATIONELLE operatorer. Aritmetiske operatorer vil altid have forrang for relationelle operatorer.

=	er lig med
<	er mindre end
>	er større end
<= eller =<	er mindre end eller lig med
>= eller =>	er større end eller lig med
<> eller ><	er ikke lig med

Endelig er der tre LOGISKE operatorer med lavere prioritet end både aritmetiske og relationelle operatorer:

AND
OR
NOT

De anvendes hyppigst til at forbinde flere formler i IF ... THEN instrukser. Når de anvendes med aritmetiske operatorer evalueres de til sidst (d.v.s. efter + og −).

EKSEMPLER:

IF A = B AND C = D THEN 100	kræver at både A = B og C = D er opfyldt.
IF A = B OR C = D THEN 100	tillader at enten A = B eller C = D er opfyldt
A = 5:B = 4:PRINT A = B	viser værdien 0
A = 5:B = 4:PRINT A>B	viser en værdi af −1
PRINT 123 AND 15:PRINT 5 OR 7	viser 11 og 7

Konverteringsprogrammer

KONVERTERING AF STANDARD BASIC PROGRAMMER TIL COMMODORE BASIC 3.5

Hvis du har programmer skrevet i andet end Commodore BASIC, kan det være nødvendigt med nogle mindre justeringer, før du kører dem på **C 16-plus/4**. Her er nogle tips der vil gøre konvertering lettere.

STRENG DIMENSIONER

Streg alle de instrukser der bruges til at angive strengenes længde. En instruks som f.eks. DIM A$(I,J), der dimensionerer et strengsæt for J elementer på længden I, bør konverteres til Commodore BASIC instruks DIM A$(J).

Nogle BASIC versioner bruger et komma eller et & til sammenkædning af

strenge (sammenknytning). Hvert af disse skal ændres til et + tegn, som er Commodore BASIC 3.5's operator for sammenkædning af strenge.

I Commodore BASIC bruges MID$, RIGHT$ og LEFT$ funktionerne til at flytte understrenge af strenge. Former som A$(I) for adgang til I'ste karakter i A$, eller A$(I,J) til at flytte en understreng af A$ fra position I til J skal ændres som følger:

Anden BASIC	**Commodore ASIC 3.5**
A$(I) = X$	MID$(A$,I,J) = X$
A$(I,J) = X$	MID$(A$,I,J) = X$

TILDELING AF SAMME VÆRDI TIL FLERE VARIABLER

For at sætte B og C lig med nul tillader nogle BASIC versioner instrukser i følgende form:

```
10 LET B = C = 0
```

Commodore BASIC ville opfatte det andet = tegn som en logisk operator og sætte B = −1, hvis C = 0. Lav i stedet denne instruks om til:

```
10 C = 0:B = 0
```

FLERE INSTRUKSER I SAMME LINIE

Nogle BASIC versioner bruger en skråstreg (/) til adskillelse af flere instrukser i samme linie. I BASIC 3.5. skal man bruge et kolon (:) til adskillelse af alle instrukser.

MATRICE FUNKTIONER

Programmer, der bruger de matrice funktioner, der kan fås i nogle af BASIC versionerne, skal omskrives ved brug af FOR ... NEXT løkker for korrekt udførelse.

OMPROGRAMMERING AF FUNKTIONSTASTER

Man kan omprogrammere funktionstasterne, så de passer til funktionstasterne på Commodore 64 og VIC 20. (Det gør det også lettere at konvertere programmer fra disse maskiner).

Til omprogrammering af tasterne indsætter du følgende linie i dit program:

```
10 FOR I = 1 TO 8:KEY I, CHR$(I+132): NEXT
```

Når du herefter bruger en funktionstast, vil den skrive en usynlig og virkningsløs karakter fra 133-140, ligesom Commodore 64 gør. For at kontrollere dette i et program, kan du bruge denne metode:

```
20 GETKEY A$: IF ASC(A$) = 133 THEN PRINT "FUNKTIONSTAST 1
RAMT: GOTO 20
30 IF ASC(A$) >133 AND ASC(A$) <141 THEN PRINT "EN ANDEN
FUNKTIONSTAST RAMT"
40 GOTO 20
```

Efter at dit program er udført, må du omdefinere tasterne igen, hvis du ønsker, at de skal sige directory, dload o.s.v. Du kan gøre det manuelt, i et program eller ved at nulstille **C 16-plus/4.**

Fejlmeddelelser

Disse fejlmeddelelser udskrives af BASIC. Man kan også udskrive meddelelsen ved brug af ERR$ funktionen. Fejltallet refererer kun til det tal, der er tildelt fejlen til brug med denne funktion.

FEJL NR.	FEJLNAVN	
1	TOO MANY FILES	Der kan højst være 10 filer åbne ad gangen.
2	FILE OPEN	Forsøg på åbning af en fil ved brug af nummeret på en allerede åben fil.
3	FILE NOT OPEN	Filnummeret angivet i en I/0 instruks skal åbnes før brug.
4	FILE NOT FOUND	Enten er der ingen fil med det nummer (diskette) eller også læstes et slut på bånd mærke (bånd).
5	DEVICE NOT PRESENT	Den nødvendige I/O enhed er ikke tilgængelig.
6	NOT INPUT FILE	Forsøg på at FÅ eller INDLÆSE data fra en fil, der kun var angivet som uddata fil.
7	NOT OUTPUT FILE	Forsøg på at sende data til en fil der kun var angivet som inddata fil.
8	MISSING FILE NAME	Instrukserne OPEN, LOAD eller SAVE til diskettedrevet kræver normalt et filnavn.
9	ILLEGAL DEVICE NUMBER	Forsøg på at anvende en enhed forkert (SAVE instruks til skærmen o.s.v.)
10	NEXT WITHOUT FOR	Enten sammenflettes løkker ukorrekt eller også er der et variabelnavn i en NEXT instruks, der ikke passer til et i en FOR instruks.
11	SYNTAX	En instruks kan ikke genkendes af BASIC. Dette kan skyldes manglende eller ekstra parenteser, forkert stavede nøgleord o.s.v.
12	RETURN WITHOUT GOSUB	Forekomst af en RETURN instruks uden aktiv GOSUB instruks.
13	OUT OF DATA	En READ instruks uden tilbageværende ulæste data.
14	ILLEGAL QUANTITY	Et tal anvendt som argument i en funktion eller instruks uden for det tilladelige område.

15	OVERFLOW	Resultatet af en udregning er større end det størst tilladte tal (1.701411833E+38).
16	OUT OF MEMORY	Enten er der ikke mere plads til programmer og programvariabler, eller der er for mange DO, FOR eller GGSUB instrukser som resultat.
17	UNDEF'D STATEMENT	Et linienummer, hvortil der er henvist, eksisterer ikke i programmet.
18	BAD SUBSCRIPT	Programmet forsøgte henvisning til et element i et sæt uden for det område, der er angivet af DIM instruksen.
19	REDIM'D ARRAY	Et sæt kan kun dimensioneres en gang. Hvis der henvises til et sæt, før det er dimensioneret, udføres en automatisk DIM instruks (til 10).
20	DIVISION BY ZERO	Division med nul er ikke tilladt.
21	ILLEGAL DIRECT	INPUT eller GET instrukser er kun tilladt i et program.
22	TYPE MISMATCH	Forekommer hvor et tal bruges i stedet for en streng eller omvendt.
23	STRING TOO LONG	En streng kan indeholde indtil 255 karakterer.
24	FILE DATA	Forkerte data indlæst fra en båndfil.
25	FORMULA TOO COMPLEX	Gør udtrykket enklere (del det i to dele eller brug færre parenteser)
26	CAN'T CONTINUE	CONT kommandoen fungerer ikke, dersom programmet ikke KØRTES, eller hvis der var en fejl eller en linie er blevet redigeret.
27	UNDEF'D FUNCTION	Funktion er ikke defineret.
28	VERIFY	Programmet på bånd eller diskette svarer ikke til programmet i hukommelsen.
29	LOAD	Der var problemer med indlæsningen. Prøv igen.
30	BREAK	Der blev trykket på stoptasten for at standse udførelsen af programmet.
31	CAN'T RESUME	En RESUME instruks mødes, uden at der er en virksom TRAP instruks.
32	LOOP NOT FOUND	Programmet har mødt en DO instruks og kan ikke finde den dertil svarende LØKKE.

33 LOOP WITHOUT DO	LØKKE uden aktiv DO instruks.
34 DIRECT MODE ONLY	Denne kommando tillades kun i direkte tilstand, ikke fra et program.
35 NO GRAPHICS AREA	En kommando (DRAW, BOX, etc.) til frembringelse af grafiske tegn mødt, før GRAPHIC kommandoen blev udført.
36 BAD DISK	Et forsøg på at FORMATTERE en diskette slog fejl, fordi den hurtige formatteringsmetode (no ID) blev forsøgt på en uformatteret diskette, eller disketten er defekt.

Beskrivelse af DO's error meddelelser

Disse meddelelser fremkommer via DS og DS$ reserverede variabler.

BEMÆRK: Fejlmeddelelser med numre lavere end 20 bør ignoreres med undtagelse af 01, som giver oplysning om det antal filer, der er slettet ved brug af SCRATCH kommandoen.

20 READ ERROR (blokformat ikke fundet)	Diskette controlleren er ikke i stand til at lokalisere formatet på den forlangte data blok. Skyldes et ulovligt sektornummer, eller at formatet er ødelagt.
21 READ ERROR (ingen sync karakter)	Diskette controlleren er ude af stand til at finde et sync mærke på det ønskede spor. Skyldes fejljustering af læse/skrivehovedet, manglende diskette eller uformatteret eller ukorrekt lukket diskette. Kan også betyde hardwarefejl.
22 READ ERROR (manglende datablok)	Diskette controlleren er blevet bedt om at læse eller verificere en data blok, der ikke var korrekt skrevet. Denne fejlmeddelelse fremkommer i forbindelse med BLOCK kommandoerne og betegner et spor og/eller sektorforlangende, der er ulovligt.

23	READ ERROR (kontrolsumfejl i data blok)	Denne fejlmeddelelse angiver, at der er fejl i en eller flere data bytes. Data er indlæst i DOS hukommelsen, men kontrolsummen over data er forkert. Denne meddelelse kan også betyde problemer med jordforbindelsen.
24	READ ERROR (byte tolkningsfejl)	Data eller format er indlæst i DOS hukommelsen, men der er opstået maskinfejl på grund af et ugyldigt bitmønster i data byten. Denne meddelelse kan også betyde problemer med jordforbindelsen.
25	WRITE ERROR (skrive-verificer fejl)	Denne meddelelse fremkommer, hvis controlleren opdager en uoverensstemmelse mellem de skrevne data og data i DOS hukommelsen.
26	WRITE PROTECT ON	Denne meddelelse fremkommer, når controlleren er blevet bedt om at skrive en data blok medens "write protect" kontakten er nedtrykket. Det sker typisk, når der anvendes en diskette med en "write protect" mærkat påsat over skriveslidsen.
27	READ ERROR (kontrolsumfejl i format)	Controlleren har opdaget en fejl i den forlangte data bloks format. Blokken er ikke blevet indlæst i DOS hukommelsen. Denne meddelelse kan også betyde problemer med jordforbindelsen.
28	WRITE ERROR (lang data blok)	Controlleren forsøger at opdage sync mærket på næste format efter at have skrevet en data blok. Hvis sync mærket ikke fremkommer inden for et forud fastsat tidspunkt, frembringes fejlmeddelelsen. Fejlen skyldes et forkert disketteformat (data strækker sig ind i næste blok) eller en maskinfejl.
29	DISKETTE ID MISMATCH	Denne meddelelse fremkommer, når controlleren er blevet bedt om at gå ind i en diskette, der ikke er blevet initialiseret. Meddelelsen kan også fremkomme, hvis en diskette har et forkert format.

30	SYNTAX ERROR (generel syntaks)	DOS kan ikke oversætte den kommando, der er sendt til kommando kanalen. Dette skyldes typisk et ulovligt antal filnavne eller ulovlig brug af mønstre. Der kan f.eks. forekomme to filnavne på venstre side af COPY kommandoen.
31	SYNTAX ERROR (ugyldig kommando)	DOS genkender ikke kommandoen. Kommandoen skal begynde i første position.
32	SYNTAX ERROR (ugyldig kommando)	Den afsendte kommando er længere end 58 karakterer.
33	SYNTAX ERROR (ugyldigt filnavn)	Mønster matching anvendes ugyldigt i OPEN eller SAVE kommandoen.
34	SYNTAX ERROR (ingen fil opgivet	Filnavnet udeladt i en kommando eller DOS genkender det ikke som sådant. Der er typisk tale om at et kolon (:) er udeladt i kommandoen.
39	SYNTAX ERROR (ugyldig kommando)	Denne fejl kan blive resultatet, hvis den kommando, der er sendt til kommandokanalen (sekundær adresse 15), ikke genkendes af DOS.
50	RECORD NOT PRESENT	Resultatet af diskettelæsning, efter sidste post gennem INPUT#, eller GET# kommandoerne. Denne meddelelse vil også forekomme efter positionering til en post efter slut på filen i en relativ fil. Hvis hensigten er at udvide filen ved at tilføje den nye post (med en PRINT# kommando), kan fejl meddelelsen ignoreres. INPUT eller GET bør ikke forsøges, efter at denne fejl er opdaget, uden først at ompositionering.
51	OVERFLOW IN RECORD	Print# instruks overskrider grænsen for posten, oplysninger afskæres. Eftersom vognretur, som sendes som slut på posten, tæller med i poststørrelsen, vil denne meddelelse fremkomme, hvis de samlede karakterer i posten (inkl. den sidste vognretur) overskrider den definerede størrelse.

52	FILE TOO LARGE	En posts placering i en relativ fil indikerer, at resultatet vil blive overfyldning af disketten.
60	WRITE FILE OPEN	Denne meddelelse fremkommer, når en skrivefil, der ikke er blevet lukket, åbnes for indlæsning.
61	FILE NOT OPEN	Denne meddelelse fremkommer, når man går ind i en fil, der ikke er blevet åbnet i DOS. I dette tilfælde sker det somme tider, at der ikke fremkommer nogen meddelelse; anmodningen ignoreres simpelthen.
62	FILE NOT FOUND	Den forlangte fil findes ikke i det indikerede drev.
63	FILE EXISTS	Filnavnet på den fil, der er under fremstilling, findes i forvejen på disketten.
63	FILE TYPE MISMATCH	Filtypen svarer ikke til filtypen i index starten på den forlangte fil.
65	NO BLOCK	Denne meddelelse fremkommer i forbindelse med B-A kommandoen. Den indikerer, at den blok, der skal allokeres er blevet allokeret tidligere. Parametrene indikerer det spor og den sektor der er til rådighed med det næsthøjeste nummer. Hvis parametrene er nul (0), er alle blokke med højere nummer i brug.
66	ILLEGAL SYSTEM AND SECTOR	DOS har forsøgt at gå ind i et spor eller en blok, der ikke eksisterer i det anvendte format. Dette kan indikere et problem m.h.t. at læse adressen til næste blok.
67	ILLEGAT SYSTEM T OR S	Denne specielle fejlmeddelelse indikerer et ulovligt systemspor eller en ulovlig systemsektor.
70	NO CHANNEL (ledig)	Den forlangte kanal er ikke ledig, eller alle kanalerne er i brug. Højst fem sekventielle filer kan åbnes på en gang til DOS. Kanaler for direkte tilgang kan have seks åbne filer.

71 DIRECTORY ERROR	BAM svarer ikke til den interne tællerstand. Der er et problem i BAM allokeringen eller BAM er overskrevet i DOS hukommelsen. For at rette dette ominitialiseres disketten for genlagring af BAM i hukommelsen. Nogle aktive filer kan sluttes med rettefunktionen. BEMÆRK: BAM = BLOCK AVAILABILITY MAP.
72 DISK FULL	Enten er blokkene på disketten brugt eller adressen har nået sin indtastningsgrænse. DISK FULL sendes, når der er to blokke til rådighed på 1541, for at tillade lukning af den igangværende fil.
73 DOS MISMATCH (73, CBM DOS V2 6 1541)	DOS 1 og 2 er læsekompatible, men ikke skrivekompatible. Disketter kan ombyttes indbyrdes for læsning med begge DOS, men en diskette, der er formatteret i den ene version, kan der ikke skrives på med den anden version, fordi formatet er anderledes. Denne fejl vises, når der gøres forsøg på at skrive på en diskette, der er blevet formatteret i et ikke kompatibelt format. (En servicerutine er til rådighed til hjælp ved konvertering fra et format til et andet). Denne meddelelse kan også forekomme efter, at der er tændt for computeren.
74 DRIVE NOT READY	Der er gjort forsøg på at bruge diskettedrevet, uden at der er isat diskette.

Tedmon

INTRODUKTION

TEDMON er et indbygget maskinkodeprogram, som gør det muligt for dig at skrive maskinkodeprogrammer uden vanskelighed. TEDMON omfatter en maskinkodemonitor, en assembler og en disassembler.

Maskinkodeprogrammer skrevet ved brug af TEDMON kan køre selv, eller de kan bruges som meget hurtige "underrutiner" til BASIC programmer, da TEDMON har evnen til fredelig sameksistens med BASIC.

TEDMON KOMMANDOER

A	ASSEMBLE	Oversætter en linie i kode 6502.
C	COMPARE	Sammenligner to sektioner i hukommelsen og meddeler forskelle.
D	DISASSEMBLE	Disassemblerer er en linie i kode 6502.
F	FILL	Fylder hukommelsen med den angivne byte.
G	GO	Begynder udførelse ved den anførte adresse.
H	HUNT	Gennemsøger hukommelsen for alle forekomster af visse bytes.
L	LOAD	Indlæser en fil fra bånd eller diskette.
M	MEMORY	Viser hukommelsesadressernes hexadecimale værdier.
R	REGISTERS	Viser 6502 registrene.
S	SAVE	Lagrer på bånd eller diskette.
T	TRANSFER	Overfører kode fra en sektion i hukommelsen til en anden.
V	VERIFY	Sammenligner hukommelsen med bånd eller diskette.
X	EXIT	Udgang TEDMON.
.	(PUNKTUM)	Oversætter en linie i kode 6502.
>	(større end)	Ændrer hukommelsen.
;	(semikolon)	Ændrer visninger af 6502 registrene.

Adressen $7F8 kontrollerer om TEDMON ser på ROM eller RAM over $8000. Hvis denne adresse sættes til 0, viser TEDMON BASIC og KERNAL, når der gives ordre til disassemblering eller udskrivning af hukommelsen over $8000.Hvis denne adresse sættes til $80, viser TEDMON RAM under BASIC og KERNAL. Dette er ofte nyttigt ved udvikling af maskinsprogsprogrammer. Bemærk at adressen $7F8 ikke indvirker på GO kommandoen. GO kommandoen starter udførelsen i den øjeblikkelige hukommelsesoversigt (ROM eller RAM) uafhængigt af indstillingen af adressen $7F8.

BRUG AF TEDMON

Indtast TEDMON ved at skrive:
MONITOR
TEDMON svarer ved at vise 6502 registrene og markøren blinker. Markøren er dit stikord, der meddeler, at TEDMON venter på dine kommandoer.

KOMMANDOBESKRIVELSER

KOMMANDO: A

FORMÅL: Indtast en linie i assembler kode.

SYNTAKS: A <adresse> <opcode mnemonic> <operand>

<adresse> Et hexadecimalt tal, der angiver den position i hukommelsen, hvor opcode (operationskode) skal anbringes.

<opcode mnemonic> En standard assembler mnemonic for maskinsprog i MOS teknologi, f.eks. LDA, STX, ROR, o.s.v.

<operand> Når den er nødvendig, kan operanden være en hvilken som helst af de lovlige adresseringsmåder. (Ved zero-page tilstande er et 2-cifret hex tal et, hvis værdi er mindre end $100. I adresser uden zero page kræves 4-cifrede hex.tal).

En RETURN bruges til at angive slutningen på assemblerlinien. Hvis der forekommer fejl på linien, vises et spørgsmålstegn for at angive, at der er en fejl, og markøren flytter sig til næste linie. Rettelser foretages på samme måde som rettelser af BASIC programmer.
Efter at en kodelinie er korrekt oversat, udskriver assembler et stikord, der indeholder næste lovlige hukommelsesadresse for en instruks, derfor behøver man ikke at indtaste A og linienummeret mere end én gang, når man indtaster assembler-sprog programmer på **C 16-plus/4**.

EKSEMPEL:

```
.A 1200 LDX $00
.A 1202
```

BEMÆRK: Et punktum (.) svarer til kommandoen ASSEMBLE.

EKSEMPEL:

.2000 LDA $23

KOMMANDO: C

FORMÅL: Sammenligning af to områder i hukommelsen

SYNTAKS: C <adresse 1> <adresse 2> <adresse 3>

<Adresse 1> er et hexadecimalt tal, der angiver startadressen i det område af hukommelsen, der skal sammenlignes med.
<Adresse 2> er et hexadecimalt tal, der angiver slutadressen i det område af hukommelsen, der skal sammenlignes med.
<Adresse 3> er et hexadecimalt tal, der angiver startadressen i det andet område i hukommelsen, der skal sammenlignes med.

Hvis de to områder i hukommelsen er det samme, udskriver TEDMON en RETURN, der angiver, at det andet område i hukommelsen er det samme som det første. Adresserne på eventuelle bytes i de to områder, der er forskellige, udskrives på skærmen.

KOMMANDO: D

FORMÅL: Disassemblerer maskinkode til assemblersprog mnemonic og operander.

SYNTAX: D [<adresse>] [<adresse 2>]

<adresse> Et hexadecimalt tal, der indsætter adressen for påbegyndelse af disassemblering.
<adresse 2> En valgfri hexadecimal slutadresse i kode til disassemblering.

Formatet for disassemblering er kun lidt forskelligt fra input formatet for en assemblering. Forskellen er, at den første karakter i en disassemblering er et komma og ikke et A (for læselighedens skyld), og kodens hexadecimal listes også.

En disassemblerings listning kan ændres ved hjælp af skærmredigeringen. Foretag en eventuel ændring af mnemonic eller operanden på skærmen og tryk derefter på vognreturtasten. Derved indtastes linien og assembler kaldes for yderligere ændringer.

En disassemblering kan sideopdeles. Indtastning af et D får næste side i disassemblering til at komme frem på skærmen.

EKSEMPEL

```
D 3000 3004
.3000 A900   LDA $00
.3002 FF     ???
.3003 DO 2B BNE $3030
```

KOMMANDO: F (FILL)

FORMÅL — Udfyldning af en række pladser med en angivet byte.

SYNTAKS: — F <adresse 1> <adresse 2> <byte>

<adresse 1> Den første plads der skal udfyldes med <byten>
<adresse 2> Den sidste plads der skal udfyldes med <byten>
<byte value> Et en- eller tocifret hexadecimalt tal, der skal skrives.

Denne kommando er nyttig ved initialisering af data strukturer eller et hvilket som helst andet RAM område.

EKSEMPEL: — F 0400 0518 EA

Udfylder hukommelsespladserne fra $0400 til $0518 med $EA (en NOP (ingen funktion) instruks).

KOMMANDO: G

FORMÅL: — Påbegynder udførelsen af et program på en angivet adresse.

SYNTAKS: — G [<adresse>]

<adresse> Et valgfrit argument, der angiver programtællerens nye værdi samt adressen, hvor udførelsen skal starte. Når <adresse> udelades, begynder udførelsen ved PC's øjeblikkelige stilling (den øjeblikkelige PC stilling fås ved hjælp af R kommandoen).
GO kommandoen nulstiller alle registre (der kan vises ved hjælp af R kommandoen) og påbegynder udførelsen ved den angivne startadresse. GO kommandoen bør bruges med forsigtighed. For at komme tilbage til TEDMON efter udførelse af et maskinsprogsprogram, bruger man BRK instruksen..

EKSEMPEL: G 140C

Udførelsen starter på positionerne $ 140C.

KOMMANDO: H (HUNT)

FORMÅL: Gennemsøgning af et nærmere angivet område i hukommelsen efter alle forekomster af et sæt bytes.

SYNTAKS: H <adresse 1> <adresse 2> <data>

<adresse 1> begyndelsesadresse for gennemsøgningsproceduren.
<adresse 2> slutadresse for gennemsøgningsproceduren.
<data> det datasæt, der skal søges efter. Data kan være hexadecimale eller en ASCII streng. En ASCII angives ved, at man sætter en apostrof foran den første karakter, f.eks. 'STRENG. Data kan være argumenter bestående af et eller flere elementer. Når der er tale om argumenter bestående af flere elementer eller om hexadecimaler, skal de enkelte tal adskilles med et mellemrum.

EKSEMPEL: H C000 FFFF 'READ Søg efter ASCII streng, læs fra C000 til FFFF
H A000 A101 A9 FF 4C Søg efter data $A9, $ff, $4C, fra A100 til A101

KOMMANDO: L (LOAD)

FORMÅL: Indlæs en fil fra kasette eller diskette.
SYNTAKS: L <"filnavn">, <enhed>

<filnavn> er ethvert lovligt **C 16-plus/4** filnavn i anførselstegn.
<enhed> er et hexadecimalt tal, der angiver den enhed, der skal indlæses fra.

1 er kassette
8 er diskette (eller 9, A, o.s.v.)

Load kommandoen resulterer i indlæsning af en fil i hukommelsen. De to første bytes i filen indeholder startadressen (en programfil). Med andre ord, LOAD kommandoen indlæser altid en fil på den plads, hvorfra den blev lagret. Dette er meget vigtigt, når man arbejder med maskinsprog, da kun få programmer er fuldstændigt relocatable (kan placeres og udføres

overalt i et sammenhængende område i hukommelsen). Filen indlæses i hukommelsen indtil end of file (EOF) dukker op.

EKSEMPEL: L "BILLEDE", 1 indlæser en fil fra kassette

L "TANK", 8 indlæser en fil fra diskette

KOMMANDO: M (MEMORY DISPLAY)

FORMÅL: At vise hukommelsen som et hexadecimalt tal og ASCII udskrift inden for det angivne adresseområde.

SYNTAKS: M [<adresse 1>] [<adresse>]

[<adresse 1>] Første adresse i hukommelsesudskriften. Valgfri. Hvis den udelades, vises én side. Den første byte er den sidst angivne adresse.
[<adresse 2>] Sidste adresse i hukommelsesudskriften. Valgfri. Hvis den udelades, vises én side. Den første byte er data i <adresse 1>.

Hukommelsen vises i følgende format:
>A048 41 E7 00 AA AA 00 98 56 45 :A1.*..VE

Indholdet af hukommelsen kan redigeres ved hjælp af skærmredigeringen. Flyt markøren til de data, der skal ændres og indtast den ønskede ændring og tryk på return. Hvis der er en forkert RAM position eller der gøres forsøg på at ændre ROM, vises en fejlmarkering (?).

En ASCII udskrift af data vises i NEGATIV skrift (for at danne modsætning til andre data, der vises på skærmen) til højre for hex data. Hvis en karakter ikke kan udskrives, vises den som et punktum (.) i negativ skrift. Ligesom ved Disassembleringskommandoen kan du sideopdele ved at indtaste M og trykke på RETURN.

EKSEMPEL
M 1C00

>1C00 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C08 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C10 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C18 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C20 41 E7 00 AA AA 00 98 56 45 :A!.*..VE

```
>1C28 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C30 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C38 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C40 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C48 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C50 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
>1C58 41 E7 00 AA AA 00 98 56 45 :A!.*..VE
```

KOMMANDO: > (tegnet for STØRRE END)

FORMÅL: Kan bruges til at sætte 1 til 8 hukommelsesoperationer på én gang.

SYNTAKS: ><adresse><data byte 1><data byte 2 ... 8>

adresse: Første hukommelsesadresse, der skal sættes.
data byte 1: Data, der skal sættes på adressen.
<data byte 2 til data byte 8>: Data, der skal placeres i de efterfølgende hukommelsespositioner, regnet fra den første adresse. Valgfri.

EKSEMPEL: >2000 08 anbringer tallet 08 i positionen 2000
>3000 23 45 65 placerer tallet 23 i positionen 3000, tallet 45 i positionen 3001 og tallet 65 i positionen 3002.

KOMMANDO: R (REGISTER DISPLAY)

FORMÅL: Visning af vigtige 6502 registre. Programstatus registreret, programtælleren, akkumulator registeret, X og Y index registrene og stack pointer registeret vises.

SYNTAKS: R

EKSEMPEL:

```
R
PC   SR  AC XR YR SP
1002 01  02  03  04  F6
```

BEMÆRK: ; (semikolon kan bruges til ændring af registervisninger på samme måde som >tegnet kan bruges til ændring af hukommelsesregistre).

KOMMANDO: S (SAVE)

FORMÅL: Indholdet af hukommelsen lagres på bånd eller diskette.

SYNTAKS: S <"filnavn">, <enhed>, <adresse 1>, <adresse 2>

<"filnavn"> Et hvilket som helst lovligt **C 16-plus/4** filnavn. For lagring af data skal filnavnet sættes i anførselstegn. Apostrof kan ikke bruges.
<enhed> To mulige enheder er kassette og diskette. For lagring på kassette bruges enhed 1: Enhedsnummeret på **C 16-plus/4** diskettedrev er sædvanligvis 8. Det kan imidlertid ændres (når man for eksempel bruger mere end et diskettedrev). Se **C 16-plus/4** DISKETTEDREV BRUGERVEJLEDNING.
<adresse 1> Startadresse på den hukommelse, der skal lagres.
<adresse 2> Slutadresse på den hukommelse, der skal lagres +1. Alle data indtil, men ikke inklusive data byten på denne adresse, lagres.

Den fil, der laves ved hjælp af denne kommando, er en programfil. De to første bytes indeholder den stående adresse <adresse 1> for data. Filen kan kaldes ved hjælp af L kommandoen.

EKSEMPEL: X "GAME", 8, 0400, 0C00

Lagrer hukommelse fra $0400 til 0BFF på diskette.
KOMMANDO: T (TRANSFER)

FORMÅL: Overfører hukommelsessegmenter fra et område i hukommelsen til et andet.

SYNTAKS: T <adresse 1> <adresse 2> <adresse 3>

<adresse 1> Startadresse på data der skal flyttes.
<adresse 2> Slutadresse på data der skal flyttes.
<adresse 3> Startadresse på ny position (hvortil data flyttes)

Data kan flyttes fra lave til høje hukommelsesadresser og vice versa. Yderligere hukommelsessegmenter af enhver længde kan flyttes et hvilket som helst antal bytes frem eller tilbage.

EKSEMPEL: T 1400 1600 1401

Flytter data fra $1400 til og med $1600 en byte højere i hukommelsen.

KOMMANDO: V (VERIFY)

FORMÅL: Verificerer en fil på kassette eller diskette med hukommelsens indhold.

SYNTAKS: V <"filnavn">, <enhed>

<filnavn> er et hvilket som helst lovligt **C 16-plus/4** filnavn.
<enhed> er et hexadecimalt tal, der angiver hvilken enhed filen befinder sig på; kassette er 1 eller 01, diskette er 8 eller 08, 09 o.s.v.

Verify kommandoen sammenligner en fil med indholdet af hukommelsen. **C 16-plus/4** svarer med VERIFYING. Hvis en fejl findes, tilføjes ordet ERROR; hvis filen verificeres, kommer den blinkende markør atter til syne.

EKSEMPEL: V "ARBEJDSPRG",8

KOMMANDO: X (EXIT)

BASIC 3.5 Forkortelser

ABS	a	SHIFT	B	numerisk-funktion
ASC	a	SHIFT	S	numerisk-funktion
ATN	a	SHIFT	T	numerisk-funktion
AUTO	a	SHIFT	U	kommando
BACKUP	b	SHIFT	A	kommando
BOX	b	SHIFT	O	instruks
CHAR	ch	SHIFT	A	instruks
CHR$	c	SHIFT	H	streng-funktion
CIRCLE	c	SHIFT	I	instruks
CLOSE	cl	SHIFT	O	instruks
CLR	c	SHIFT	L	instruks
CMD	c	SHIFT	M	instruks
COLLECT	col	SHIFT	L	kommando
COLOR	co	SHIFT	L	instruks
CONT	c	SHIFT	O	kommando
COPY	co	SHIFT	P	kommando
COS		none	P	numerisk-funktion
DATA	d	SHIFT	A	instruks
DEC		none		numerisk-funktion
DEF FN	d	SHIFT	E	instruks
DELETE	de	SHIFT	L	kommando
DIM	d	SHIFT	I	instruks
DIRECTORY	di	SHIFT	R	kommando
DLOAD	d	SHIFT	L	kommando
DO		none		instruks
DRAW	d	SHIFT	R	instruks
DSAVE	d	SHIFT	S	kommando
END	e	SHIFT	N	instruks
ERR$	e	SHIFT	R	streng-funktion
EXP	e	SHIFT	X	numerisk-funktion
FOR	f	SHIFT	O	instruks
FRE	f	SHIFT	R	numerisk-funktion
GET	g	SHIFT	E	instruks
GETKEY	getk	SHIFT	E	instruks
GET#		none		instruks
GOSUB	go	SHIFT	S	instruks
GOTO	g	SHIFT	O	instruks
GRAPHIC	g	SHIFT	R	instruks
GSHAPE	g	SHIFT	S	instruks
HEADER	he	SHIFT	A	kommando
HEX$	h	SHIFT	E	streng-funktion

IF ... GOTO		none		instruks
IF ... THEN ... ELSE		none		instruks
INPUT		none		instruks
INPUT#	i	SHIFT	N	instruks
INSTR	in	SHIFT	S	numerisk-funktion
INT		none		numerisk-funktion
JOY	j	SHIFT	O	numerisk-funktion
KEY	k	SHIFT	E	kommando
LEFT$	le	SHIFT	F	streng-funktion
LEN		none		numerisk-funktion
LET	l	SHIFT	E	instruks
LIST	l	SHIFT	I	kommando
LOAD	l	SHIFT	O	kommando
LOCATE	lo	SHIFT	C	instruks
LOG		none		numerisk-funktion
LOOP	lo	SHIFT	O	instruks
MID$	m	SHIFT	I	streng-funktion
MONITOR	m	SHIFT	O	instruks
NEW		none		kommando
NEXT	n	SHIFT	E	instruks
ON ... GOSUB on	go	SHIFT	S	instruks
ON ... GOTO on	g	SHIFT	O	instruks
OPEN	o	SHIFT	P	instruks
PAINT	p	SHIFT	A	instruks
PEEK	p	SHIFT	E	numerisk-funktion
POKE	p	SHIFT	O	instruks
POS		none		numerisk-funktion
PRINT	?			instruks
PRINT#	p	SHIFT	R	instruks
PRINT USING	?us	SHIFT	I	instruks
PUDEF	p	SHIFT	U	instruks
RCLR	r	SHIFT	C	numerisk-funktion
RDOT	r	SHIFT	D	numerisk-funktion
READ	r	SHIFT	E	instruks
REM		none		instruks
RENAME	re	SHIFT	N	kommando
RENUMBER	ren	SHIFT	U	kommando
RESTORE	re	SHIFT	S	instruks
RESUME	res	SHIFT	U	instruks
RETURN	re	SHIFT	T	instruks
RGR	r	SHIFT	G	numerisk-funktion
RIGHT$	r	SHIFT	I	streng-funktion
RLUM	r	SHIFT	L	numerisk-funktion

RND	r	SHIFT	N	numerisk-funktion
RUN	r	SHIFT	U	kommando
SAVE	s	SHIFT	A	kommando
SCALE	sc	SHIFT	A	instruks
SCNCLR	s	SHIFT	C	instruks
SCRATCH	sc	SHIFT	R	kommando
SGN	s	SHIFT	G	numerisk-funktion
SIN	s	SHIFT	I	numerisk-funktion
SOUND	s	SHIFT	O	instruks
SPC(	s	SHIFT	P	special-funktion
SQR	s	SHIFT	Q	numerisk-funktion
SSHAPE	s	SHIFT	S	instruks
STatus		none		reserveret variabel
STOP	s	SHIFT	T	instruks
STR$	st	SHIFT	R	streng-funktion
SYS	s	SHIFT	Y	instruks
TAB(	t	SHIFT	A	special-funktion
TAN		none		numerisk-funktion
TI		none		reserveret variabel
TI$		none		reserveret variabel
TRAP	t	SHIFT	R	instruks
TROFF	tro	SHIFT	F	instruks
TRON	tr	SHIFT	O	instruks
UNTIL	u	SHIFT	N	instruks
USR	u	SHIFT	S	special-funktion
VAL		none		numerisk-funktion
VERIFY	v	SHIFT	E	kommando
VOL	v	SHIFT	O	instruks
WAIT	w	SHIFT	A	instruks
WHILE	w	SHIFT	H	instruks

Nodeværdier

NODE	SOUND REGISTER VÆRDI	FREKVENS (HZ)
A	7	110
B	118	123.5
C	169	130.8
D	162	146.8
E	345	164.7
F	383	174.5
G	453	195.9
A	516	220.2
B	571	246.9
C	596	261.4
D	643	293.6
E	685	330
F	704	349.6
G	739	392.5
A	770	440.4
B	798	494.9
C	810	522.7
D	834	588.7
E	854	658
F	864	699
G	881	782.2
A	897	880.7
B	911	989.9
C	917	1045
D	929	1177
E	939	1316
F	944	1398
G	953	1575

Værdien i SOUND-kommandoen beregnes således:
SOUND REGISTER VÆRDI = 1024-(111860.781/FREKVENS)

Skærmkoder

Hvert tal i tabellen repræsenterer den værdi, der skal POKEs ind på skærmen (adresserne 3072-4095) for at få den ønskede karakter.

Hvis du ønsker tegnet udskrevet med negativ skrift, skal du lægge 128 til værdien før du POKEr.

SET1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	T	t	20	(		40
A	a	1	U	u	21	)		41
B	b	2	V	v	22	*		42
C	c	3	W	w	23	+		43
D	d	4	X	x	24	,		44
E	e	5	Y	y	25	–		45
F	f	6	Z	z	26	.		46
G	g	7	[		27	/		47
H	h	8	£		28	0		48
I	i	9	]		29	1		49
J	j	10	↑		30	2		50
K	k	11	←		31	3		51
L	l	12	SPACE		32	4		52
M	m	13	!		33	5		53
N	c	14	”		34	6		54
O	o	15	#		35	7		55
P	p	16	$		36	8		56
Q	q	17	%		37	9		57
R	r	18	&		38	:		58
S	s	19	’		39	;		59

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
<		60		T	84			108
=		61		U	85			109
>		62		V	86			110
?		63		W	87			111
		64		X	88			112
	A	65		Y	89			113
	B	66		Z	90			114
	C	65			91			115
	D	68			92			116
	E	69			93			117
	F	70			94			118
	G	71			95			119
	H	72	SPACE		96			120
	I	73			97			121
	J	74			98		✓	122
	K	75			99			123
	L	76			100			124
	M	77			101			125
	N	78			102			126
	O	79			103			127
	P	80			104			
	Q	81			105			
	R	82			106			
	S	83			107			

ASCII og CHR$ koder

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	0	↑	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
WHT	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
DISABLES SHIFT C=	8		25	*	42	;	59
ENABLES SHIFT C=	9		26	+	43	<	60
	10	ESCAPE	27	,	44	=	61
	11	RED	28	−	45	>	62
	12	→	29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
SWITCH TO LOWER CASE	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
D	68		97		126		155
E	69		98		127	PUR	156
F	70		99		128	←	157
G	71		100		129	YEL	158
H	72		101	FLASH ON	130	CYN	159
I	73		102	FLASH OFF	131	SPACE	160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	HELP	140		169
S	83		112	SHIFT RETURN	141		170
				SWITCH TO			
T	84		113	UPPER CASE	142		171
U	85		114		143		172
V	86		115	BLK	144		173
W	87		116	↓	145		174
X	88		117	RVS OFF	146		175
Y	89		118	CLR HOME	147		176
Z	90		119	INST DEL	148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183

RS232 Interface

Introduktion.

Din **Commodore plus/4** har et indbygget RS-232 interface for tilslutning til ethvert RS-232 modem, til en RS-232 printer, eller til andre RS-232 enheder. For at kunne anvende en RS-232 enhed til din **plus/4,** kræves der en niveau omformer (model 1101) og lidt ekstra programmering.

RS-232 på **plus/4** er standard RS-232, men spændings-niveauet er TTL-niveau (0 til 5V), i modsætning til den normale RS-232 (−12 til +12 V).

Niveau-omformeren mellem **Commodore plus/4** og RS-232 enheden foretager den nødvendige spændingsomformning.

RS-232 interfacet kan styres fra BASIC eller fra KERNALen med maskinkode programmering. Dette kapitel beskriver styring af RS-232 med BASIC. RS-232 styring med BASIC BRUGER DE NORMALE BASIC instruktioner: OPEN, CLOSE, CMD, INPUT#, GET#, PRINT# og den reserverede statusvariabel ST. INPUT# og GET# henter data fra den modtagende buffer, mens PRINT# og CMD placerer data i den transmitterende buffer.

Åbning af RS-232 kanalen.

Du bør kun have en RS-232 kanal åben af gangen; hvis du åbner een til, bliver bufferpointeren nulstillet, og dermed mister du alle data i bufferen.

Der kan sendes op til fire karakterer i filnavn feltet. De to første er kontrol og kommando karakterer; disse bruges til at sætte baud rate, paritet og andre attributter. De to sidste karakterer er reserveret til fremtidige system attributter.

BASIC Syntax:

OPEN lf,2,0,"<kontrol register> <kommando register>"

Eksempel:
OPEN 2,2,0,CHR$(5)+CHR$(15)

lf – normalt logisk filnummer (1-255). Hvis lf> 127 følger der et linieskift med en vognretur.

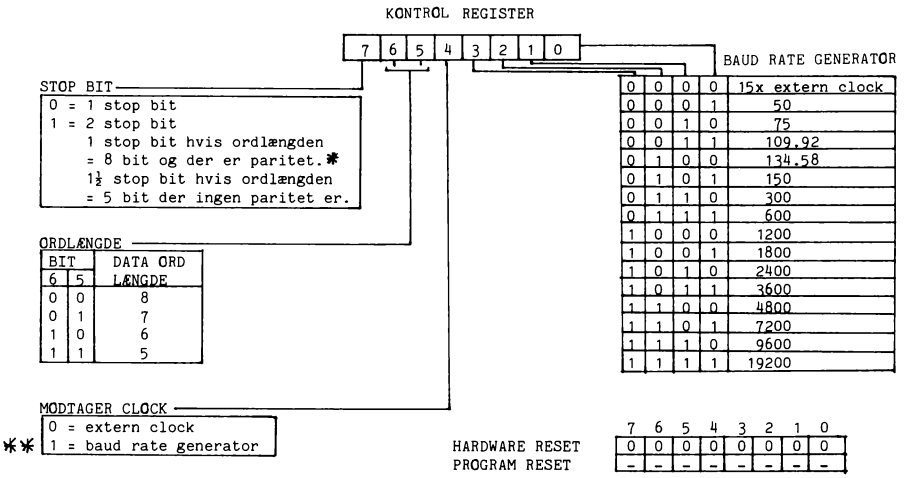
kontrol register – enkelt byte tegn (se fig. 1) (kræves til at specificere baud raten)

kommando register – enkelt byte tegn (se fig. 2)

Hente data fra RS-232 kanalen.

Når du modtager data, kan **Commodore plus/4**'s modtagerbuffer gemme op til 127 karakterer, før den løber over. Dette indikeres af RS-232 statusordet (ST i BASIC). Du mister alle karakterer, der modtages efter at bufferen er fyldt. Som du måske kan regne ud, kan det betale sig at holde bufferen så "tom" som muligt.

For at modtage RS-232 data ved høje hastigheder skal du bruge maskinkode, idet hastigheden på BASIC sætter en vis begrænsning.

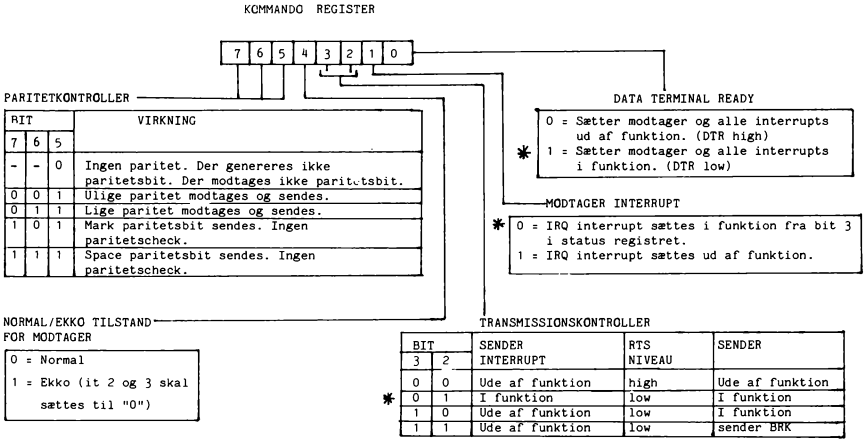


* Dette giver mulighed for 9-bit transmission (8 databit + paritet)

** Disse bit skal altid udfyldes, idet de ingen default værdi har.

BASIC Syntax:

Abefalet: GET#lf,<streng-variabel>
IKKE anbefalet: INPUT#lf,<variabelliste>



	7	6	5	4	3	2	1	0
HARDWARE RESET	0	0	0	0	0	0	0	0
PROGRAM RESET	-	-	-	0	0	0	0	0

* Disse bit skal altid udfyldes, idet de ingen default værdi har.

NOTER

Hvis ordlængden er mindre end 8 bit, får alle de ubrugte bit's værdien nul.

Hvis GET# ikke finder nogen data i bufferen, returneres en tom karakter (CHR$(0)).

Hvis du bruger INPUT#, "hænger" systemet indtil det modtager en karakter der er forskellig fra CHR$(0) (en "tom" karakter), efterfulgt af en vognretur (carriage return). Deraf følger, at hvis CTS eller DSR linierne forsvinder, mens der hentes data med INPUT#, "hænger" systemet, og kan kun oprettes ved tryk på RESET knappen. Derfor kan INPUT# og CHRIN rutinerne ikke anbefales.

Sende data til RS-232 kanalen.

BASIC Syntax:

CMD lf – virker på samme måde som i almindelig BASIC.
PRINT#lf,<variabelliste>

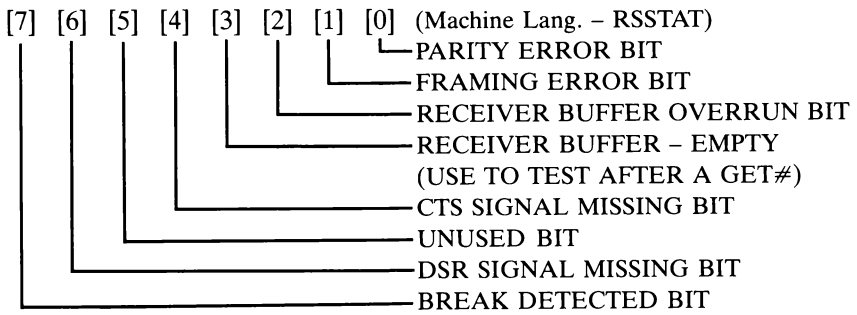
Lukning af RS-232 kanalen.

Når du lukker RS-232 kanalen, mister du alle de data, der ligger i bufferen, RS-232 kanalen stopper med at sende og modtage, og sætter RTS og Sout linierne høje.

BASIC Syntax:

CLOSE lf

PIN ID	DESCRIPTION	ABV	OUT
C	RECEIVED DATA	Sin	IN
D	REQUEST TO SEND	RTS	OUT
E	DATA TERMINAL READY	DTR	OUT
F	RING INDICATOR	RI	IN
H	RECEIVED LINE SIGNAL	DCD	IN
J	UNASSIGNED	XXX	IN
K	CLEAR TO SEND	CTS	IN
L	DATA SET READY	DSR	IN
B	RECEIVED DATA	Sin	IN
M	TRANSMITTED DATA	Sout	OUT
A	PROTECTIVE GROUND	GND	
N	SIGNAL GROUND	GND	



NOTER

Hvis den pågældende bit er lig nul, er der ikke fundet nogen fejl.

RS-232 statusregistret kan læses fra BASIC ved brug af systemvariablen ST.

Hvis ST læses fra BASIC eller fra KERNAL-rutinen READST, nulstilles RS-232 status registret straks efter læsning.
Hvis du har brug for status-registret i flere forbindelser, bør du gemme ST i en anden variabel, f.eks.:

```
SR = ST: REM GEMMER ST I SR
```

RS-232 status læses (og nulstilles) kun, når RS232 kanalen er den sidst brugte eksterne I/0.

Eksempel på BASIC program.

Dette program åbner telekommunikationskanalen, og gør dig i stand til at sende og modtage via et modem.

```
100 OPEN 5,2,2,CHR$(22)+CHR$(5)
110 DIM F%(255), T%(255)
120 FOR J = 32 TO 64:T%(J) = J:NEXT
130 T%(13) = 13:T%(20) = 8:RV = 18:CT = 0
220 FOR J = 65 TO 90:K = J+32:T%(J) = K:NEXT
230 FOR J = 91 TO 95:T%(J) = J:NEXT
240 FOR J = 193 TO 218:K = J-128:T%(J) = K:NEXT
250 T%(146) = 16:T%(133) = 16
255 T%(137) = 3:T%(134) = 17:T%(138) = 19
260 FOR J = 0 TO 255
270 K = T%(J)
280 IF K <>0 THEN F%(K) = J:F%(K+128) = j
290 NEXT
300 PRINT " "CHR$(147)
310 GET#5,A$
320 IF A$ = " " THEN 360
330 PRINT " "CHR$(157);CHR$(F%(ASC(A$)));
340 IF F%(ASC(A$)) = 34 THEN PRINT CHR$(27)"0";
350 GOTO 310
360 PRINT CHR$(RV)" "CHR$(157);CHR$(146);: GET A$
370 IF A$<>" " THEN PRINT#5,CHR$(T%(ASC(A$)));
380 CT = CT+1
390 IF CT = 8 THEN CT = 8:RV = 164-RV
400 GOTO 310
```

Prøveprogrammer

```
10 GRAPHIC 2,1
20 DO
30 DRAW1,1,10TO300,10
35 FORL=1T050:NEXTL
40 DRAW0,1,10TO300,10
45 FORL=1T050:NEXTL
50 LOOP
60 GRAPHIC0
```

```
 10 VOL7
 20 FORM=1TO20
 30 FORL=900TO850STEP-10
 40 SOUND1,L,1
 50 NEXT
 60 FORL=850TO900STEP10
 70 SOUND1,L,1
 80 NEXT
100 NEXT
```

```
  5 GRAPHIC3,1:GRAPHIC0,1
 20 INPUT "SKAL JEG ROTERE";B$
 30 INPUT "SKAL JEG VARIERER BEVÆGELSEN";C$
 40 INPUT "SKAL JEG VÆLGE START";D$
 60 DEF FNA(X)=INT(RND(1)*X)
 70 IF D$="J"THENX1=FNA(80)+80:X2=FNA(80)+80:Y1=FNA(100)+100
 75 IFD$="J"THENY2=FNA(100)+100
 80 IFD$<>"J"THENX1=80:X2=80:Y1=100:Y2=100
 90 GRAPHIC3:FORL=1TO3:COLORL,FNA(15)+2,FNA(8):NEXT
100 IF C1>1THEN COLORFNA(3)+1,FNA(15)+2,FNA(8):C1=FNA(40)+20
110 IFC2<>0THEN140:ELSEXA=FNA(11)-5:XB=FNA(11)-5
    :YA=FNA(13)-6
115 YB=FNA(13)-8
120 IFC$="J"THENC2=FNA(10)+5
130 IFB$="J"THENXB=-XA:YB=-YA
140 IFC3<1THENC=FNA(3)+1:C3=FNA(10)
150 X1=X1+XA:X2=X2+XB:Y1=Y1+YA:Y2=Y2+YB
160 IFX1<0ORX1>158THENXA=-XA:X1=X1+XA
170 IFX2<0ORX2>158THENXB=-XB:X2=X2+XB
180 IFY1<0ORY1>188THENYA=-YA:Y1=Y1+YA
190 IFY2<0ORY2>188THENYB=-YB:Y2=Y2+YB
200 DRAWC,X1,Y1TOX2,Y2
220 C1=C1-1:C2=C2-1:C3=C3-1:GOTO100
```

```
 10VOL7
 30 FORL=1TO100
 30 SOUND1,INT(RND(0)*500)+400,4
 40 NEXT
```

Commodore ted memory map

Label	Hex. Adr.	Decimal Adr.	Beskrivelse
PDIR	$0000	0	6510 on-chip data-direction register
PORT	$0001	1	6510 on-chip 8-bit Input/Output register
SRCHTK	$0002	2	Token 'search' looks for (run-time stack)
ZPVEC1	$0003-0004	3-4	Temp (renumber)
ZPVEC2	$0005-0006	5-6	Temp (renumber)
CHARAC	$0007	7	Search character
ENDCHR	$0008	8	Flag: scan for quote at end of string
TRMPOS	$0009	9	Screen column from last TAB
VERCK	$000A	10	Flag: 0 = load 1 = verify
COUNT	$000B	11	Input buffer pointer / No. of subscripts
DIMFLG	$000C	12	Flag: Default Array DIMension
VALTYP	$000D	13	Data type: $ff = string $00 = numeric
INTFLG	$000E	14	Data type: $80 = integer $00 = floating
DORES	$000F	15	Flag: DATA scan/LIST quote/garbage coll
SUBFLG	$0010	16	Glag: subscript ref / user function cal
INPFLG	$0011	17	Flag: $00 = INPUT $40 = GET $98 = READ
TANSGN	$0012	18	Flag: TAN sign / comparison result
CHANNL	$0013	19	Flag: INPUT prompt
LINNUM	$0014-0015	20-21	Temp: integer value
TEMPPT	$0016	22	pointer: temporary string stack
LASTPT	$0017-0018	23-24	Last temp string address
TEMPST	$0019-0021	25-33	Stack for temporary strings
INDEX1	$0022-0023	34-35	Utility pointer area
INDEX2	$0024-0025	36-37	Utility pointer Crea
RESHO	$0026	38	
RESMOH	$0027	39	
RESMO	$0028	40	
RESLO	$0029	41	
	$002A	42	
TXTTAB	$002B-002C	43-44	Pointer: start of BASIC text
VARTAB	$002D-002E	45-46	Pointer: start of BASIC variables
ARYTAB	$002F-0030	47-48	Pointer: start of BASIC arrays
STREND	$0031-0032	49-50	Pointer: end of BASIC arrays (+1)
FRETOP	$0033-0034	51-52	Pointer: bottom of string storage
FRESPC	$0035-0036	53-54	Utility string pointer
MEMSIZ	$0037-0038	55-56	Pointer: highest address used by BASIC
CURLIN	$0039-003A	57-58	Current BASIC line number
TXTPTR	$003B-003C	59-60	
FNDPNT	$003D-003E	61-62	
DATLIN	$003F-0040	63-64	Current DATA line number

DATPTR	$0041-0042	65-66	Pointer: current DATA item address
INPPTR	$0043-0044	67-68	Vector: INPUT routine
VARNAM	$0045-0046	69-70	Current BASIC variable name
VARPNT	$0047-0048	71-72	Pointer: current BASIC variable data
FORPNT	$0049-004A	73-74	Pointer: index variable for FOR/NEXT
OPPTR	$004B-004C	75-76	
OPMASK	$004D	77	
DEFPNT	$004E-004F	78-79	
DSCPNT	$0050-0051	80-81	
	$0052	82	
HELPER	$0053	83	
JMPER	$0054	84	
SIZE	$0055	85	
OLDOV	$0056	86	
TEMPF1	$0057	87	
HIGHDS	$0058-0059	88-89	
HIGHTR	$005A-005B	90-91	
	$005C	92	
LOWDS	$005D-005E	93-94	
LOWTR	$005F	95	
EXPSGN	$0060	96	
FACEXP	$0061	97	Floating-point accumulator #1: exponent
FACHO	$0062	98	Floating accum. #1: mantissa
FACMOH	$0063	99	
FACMO	$0064	100	
FACLO	$0065	101	
FACSGN	$0066	102	Floating accum. #1: sign
SGNFLG	$0067	103	Pointer: series evaluation constant
BITS	$0068	104	Floating accum. #1: overflow digit
ARGEXP	$0069	105	Floating-point accumulator #2: exponen
ARGHO	$006A	106	Floating accum. #2: mantissa
ARGMOH	$006B	107	
ARGMO	$006C	108	
ARGLO	$006D	109	
ARGSGN	$006E	110	Floating accum. #2: sign
ARISGN	$006F	111	Sign comparison result: accum. #1 vs #
FACOV	$0070	112	Floating accum. #1. low-order (rounding)
FBUFPT	$0071-0072	113-114	Pointer: cassette buffer
AUTINC	$0073-0074	115-116	Increment value for auto (0=off)
MUDFLG	$0075	117	Flag if 10K hires allocated
KEYNUM	$0076	118	
KEYSIZ	$0077	119	
SYNTMP	$0078	120	Used as temp for indirect loads

DSDESC	$0079-007B	121-123	Descriptor for ds$
TOS	$007C-007D	124-125	Top of run time stack
TMPTON	$007E-007F	126-127	Temps used by music (tone & volume)
VOICNO	$0080	128	
RUNMOD	$0081	129	
POINT	$0082	130	
GRAPHM	$0083	131	Current graphic mode
COLSEL	$0084	132	Current color selected
MC1	$0085	133	Multicolor1
FG	$0086	134	Foreground color
SCXMAX	$0087	135	Maximum # of columns
SCYMAX	$0088	136	Maximum # of rows
LIFLAG	$0089	137	Paint-left flag
RTFLAG	$008A	138	Paint-Right flag
STOPNB	$008B	139	Stop paint if not BG (Not same Color)
GRAPNT	$008C-008D	140-141	
VTEMP1	$008E	142	
VTEMP2	$008F	143	
STATUS	$0090	144	Kernal I/O status word: ST
STKEY	$0091	145	Flag: STOP key / RVS key
SPVERR	$0092	146	Temp
VERFCK	$0093	147	Flag: 0 = load 1 = verify
C3PO	$0094	148	Flag: serial bus – output char buffer
BSOUR	$0095	149	Buffered character for serial bus
XSAV	$0096	150	Temp for basin
LDTND	$0097	151	# of open files / index to file table
DFLTN	$0098	152	Default input device (0)
DFLTO	$0099	153	Default output (CMD) device (3)
MSGFLG	$009A	154	Flag: $80 = direct mode $00 = program
SAL	$009B	155	Tape pass 1 error log
SAH	$009C	156	Tape pass 2 error log
EAL	$009D	157	
EAH	$009E	158	
T1	$009F-00A0	159-160	Temp data area
T2	$00A1-00A2	161-162	Temp data area
TIME	$00A3-00A5	163-165	Real-time jiffy clock (approx) 1/60 sec
R2D2	$00A6	166	Serial bus usage
TPBYTE	$00A7	167	Byte to be written/read on/off tape
BSOUR1	$00A8	168	Temp used by serial routine
FPVERR	$00A9	169	
DCOUNT	$00AA	170	
FNLEN	$00AB	171	Length of current file name
LA	$00AC	172	Current logical file number
SA	$00AD	173	Current secondary address
FA	$00AE	174	Current device number
FNADR	$00AF-00B0	175-176	Pointer: current file name

ERRSUM	$00B1	177	
STAL	$00B2	178	I/O start address
STAH	$00B3	179	
MEMUSS	$00B4-00B5	180-181	Load ram base
TAPEBS	$00B6-00B7	182-183	Base pointer to cassette base
TMP2	$00B8-00B9	184-185	
WRBASE	$00BA-00BB	186-186	Pointer to data for tape writes
IMPARM	$00BC-00BD	188-189	Pointer to immediate string for primms
FETPTR	$00BE-00BF	190-191	Pointer to byte to be fetc hed in bank f
SEDSAL	$00C0-00C1	192-193	Temp for scrolling
RVS	$00C2	194	RVS field flag on
INDX	$00C3	195	
LSXP	$00C4	196	X position at start
LSTP	$00C5	197	
SFDX	$00C6	198	Flag: shift mode for print
CRSW	$00C7	199	Flag: INPUT og GET from keyboard
PNT	$00C8-00C9	200-201	Pointer: current screen line address
PNTR	$00CA	202	Cursor column on current line
QTSW	$00CB	203	Flag: editor in quote mode $00 = no
SEDT1	$00CC	204	Editor temp use
TBLX	$00CD	205	Current cursor physical line number
DATAX	$00CE	206	Temp data area
INSRT	$00CF	207	Flag: insert mode, >0 = # INSTs
	$00D0-00D7	208-215	Area for use by speech software
	$00D8-00E8	216-232	Area for use by application software
CIRSEG	$00E9	233	Screen line link table / editor temps
USER	$00EA-00EB	234-235	Screen editor color IP
KEYTAB	$00EC-00ED	236-237	Key scan table indirect
TMPKEY	$00EE	238	
NDX	$00EF	239	Index to keyboard queue
STPFLG	$00F0	240	Pause flag
TO	$00F1-00F2	241-242	Monitor ZP storage
CHRPTR	$00F3	243	
BUFEND	$00F4	244	
CHKSUM	$00F5	245	Temp for checksum calculation
LENGTH	$00F6	246	
PASS	$00F7	247	Which pass we are doing str
TYPE	$00F8	248	Type of block
USEKDY	$00F9	249	(B.7=1)=> for wr,(B.6=1)=> for rd
XSTOP	$00FA	250	Save xres for quick stopkey test
CURBNK	$00FB	251	Current bank configuration
XON	$00FC	252	Char to send for a x-on
XOFF	$00FD	253	Char to send for a x-off
SEDT2	$00FE	254	Editor temporary use
LOFBUF	$00FF	255	
FBUFFR	$0100-$010F	256-271	

SAVEA	$0110	272	Temp Locations for
SAVEY	$0111	273	... for Save and
SAVEX	$0112	274	...Restore
COLKEY	$0113-0122	275-289	Color/luminance table in RAM
SYSSTK	$0124-01FF	291-511	System stack
BUF	$0200-0258	512-600	Basic/monitor buffer
OLDLIN	$0259-025A	601-602	Basic storage
OLDTXT	$025B-025C	603-604	Basic storage
	$025D-02AC	605-684	BASIC/DOS INTERFACE AREA
XCNT	$025D		DOS loop counter
FNBUFR	$025E-026D		Area for filename
DOSF1L	$026E		DOS filename 1 length
DOSDS1	$026F		DOS disk drive 1
DOSF1A	$0270-0271		DOS filename 1 addr
DOSF2L	$0272		DOS filename 2 length
DOSDS2	$0273		DOS disk drive 2
DOSF2A	$0274-0275		DOOS filename 2 addr
DOSLA	$0276		DOS logical addrress
DOSFA	$0277		DOS phys addr
DOSSA	$0278		DOS secondary address
DOSDID	$0279-027A		DOS disk identifier
DIDCHK	$0278		DOS DID flag
DOSSTR	$027C		DOS output string buffer
DOSSPC	$027D-02AC		Area used to build DOS string
			GRAPHIC VARIABLES
XPOS	$02AD-02AE	685-686	Current x position
YPOS	$02AF-02B0	687-688	Current y position
XDEST	$02B1-02B2	689-690	X coordinate destination
YDEST	$02B3-02B4	691-692	Y coordinate destination
XABS	$02B5-02B6	693-694	
YABS	$02B7-02B8	695-696	
XSGN	$02B9-02BA	697-698	
YSGN	$02BB-02BC	699-700	
FCT1	$02BD-02BE	701-702	
FCT2	$02BF-02C0	703-704	
ERRVAL	$02C1-02C2	705-706	
LESSER	$02C3	707	
GREATR	$02C4	708	
ANGSGN	$02C5	709	Sign of angle
SINVAL	02C6-02C7	710-711	Sine of value of angle

COSVAL	$02C8-02C9	712-713	Cosine of value of angle
ANGCNT	$02CA-02CB	714-715	Temps for angle distance routines
			Start of multiply defined area #1
	$02CC	716	Placeholder
BNR	$02CD	717	Pointer to begin no.
ENR	$02CE	718	Pointer to end no.
DOLR	$02CF	719	Dollar flag
FLAG	$02D0	720	Comma flag
SWE	$02D1	721	Counter
USGN	$02D2	722	Sign exponent
UEXP	$02D3	723	Pointer to exponent
VN	$02D4	724	# of digits before decimal point
CHSN	$02D5	725	Justify flag
VF	$02D6	726	# of pos before decimal point (field)
NF	$02D7	727	# of pos after decimal point (field)
POSP	$02D8	728	+/− flag (field)
FESP	$02D9	729	Exponent flag (field)
ETOF	$02DA	730	Switch
CFORM	$02DB	731	Char counter (field)
SNO	$02DC	732	Sign no.
BLFD	$02DD	733	Blank/star flag
BEGFD	$02DE	734	Pointer to beginning of field
LFOR	$02DF	735	Length of format
ENDFD	$02E0	736	Pointer to end of field
XCENTR	$02CC-02CD	716-717	
YCENTR	$02CE-02DF	718-719	
XDIST1	$02D0-02D1	720-721	
YDIST1	$02D2-02D3	722-723	
XDIST2	$02D4-02D5	724-725	
YDIST2	$02D6-02D7	726-727	
	$02D8-02D9	728-729	Placeholder
COLCNT	$02DA	730	Characters column counter
ROWCNT	$02DB	731	Characters row counter
STRCNT	$02DC	732	
			START OF MULTIPLY DEFINED AREA #2
XCORD1	$02CC-02CD	716-717	
YCORD1	$02CE-02CF	718-719	
BOXANG	$02D0-02D1	720-721	Rotation angle
XCOUNT	$02D2-02D3	722-723	
YCOUNT	$02D4-02D5	724-725	
BXLENG	$02D6-02D7	726-727	Length of a side
XCORD2	$02D8-02D9	728-729	
YCORD2	$02DA-02DB	730-731	

XCIRCL	$02CC-02CD	716-717	Circle center, x coordinate
YCIRCL	$02CE-02CF	718-719	Circle center, y coordinate
XRADUS	$02D0-02D1	720-721	X radius
YRADUS	$02D2-02D3	722-723	Y radius
ROTANG	$02D4-02D5	724-725	Rotation angle
ANGBEG	$02D8-02D9	728-729	Arc angle start
ANGEND	$02DA-02DB	730-731	Arc angle end
XRCOS	$02DC-02DD	732-733	X radius * cos (rotation angle)
YRSIN	$02DE-02DF	734-735	Y radius * sin (rotation angle)
XRSIN	$02E0-02E1	736-737	X radius * sin (rotation angle)
YRCOS	$02E2-02E3	738-739	Y radius * cos (rotation angle)
			Start of multiply defined area #3
	$02CC	716	Placeholder
KEYLEN	$02CD	717	
KEYNXT	$02CE	718	
STRSZ	$02CF	719	String length
GETTYP	$02D0	720	Replace string mode
STRPTR	$02D1	721	String position counter
OLDBYT	$02D2	722	Old bit map byte
NEWBYT	02D3	723	New string or bit map byte
	$02D4	724	Placeholder
XSIZE	$02D5-02D6	725-726	Shape column length
YSIZE	$02D7-02D8	727-728	Shape row length
XSAVE	$02D9-02DA	729-730	Temp for column length
STRADR	$02DB-02DC	731-732	Save shape string descriptor
BITIDX	$02DD	733	Bit incex into byte
SAVSIZ	$02DE-02E1	734-737	Temporary work locations
CHRPAG	$02E4	740	High byte addr of char ROM for char CMD
BITCNT	$02E5	741	Temp for gshape
SCALEM	$02E6	742	Scale mode flag
WIDTH	$02E7	743	Double width flag
FILFLG	$02E8	744	Box fill flag
BITMSK	$02E9	745	Temp for bit mask
NUMCNT	$02EA	746	
TRCFLG	$02EB	747	Flags trace mode
T3	$02EC	748	
T4	$02ED-02EE	749-750	
VTEMP3	$02EF	751	Graphic temp storage
VTEMP4	$02F0	752	
VTEMP5	$02F1	753	
ADRAY1	$02F2-02F3	754-755	Ptr to routine: convert float to integer

ADRAY2	$02F4-02F5	756-757	Ptr to routine: convert integer tofloat
BNKVEC	$02FE-02FF	766-767	Vector for function cartridge users
IERROR	$0300-0301	768-769	Indirect Error (Output Error in .X)
IMAIN	$0302-0303	770-771	Indirect Main (System Direct Loop)
ICRNCH	$0304-0305	772-773	Indirect Crunch (Tokenization Routine)
IQPLOP	$0306-0307	774-775	Indirect List (Char List)
IGONE	$0308-0309	776-777	Indirect Gone (Character Dispatch)
IEVAL	$030A-030B	778-779	Indirect Eval (Symbol Evaluation)
IESCLK	$030C-030D	780-781	Escape token crunch
IESCPR	$030E-030F	782-783	
IESCEX	$0310-0311	784-785	
ITIME	$0312-0313	786-787	
CINV	$0314-0315	788-789	IRQ Ram Vector
CBINV	$0316-0317	790-791	Brk Instr RAM Vector
IOPEN	$0318-0319	792-793	Indirects for Code
ICLOSE	$031A-031B	794-795	
ICHKIN	$031C-031D	796-797	
ICKOUT	$031E-031F	798-799	
ICLRCH	$0320-0321	800-801	
IBASIN	$0322-0323	802-803	
IBSOUT	$0324-0325	804-805	
ISTOP	$0326-0327	806-807	
IGETIN	$0328-0329	808-809	
ICLALL	$032A-032B	810-811	
USRCMD	$032C-032D	812-813	
ILOAD	$032E-032F	814-815	
ISAVE	$0330-0331	816-817	Savesp
TAPBUF	$0333-03F2	819-1010	Cassette tape buffer
WRLEN	$03F3-03F4	1011-1012	Length of data to be written to tape
RDCNT	$03F5-03F6	1013-1014	Length of data to be read from tape
INPQUE	$03F7-0436	1015-1078	RS-232 input queue
ESTAKL	$0437-0454	1079-1108	
ESTAKH	$0455-0472	1109-1138	
CHRGET	$0473-0478	1139-1144	
CHRGOT	$0479-0484	1145-1156	
QNUM	$0485-0493	1157-1171	
INDSUB	$0494-04A1	1172-1185	Shared ROM fetch sub
ZERO	$04A2-04A4	1186-1188	Numeric constant for Basic
INDTXT	$04A5-04AF	1189-1199	Txtptr

INDIN1	$04B004BA	1200-1210	Index & Index1
INDIN2	$04BB-04C5	1211-1221	Index2
INDST1	$04C6-04D0	1222-1232	Strng1
INDLOW	$04D1-04DB	1233-1243	Lourtr
INDFMO	$04DC-04E6	1244-1254	Facmo
PUFILL	$04E7	1255	Print using fill symbol
PUCOMA	$04E8	1256	Print using comma symbol
PUDOT	$04E9	1257	Print using D.P. symbol
PUMONY	$04EA	1258	Print using monetary symbol
TMPOES	$04EB-04EE	1259-1262	Temp for instr
ERRNUM	$04EF	1263	Last error number
ERRLIN	$04F0-04F1	1264-1265	Line #of last error
TRAPNO	$04F2-04F3	1266-1267	Line to go on error
TMPTRP	$04F4	1268	Hold trap no. temporarily
ERRTXT	$04F5-04F6	1269-1270	
OLDSTK	$04F7	1271	
TMPTXT	$04F8-04F9	1272-1273	
TMPLIN	$04FA-04FB	1274-1275	
MTIMLO	$04FC-04FD	1276-1277	Table of pending jiffies (2's comp)
MTIMHI	$04FE-04FF	1278-1279	
USRPOK	$0500-0502	1280-1282	
RNDX	$0503-0507	1283-1287	
DEJAVU	$0508	1288	'cold' or 'warm' start status
LAT	$0509-0512	1289-1298	Logical file numbers
FAT	$0513-051C	1299-1308	Primary device numbers
SAT	$0510-0526	1309-1318	Secondary addresses
KEYD	$0527-0530	1319-1328	IRQ keyboard buffer
MEMSTR	$0531-0532	1329-1330	Start of memory
MSIZ	$0533-0534	1331-1332	Top of memory
TIMOUT	$0535	1333	IEEE timeout flag
FILEND	$0536	1334	File end reached =1, 0 othervise
CTALLY	$0537	1335	#of chars left in buffer (for R&W)
CBUFVA	$0538	1336	#of total valid chars in buffer (R)
TPTR	$0539	1337	Ptr to next char in buffer (for R&W)
FLTYPE	$053A	1338	Contains type of current cass file
COLOR	$053B	1339	Active attribute byte
FLASH	$053C	1340	Character flash flag
	$053D	1341	FREE

HIBASE	$053E	1342	Base location of screen (top)
XMAX	$053F	1343	
RPTFLG	$0540	1344	Key repeat flag
KOUNT	$0541	1345	
DELAY	$0542	1346	
SHFLAG	$0543	1347	Shif flag byte
LSTSHF	$0544	1348	Last shift pattern
KEYLOG	$0545-0546	1349-1350	Indirect for keyboard table setup
MODE	$0547	1351	
AUTODN	$0548	1352	Auto scroll down flag (O=on,O<yof)
LINTMP	$0549	1353	
ROLFLG	$054A	1354	
FORMAT	$054B	1355	Monitor non-zpage storage
MSAL	$054C-054E	1356-1358	
WRAP	$054F	1359	
TMPC	$0550	1360	
DIFF	$0551	1361	
PCH	$0552	1362	
PCL	$0553	1363	
FLGS	$0554	1364	
ACC	$0555	1365	
XR	$0556	1366	
YR	$0557	1367	
SP	$0558	1368	
INVL	$0559	1369	
INVH	$055A	1370	
CMPFLG	$055B	1371	Used by various monitor routines
BAD	$055C	1372	
KYNDX	$0550	1373	Used for programmable keys
KEYIDX	$055E	1374	
KEYBUF	$055F-0566	1375-1382	Table of P.F. lengths
PKYBUF	$0567-05E6	1383-1510	P.F. key storage area
KDATA	$05E7	1511	Temp for data write to kennedy
KDYCMD	$05E8	1512	Select for kennedy read or write
KDYNUM	$05E9	1513	Kennedy's dev#
KDYPRS	$05EA	1514	Kennedy present = $ff,else=$00
KDYTYP	$05EB	1515	Temp for type of open for kennedy
SAVRAM	$05EC-06EB	1516-1771	1 rage used by banking routines
PAT	$05EC-05EF	1516-1519	Physical Address Table
LNGJMP	$05F0-05F1	1520-1521	Long jump address
FETARG	$05F2	1522	Long jump accumulator
FETXRG	$05F3	1523	Long jump x register
FETSRG	$05F4	1524	Long jump status register

AREAS	$05F5-065D	1525-1629	RAM areas for banking
ASPECH	$065E-06EB	1630-1771	RAM area for speech
STKTOP	$06EC-07AF	1772-1967	BASIC run-time stack
WROUT	$07B0	1968	Byte to be written on tape
PARITY	$07B1	1969	Temp for parity calc
TT1	$07B2	1970	Temp for write-header
TT2	$07B3	1971	Temp for write-header
RDBITS	$07B5	1973	Local index for READBYTE routine
ERRSP	$07B6	1974	Pointer into the error stack
FPERRS	$07B7	1975	Number of first pass errors
DSAMP1	$07B8-07B9	1976-1977	Time constant
DSAMP2	$07BA-07BB	1978-1979	Time cinstant
ZCELL	$07BC-07BD	1980-1981	Time constant
SRECOV	$07BE	1982	Stack marker for stopkey recover
DRECOV	$07BF	1983	Stack marker for dropkey recover
TRSAVE	$07C0-07C3	1984-1987	Params passed to RDBLOK
RDETMP	$07C4	1988	Temp stat save for RDBLOK
LDRSCN	$07C5	1989	# consec shorts to find in leader
CDERRM	$07C6	1990	# Errors fatal in RD countdown
VSAVE	$07C7	1991	Temp for Verify command
T1PIPE	$07C8-07CB	1992-1995	Pipe temp for T1
ENEXT	$07CC	1996	Read error propagate
			FOR RS-323
UOUTQ	$07CD	1997	User character to send
UOUTFG	$07CE	1998	0=empty ; 1=full
SOUTQ	$07CF	1999	System character to send
SOUNFG	$07D0	2000	0=empty ; 1=full
INQFPT	$07D1	2001	Pntr to front of input queue
INQRPT	$07D2	2002	Pntr to rear of input queue
INCNT	$07D3	2003	# of chars in input queue
ASTAT	$07D4	2004	Temp status for ACIA
AINTMP	$07D5	2005	Temp for input routine
ALSTOP	$07D6	2006	FLG for local pause
ARSTOP	$07D7	2007	FLG for remote pause
APRES	$07D8	2008	FLG to indicate presence of ACIA
KLUDES	$07D9-07E4	2009-2020	Indirect routine downloaded
SCBOT	$07E5	2021	Screen bottom line in window
SCTOP	$07E6	2022	Screen top line in window

SCLE	$07E7	2023	Screen left row in window
SCRT	$07E8	2024	Screen right row in window
SCRDIS	$07E9	2025	
INSFLG	$07EA	2026	
LSTCHR	$07EB	2027	
LOGSCR	$07EC	2028	
TCOLOR	$07ED	2029	
BITABL	$07EE-07F1	2030-2033	
SAREG	$07F2	2023	Registers for SYS command
SXREG	$07F3	2035	
SYREG	$07F4	2036	
SPREG	$07F5	2037	
LSTX	$07F6	2038	Key scan index
STPDSB	$07F7	2039	Flag to disable CTL-S pause
RAMROM	$07F8	2040	MSB for monitor fetches from ROM=0;RAM=1
COLSW	$07F9	2041	MSB for colot/lim table in RAM=0;ROM=1
FERMSK	$07FA	2042	ROM mask for split screen
VMBMSK	$07FB	2043	VM base mask for split screen
LSEM	$07FC	2044	Motor lock semaphore for cassette
PALCNT	$07FD	2045	PAL tod
TEDATR	$0800-0BFF	2048-3071	TED attribute bytes
TEDSCN	$0C00-0FFF	3072-4095	TED character pointers
BASBGN	$10000-	4096-	Start of BASIC text area
BMLUM	$1800-$1BFF	6144-7167	Luminance for bit map screen
BMCOLR	$1C00-1FFF	7168-8191	Color for bit map
GRBASE	$2000-	8192-	Start of bit map screen data
GRBASE	$4000-	16384-	Start of BASIC when HIRES is on (in 64K machine only)
CHRBAS	$D000	53248	Beginning of character ROM data Text Display (TED) Chip
	$FF00	65280	Timer #1 reload value, bits0-7 (low)
	$FF01	65281	Timer #1 reload value, bits 8-15 (high)

$FF02	65282	Timer #2 reload value, bits 0-7 (low)
$FF03	65283	Timer #2 reload value, bits 8-15 (high)
$FF04	65284	Timer #3 reload value, bits 0-7 (low)
$FF05	65285	Timer #3 reload value, bits 8-15 (high)
$FF06	65286	Bits 0-2: screen vertical offset Bit 3: 24 or 25 rows Bit 4: Screen blank Bit 6: Extended color mode Bit 5: Bit map mode Bit 7: TEST
$FF07	65287	Bits0-2: horizontal offset Bit 3: 38 or 40 columns Bit 4: Multicolor mode Bit 5: Freeze Bit 6: NTSC/PAL Bit 7: Hardware reverse
$FF08	65288	Keyboard latch
$FF09	65289	Interrupt (IRQ) flags Bit 0: unused Bit 1: Raster Bit 2: Light pen Bit 3: Timer #1 Bit 4: Timer #2 Bit 5: unused Bit 6: Timer #3 Bit 7: Interrupt flag
$FF0A	65290	Interrupt enable register Bit 0: Bit 8 raster interrupt set Bit 1: Raster Bit 2: Light pen Bit 3: Timer #1 Bit 4: Timer #2 Bit 5: unused Bit 6: Timer #3 Bit 7: unused

$FF0B	65291	Raster interrupt set, bits 0-7
$FF0C	65292	Hardware cursor position bits 8 & 9 (bit 0 is bit 8, bit 1 is bit 9)
$FF0D	65293	Hardware cursor position, bits 0-7
$FF0E	65294	Voice #1 frequency, bits 0-7
$FF0F	65295	Voice #2 frequency, bits 0-7
$FF10	65296	Voice #2 frequency, bits 8 & 9 (bit 0 is bit 8, bit 1 is bit 9)
$FF11	65297	Bits 0-3: Volume control Bit 4: Voice #1 select Bit 5: Voice #2 select Bit 6: Voice #2 noise select Bit 7: Sound reload
$FF12	65298	Bit 0-1: Voice #1 frequency, bits 8 & 9 Bit 2: TED data fetch ROM/RAM select Bits 0-5: Bit map base address
$FF13	65299	Bit 0: Clock status Bit 1: Set single clock Bits 2-7: Character data base address
$FF14	65300	Bits 3-7: Video matrix/color memory base address
$FF15	65301	Background color register Bits 0-3: Color Bits 4-6: Luminance
$FF16	65302	Color register #1 Bits 0-3: Color Bits 4-6: Luminance
$FF17	65303	Color register #2 Bits 0-3: Color Bits 4-6: Luminance
$FF18	65304	Color register #3 Bits 0-3: Color Bits 4-6: Luminance

$FF19	65305	Color register #4 Bits 0-3: Color Bits 4-6: Luminance
$FF1A	65306	Bit map reload
$FF1B	65307	Bit map reload
$FF1C	65308	Bit 0: Vertical line bit 8
$FF1D	65309	Bits 0:7: Vertical line bits 0-7
$FF1E	65310	Horizontal position
$FF1F	65311	Blink, vertical sub address
$FF3E	65342	ROM select
$FF3F	65343	RAM select

BANKING JUMP TABLE

$FCF1	64753	JMP to cartridge IRQ routine
$FCF4	64756	JMP to PHOENIX routine
$FCF7	64759	JMP to LONG FETCH routine
$FCFA	64762	JMP to LONG JUMP routine
$FCFD	64765	JMP to LONG IRQ routine
$FF49	65353	JMP to define function key routine
$FF4C	65356	JMP to PRINT routine
$FF4F	65359	JMP to PRIMM routine
$FF52	65362	JMP to ENTRY routine
$FF80	65408	RELEASE # OF KERNAL (msb 0=NTSC;1=PAL)

KERNAL TUMP TABLE

NAME	ADDRESS		DESCRIPTION
CINT	$FF81	65409	Initialize screen editor
IOINIT	$FF84	65412	Initialize I/O devices
RAMTAS	$FF87	65415	Ram test
RESTOR	$FF8A	65418	-Restore vectors to initial values
VECTOR	$FF8D	65421	Change vectors for user
SETMSG	$FF90	65424	Control O.S. messages
SECND	$FF93	65427	Send SA after LISTEN
TKSA	$FF96	65430	Send SA after TALK

MEMTOP	$FF99	65433	Set/Read top of memory
MEMBOT	$FF9C	65436	Set/Read bottom of memory
SCNKEY	$FF9F	65439	Scan keyboard
SETTMO	$FFA2	65442	Set timeout in DMA disk
ACPTR	$FFA5	65445	Handshake serial bus or DMA disk byte in
CIOUT	$FFA8	65448	Handshake serial bus or DMA disk byte out
UNTLK	$FFAB	65451	Send UNTALK out serial bus or DMA disk
UNLSN	$FFAE	65454	Send UNLISTEN out serial bus or DMA disk
LISTN	$FFB1	65457	Send LISTEN out serial bus or DMA disk
TALK	$FFB4	65460	Send TALK out serial bus or DMA disk
READSS	$FFB7	65463	Return I/O STATUS byte
SETLFS	$FFBA	65466	Set LA, FA, SA
SETNAM	$FFBD	65469	Set length and FN address
OPEN	$FFC0	65472	Opel logical file
CLOSE	$FFC3	65475	Close logical file
CHKIN	$FFC6	65478	Open channel in
CHOUT	$FFC9	65481	Open channel out
CLRCH	$FFCC	65484	Close I/O channels
BASIN	$FFCF	65487	Input from channel
BSOUT	$FFD2	65490	Output to channel
LOADSP	$ FFD5	65493	Load from file
SAVESP	$ FFD8	65496	Save to file
SETTIM	$FFDB	65499	Set internal clock
RDTIM	$FFDE	65502	Read internal clock
STOP	$FFE1	65505	Scan STOP key
GETIN	$FFE4	65508	Get character from queue
CLALL	$FFE7	65511	Close all files
UDTIM	$FFEA	65514	Increment clock
SCRORG	$FFED	65517	Screen org.
PLOT	$FFF0	65520	Read/Set X,Y coord of cursor
IOBASE	$FFF3	65523	Return location of start of I/O

Index