COMMODORE 128

SYSTEMVEJLEDNING

Commodore

Commodore BASIC 7.0

CP/M® Plus Version 3.0

Datastyret fotosats: stanley grafik aps . risskov

KAPITEL

1

INTRODUKTION

AFSNIT 1
Brugen af denne vejledning

INDHOLDSFORTEGNELSE
COMMODORE 128
BRUGERVEJLEDNING

KAPITEL 5 - BASIC 7.0 LEKSIKON

APPENDIKS

APPENDIKS K - BASIC 7.0 FORKORTELSER
APPENDIKS L - DISKETTEKOMMANDO OVERSIGT

Denne Commodore 128 System Vejledning er skrevet for at hjælpe Dem til fuld forståelse af de avancerede muligheder, De har med Commodore 128 Computeren. Vejledningen bruges på følgende måde:

Læs ikke længere i denne vejledning, før De har gennemlæst "Introductory Guide", som indeholder vigtige oplysninger om, hvordan man kommer igang med Commodore 128.

Hvis De fortrinsvis er interesseret i brug af BASIC sproget for at udvikle og køre Deres egne programmer, bør De først læse afsnit 2 i denne introduktion. I dette afsnit forklares de tre operations modes, som Commodore 128 tilbyder. Læs derefter kapitel 2, BRUG AF BASIC PÅ COMMODORE 128. Kapitel 2 fortæller Dem, hvorledes BASIC programmeringssproget bruges både i C128 og C64 mode; forklarer Commodore 128 tastaturet; beskriver visse avancerede kommandoer, De kan bruge i både C128 og C64 mode; viser hvordan en lang række nye kraftfulde BASIC kommandoer skal bruges - herunder også farve, grafik og lydkommandoer, som er unikke i C128 mode; og beskriver hvordan der er mulighed for at arbejde med 80 kolonner i C128 mode.

Hvis De ønsker at anvende BASIC i C64 mode, bør De gennemlæse kapitel 3, C64 MODE. Læg imidlertid mærke til, at Commodore 128 BASIC 7.0 indeholder væsentlig flere BASIC kommandoer end BASIC 2.0 i C64 mode. I de fleste tilfælde er disse C128 BASIC kommandoer mere kraftfulde og lettere at bruge end kommandoer i C64 mode. **Husk, at De kan benytte C64 mode for at køre de mange tusinde C64 programmer, der findes på markedet.**

Ønsker De at bruge CP/M på Commodore 128, læs da kapitel 4, CP/M MODE. Dette kapitel fortæller, hvorledes De starter og bruger CP/M på Commodore 128. I CP/M mode kan De vælge mellem flere tusinde færdige kommercielle programpakker. De kan også udvikle Deres egne CP/M programmer.

Hvis De ønsker detaljer vedrørende BASIC 7.0 kommandoer, læs da Kapitel 5, BASIC 7.0 ordforklaring. Dette kapitel fortæller om formater og detaljer om alle BASIC 7.0 kommandoer, instruktioner og funktioner.

Hvis De, efter at have gennemlæst kapitlerne 1 til og med 5, ønsker yderligere tekniske oplysninger om et specielt Commodore 128 emne, prøv da først at se i denne vejlednings **APPENDIKS.**

Dette appendiks indeholder mange forskellige oplysninger og omfatter blandt andet: en komplet liste over BASIC og DOS fejlmeddelelser, og en liste over diskettekommandoer.

COMMODORE 128 PROGRAMMER'S GUIDE indeholder komplette tekniske detaljer om enhver mulighed med Commodore 128. (Til rådighed primo 1986).

AFSNIT 2
GENNEMGANG AF COMMODORE 128 PERSONAL COMPUTER

Commodore 128 tilbyder mange kraftfulde nye muligheder, bl. a.:

* Et meget udvidet BASIC sprog - Commodore BASIC 7.0 - som giver nye kraftige kommandoer og muligheder

* 128K RAM, som kan udvides til 256 eller 512K ved brug af RAM udvidelsesmoduler

* 40 og 80 kolonners uddata

* Nye hurtige 1570 og 1571 diskettestationer

* 2 MHz operation

* CP/M 3.0 operation

* Et professionelt tastatur med separat titalstastatur

* Indbygget maskinsprogs monitor

Commodore 128 Personal Computer er i virkeligheden tre computere i een, og tilbyder tre primære operationstilstande (modes):

* **C128 Mode**
* **C64 Mode**
* **CP/M Mode**

Her er en kort gennemgang af hver mode:

C128 Mode

I C128 mode giver Commodore 128 Personal Computer adgang til 128K RAM og en kraftig, udvidet BASIC, kendt som BASIC 7.0 - som tilbyder mere end 140 kommandoer, instruktioner og funktioner - BASIC 7.0 er skabt af Commodore for at give bedre og lettere måder til at udføre avancerede programmeringsopgaver på, incl. de, som omfatter grafik, animation, lyd og musik. C128 mode har også mulighed for uddata i både 40 og 80 kolonners format og anvendelse af fuldt tastatur med 92 taster. Tastaturet omfatter et titalstastatur samt Escape-, Tab-, Skifte- og Hjælpetaster. Den indbyggede maskinsprogs monitor sætter Dem i stand til at fremstille og fejlfinde Deres egne maskinsprogs programmer. Sådanne programmer kan bruges i forbindelse med BASIC programmer. I C128 mode kan De anvende et antal nye perifere enheder, bl. a. en ny hurtig seriel diskettestation, en mus og et standard 40/80 kolonners 'composite' video/RGBI monitor. Og De kan bruge alle Commodore standard serielle enheder.

C64 mode

Når den er i C64 mode, virker Commodore 128 nøjagtig som en Commodore 64 computer. Commodore 128 bevarer alle de muligheder den kommercielt succesrige C64 har, og tillader Dem derved at have fuld anvendelsesmulighed af det omfattende programudbud, der er på markedet til denne.
Der er også fuld overensstemmelse med C64 perifere enheder, incl. kassettestation, joystick, brugerport og serielle enheder, samt C64 'composite' video monitor og TV-apparater.

C64 mode giver BASIC 2.0 sproget, 40 kolonners uddata og 64K RAM. Tastaturets virkemåde, med undtagelse af funktionstasternes placering, er som på en Commodore 64 computer. Alle C64 grafik, farve- og lydmuligheder er bevaret og bruges nøjagtigt som på en Commodore 64.

CP/M Mode

I CP/M mode giver en indbygget Z80 mikroprocessor Dem mulighed for at anvende alle faciliteter i Digital Research's CP/M Plus version 3.0, plus en del nye muligheder, som Commodore har udvidet med. Commodore 128's CP/M pakke, kaldet CP/M Plus, giver 128K RAM; 40 eller 80 kolonners uddata; brug af fuldt tastatur; og anvendelsesmulighed af den nye Commodore 1571 hurtige serielle diskettestation, såvel som standard serielle enheder.

Kapitlerne 2, 3 og 5, som indeholder afsnittene 3 til 15, fortæller Dem om, hvorledes De har adgang til og benytter mulighederne i de tre kraftfulde og enestående operationsmodes i Commodore 128 Personal Computer.

HVORLEDES DE TÆNDER DERES COMMODORE 128

Før De tænder for Deres Commodore 128, er der nogle få ting, De skal kontrollere for at komme godt igang. En ting, De skal gøre, før De sætter strøm til computeren, er at sikre Dem, at 40/80 tasten i tastaturets øverste række er indstillet, så den passer til Deres monitor. Hvis De for eksempel har en 40 kolonners monitor, skal 40/80 tasten være i øverste position. Har De en 80 kolonners monitor, skal tasten være nedtrykket.

Hvis De benytter en Commodore 1901 dobbeltfunktions monitor i 40 kolonners format, skal 40/80 tasten være oppe og 40/80 kontakten på monitoren i midterpositionen. Bruges 80 kolonners formatet skal 40/80 tasten være nedtrykket og kontakten på monitorens forside i yderste venstre position.

Uanset hvilket skærmformat De anvender, bør De kontrollere, at tasterne CAPS LOCK og SHIFT LOCK er i øverste position. Er dette ikke tilfældet, får De måske slet intet billede, eller skærmen fyldes af besynderlige tegn.
(Se afsnit 5 for beskrivelse af de specialtaster, der bruges i C128 mode)

BRUG AF SOFTWARE

Hvis De bruger et MAGIC VOICE talemodul, indsættes modulet i udvidelsesstikket. Så holdes Commodoretasten nedtrykket, mens der tændes for strømmen. **Sæt aldrig en programkapsel i, når der er tændt for strømmen.**

Har De problemer med at starte en programkapsel i C64 mode, sæt så programkapslen i, mens der er slukket for strømmen og hold så Commodoretasten (C=) nedtrykket, mens De tænder for computeren.

Hvis De har en CP/M 2.2 kapsel beregnet for Commodore 64, må De aldrig sætte denne i Deres Commodore 128. Commodore 128 er forsynet med en Z80 mikroprocessor, som kun kan bruges med CP/M 3.0. Isætter De en CP/M 2.2 kapsel, kan det give uforudsigelige resultater.

Bruger De software, som omfatter en lyspen, skal lyspennen sluttes til kontrolport 1, som findes på højre side af C128 nær strømkontakten.

KAPITEL

2

BRUGEN AF C128 MODE

SKIFT MELLEM FORSKELLIGE MODES

UDGANGSPUNKT

TIL	SLUKKET	C128 40 KOL.	C128 80 KOL.	C64	CP/M 40 KOL.	CP/M 80 KOL.
C128 40	1. 40/80 skal være oppe. 2. Tænd computer.		1. Tryk på ESC-tasten og udløs den igen. 2. Tryk på X-tasten ELLER: 1. 40/80 skal være oppe. 2. Tryk på RESET-tast.	1. 40/80 skal være oppe. 2. Sluk computer og tænd igen.	1. 40/80 skal være oppe. 2. Sluk computer og tænd igen.	1. 40/80 skal være oppe. 2. Sluk computer og tænd igen.
C128 80	1. 40/80 skal være nede. 2. Tænd computer.	1. Tryk på ESC-tasten og udløs den igen. 2. Tryk på X-tasten ELLER: 1. Nedtryk 40/80-tast 2. Nedtryk RESET-tast.		1. 40/80 skal være nede. 2. Sluk computer og tænd igen.	1. 40/80 skal være nede. 2. Fjern CP/M disk fra drevet. 3. Sluk computer og tænd igen.	1. 40/80 skal være nede. 2. Fjern CP/M disk fra drevet. 3. Sluk computer og tænd igen.
C64	1. Hold C= tast nede. 2. Tænd computer. ELLER: 1. Isæt prg.kapsel 2. Tænd computer.	1. Indtast GO64 ⟨R⟩ 2. Tast Y ⟨R⟩ ved: ARE YOU SURE?	1. Indtast GO64 ⟨R⟩ 2. Tast Y ⟨R⟩ ved: ARE YOU SURE?		1. Sluk for computer. 2. 40/80 op 3. HOLD C= tast nede og tænd computer. ELLER: 1. Sluk for computer. 2. Isæt prg.kapsel 3. Tænd computer.	1. Sluk for computer. 2. 40/80 op 3. Hold C= tast nede og tænd computer. ELLER: 1. Sluk for computer. 2. Isæt prg.kapsel. 3. Tænd computer.
CP/M 40	1. Tænd disk.drev. 2. Isæt CP/M syst. diskette. 3. 40/80 tast oppe. 4. Tænd computer.	1. Tænd disk.drev. 2. Isæt CP/M syst. diskette. 3. 40/80 tast oppe. 4. Tast: BOOT ⟨R⟩	1. Tænd disk.drev. 2. Isæt CP/M syst. diskette. 3. 40/80 tast oppe. 4. Tast: BOOT ⟨R⟩.	1. 40/80 skal være oppe. 2. Tænd disk.drev. 3. Isæt CP/M syst. diskette. 4. Sluk for computer og tænd igen		1. Isæt CP/M utility disk. 2. Ved A⟩ sys.melding indtastes: DEVICE CONOUT: = 40 COL ⟨RETURN⟩.
CP/M 80	1. Tænd disk.drev. 2. Isæt CP/M syst. diskette. 3. Nedtryk 40/80-tast. 4. Tænd computer.	1. Tænd disk.drev. 2. Isæt CP/M syst. diskette. 3. Nedtryk 40/80-tast 4. Tast: BOOT ⟨R⟩.	1. Tænd disk.drev. 2. Isæt CP/M syst. diskette. 3. Nedtryk 40/80-tast 4. Tast: BOOT ⟨R⟩.	1. 40/80 skal være nede. 2. Tænd disk.drev 3. Isæt CP/M sys. diskette. 4. Sluk computer og tænd den igen.	1. Isæt CP/M utility disk. 2. Ved A⟩ sys.melding indtastes: DEVICE CONOUT: = 80 COL ⟨RETURN⟩	

OBS! Hvis De benytter en Commodore 1901 dobbeltfunktions monitor, skal De huske at flytte kontakten fra PAL til RGBI, når De skifter fra 40-kolonners til 80-kolonners format og vice versa. Husk også, at eventuelle programkapsler skal fjernes fra udvidelsesstikket, og at disketter skal tages ud af diskettedrevet, når der skiftes mellem modes.

AFSNIT 3
HVORDAN MAN STARTER MED BASIC

Programmeringssproget BASIC er et specielt sprog, som tillader Dem at kommunikere med Deres Commodore 128. Anvendelse af BASIC er een måde, på hvilken De kan instruere Deres computer om, hvad den skal gøre.

BASIC har sit eget ordforråd (kaldet kommandoer, instruktioner og funktioner) og sine egne regler for strukturering (kaldet syntaks). Ved hjælp af BASIC ord og syntaks kan De sammensætte et instruktionssæt kaldet et program, som Deres computer kan udføre eller "RUN"ne.

Med BASIC kan De på to måder kommunikere med Deres Commodore 128: i DIREKTE mode eller i PROGRAM mode.

DIREKTE MODE

Såsnart De har tændt for Deres Commodore 128, er den klar til at modtage BASIC kommandoer i DIREKTE mode. Kommandoerne indtastes på tastaturet og afsendes til computeren ved at trykke på RETURN tasten. Computeren udfører alle kommandoer i DIREKTE mode straks efter, at De har trykket på RETURN. De fleste kommandoer i Deres Commodore 128 kan bruges såvel i direkte mode som i et program.

PROGRAM MODE

I PROGRAM mode indtastes et instruktionssæt (også kendt som et program), som udfører en afgrænset opgave. Hver instruktion er indeholdt i en sekventiel PROGRAM-linie, som kan være op til 160 karakterer i længden. Da en skærmlinie i 40 kolonners format er 40 tegn lang, svarer dette til fire SKÆRM linier og to linier i 80 kolonners format.

Når De har indtastet et program, kan De umiddelbart 'køre' det ved at indtaste RUN kommandoen og derefter nedtrykke RETURN tasten. Ved brug af kommandoerne DSAVE (eller SAVE) kan De også gemme programmet på diskette eller kassettebånd. De kan så senere genindlæse det fra diskette eller bånd ved brug af DLOAD (eller LOAD) kommandoen. Denne kommando kopierer programmet fra diskette eller bånd, og placerer programmets i computerens hukommelse. De kan derefter bruge eller udføre programmet ved at indtaste RUN kommandoen. Alle

disse kommandoer forklares senere i dette afsnit. Det meste af tiden vil De bruge Deres computer med programmer, købte færdige programmer, eller programmer, De selv har skrevet. De vil normalt kun operere i direkte mode, når De manipulerer med eller redigerer Deres programmer med kommandoer som LIST, LOAD, SAVE og RUN. Som en hovedregel er forskellen mellem direkte mode og programkørsel, at kommandoer afgivet i direkte mode ikke skal have linienumre.

BRUG AF TASTATURET

Herunder vises tastaturet på Commodore 128 Personal Computer.

Brugen af BASIC er i hovedtrækkene ens i både C64 og C128 mode. De fleste taster, og de fleste kommandoer, som De vil lære, kan bruges til BASIC programmering i begge modes. I C64 mode kan kun de skyggelagte taster benyttes. I C128 mode kan alle taster benyttes.

TASTATURETS TEGNSÆT

Tastaturet på Commodore 128 tilbyder to forskellige karaktersæt:

- Store bogstaver og grafiske karakterer
- Store og små bogstaver

I 80 kolonners format kan begge karaktersæt bruges samtidig. Dette giver en total på 512 forskellige karakterer, som De kan vise på skærmen. I 40 kolonners format kan der kun vises et tegnsæt ad gangen.

Når der tændes for Commodore 128 i 40 kolonners format, er tastaturet normalt i upper-case/grafisk mode. Dette betyder, at alt, hvad der indtastes, er med versalier (store bogstaver). For at skifte mellem de to modes, må De nedtrykke SHIFT og C= tasterne samtidigt. Tænd for Deres computer og prøv at øve Dem i brugen af de to karaktersæt. Nedtryk derefter SHIFT og Commodore tasterne (C=). Læg mærke til, hvordan skærmen skifter til uppercase/lower-case karakterer. Tryk igen på SHIFT og C= for at vende tilbage til upper-case og grafik.

BRUG AF KOMMANDO TASTERNE

KOMMANDO taster er taster, som sender meddelelser til computeren. Visse kommandotaster, som f. eks. RETURN, bruges selvstændigt. Andre kommandotaster, som f. eks. SHIFT, CTRL, C= og RESTORE, bruges sammen med andre taster. Anvendelsen af de enkelte taster forklares herunder.

RETURN

Når De nedtrykker RETURN tasten bliver, det De har skrevet, sendt til hukommelsen i Commodore 128. Nedtrykning af RETURN får også markøren (cursoren, det lille blinkende rektangel, som markerer hvor på skærmen, De befinder Dem) til at gå til næste linie.

De vil sikkert af og til komme til at stave en kommando forkert eller indtaste noget, som computeren ikke forstår. Hvis De så nedtrykker RETURN tasten, vil der sikkert fremkomme en meddelelse som SYNTAX ERROR på skærmen.

Dette kaldes en fejlmeddelelse. Appendiks A indeholder en fortegnelse over fejlmeddelelser og forklaring på, hvorledes fejlene rettes.

OBS! I eksemplerne i denne bog, angiver følgende symbol, når De skal trykke på Return-tasten:

RETURN

SHIFT

I tastaturets nederste række findes to skiftenøgler (SHIFT), een til venstre og een til højre, fuldstændig som på en almindelig skrivemaskine.

Skiftenøglerne kan bruges på tre måder:

1. Som skiftenøglen på en almindelig skrivemaskine, det vil sige, at når SHIFT holdes nedtrykket, skrives store bogstaver eller de tegn som findes øverst på dobbelttegns tasterne.

2. SHIFT kan bruges sammen med nogle af de andre kommandotaster for at udføre bestemte funktioner.

3. Når tastaturet er i upper-case/grafisk mode kan SHIFT bruges for at få skrevet de grafiske tegn eller symboler, som findes på forsiden af visse taster. Se afsnittet "Visning af grafiske karakterer" sidst i dette afsnit.

SHIFT LOCK

Når denne tast nedtrykkes, fastlåses den. Hvad De herefter skriver, vil være enten versalier eller de øverste tegn fra dobbelttegns taster. For at udløse låsningen, skal De blot nedtrykke tasten igen. Tastens funktion er identisk med SHIFT-tasten.

FLYTNING AF MARKØREN

I C128 mode kan De bevæge markøren ved brug enten af de fire piltaster, der er placeret i tastaturets øverste højre hjørne, eller ved hjælp af de to CRSR taster, yderst til højre i tastaturets nederste række.

BRUG AF DE FIRE PIL MARKØR TASTER

I C128 mode kan markøren flyttes i alle retninger ved simpelthen at bruge den piltast, der peger i den retning, hvorhen De ønsker, at markøren skal bevæge sig. (Disse taster kan ikke bruges i C64 mode)

BRUG AF CRSR TASTERNE

Disse taster kan bruges både i C64 og C128 mode.
Yderst til højre i tastaturets nederste række findes to taster, der kan bruges til at bevæge markøren:

- Nedtrykkes kun CRSR tasten, flyttes markøren **nedad**.
- Nedtrykkes CRSR og SHIFT samtidigt, flyttes markøren **opad**.
- Nedtrykkes kun CRSR tasten, flyttes markøren mod **højre**.
- Nedtrykkes CRSR og SHIFT samtidigt, flyttes markøren mod **venstre**.

De behøver ikke at nedtrykke markørtasten flere gange for at markøren til at flytte sig mere end een plads. De skal blot holde tasten nedtrykt, og markøren vil så flytte sig, til De slipper tasten.

Læg mærke til at hvis markøren når skærmens højre side, vil den gå videre ved begyndelsen af næste linie. Bevæges markøren mod venstre, vil den, når den når skærmens venstre kant, gå tilbage til den foregående linie.

Prøv at gøre Dem fortrolig med brugen af markørtasterne, fordi markørflytning kan gøre Deres programmering langt lettere. Med en smule øvelse vil De opdage, at De kan flytte markøren næsten uden at tænke over det.

INST/DEL

Dette er en tast med dobbeltfunktion. INST står for INSerT (indsæt) og DEL for DELete (slet).

Indsættelse af karakterer:

Hvis De ønsker at indsætte ekstra tegn i en linie, må De bruge SHIFT samtidig med INST/DEL tasten. Lad os forudsætte, at De har udeladt nogle tegn i en linie, som f. eks.:

MENS ■ VAR UDE

For at indsætte de manglende tegn, flyttes markøren først, ved anvendelse af CRSR tasterne, tilbage til fejlen, således:

MENS■VAR UDE

Derefter, mens De holder SHIFT nede, trykkes på INST/DEL, til der er tilstrækkelig plads for tilføjelse af de manglende karakterer, således:

MENS■ VAR UDE

Læg mærke til, at INST ikke flytter markøren; den laver blot plads mellem markøren og karakteren til højre. For at rette, taster De nu det manglende "H", "U" og "N", således:

MENS HUN VAR UDE

Sletning af karakterer:

Hvis De nedtrykker DEL tasten, bevæger markøren sig een plads mod venstre og sletter det tegn, som står der. Det betyder, at hvis De ønsker at slette et eller andet, kan De flytte markøren til pladsen lige til højre for den karakter, De vil slette. Forestil Dem, at De har lavet en fejlindtastning, som denne:

PRINT "FEJLREN"

De ville have skrevet FEJLEN, ikke FEJLREN. For at slette det fejlagtige R, sættes markøren der, hvor det sidste E står. Når De så trykker på DEL tasten, flytter tegnet under markøren en plads mod venstre.
De vil nu få det korrekt stavede ord:

PRINT "FEJLEN"

Samtidig brug af INSerT og DELete

For at rette forkerte tegn, kan De bruge INSerT og DELete funktionerne samtidigt. Flyt først markøren til den forkerte karakter og tryk derefter på INST/DEL tasten for at slette det forkerte

tegn. Nedtryk derefter samtidigt SHIFT og INST/DEL for at få plads til et nyt tegn. Indtast så det rigtige tegn. De kan også direkte indtaste oveni de uønskede tegn, og så bruge INST for at skabe den fornødne plads.

CONTROL

Kontroltasten (ConTRoL) bruges sammen med andre taster for at udføre specielle opgaver kaldet kontrolfunktioner. For at udføre en kontrolfunktion holdes CTRL tasten nedtrykket, mens De samtidig trykker på en anden tast.
Kontrolfunktioner anvendes ofte i færdigprogrammeret software, som f. eks. tekstbehandlingssystemer.

En ofte brugt kontrolfunktion er farvevalg. For at vælge en farve skal CTRL tasten holdes nedtrykket, mens De trykker på een af taltasterne (1 til 8, på tastaturets øverste række), som kontrollerer, hvilken farve De ønsker. Der findes yderligere otte farver, De kan vælge. Disse kan, som senere forklaret, vælges med C= tasten.

RUN/STOP

Dette er en tast med dobbeltfunktion. Under visse omstændigheder kan De bruge denne tast's RUN funktion ved at nedtrykke SHIFT og RUN/STOP samtidigt. Det er også muligt at bruge tastens STOP funktion, for at standse et program eller udskrivning, ved at nedtrykke tasten, mens programmet kører.
STOP tastens STOP funktion er dog ofte undertrykket (ude af funktion) i færdig købte programmer. Dette er gjort for at forhindre, at brugeren standser programmet under programkørslen, før det når sit normale slutpunkt. Var brugeren i stand til at standse programmet, kunne værdifulde oplysninger gå tabt.

RESTORE

RESTORE tasten bruges sammen med RUN/STOP for at tilbagestille computeren til den normale tilstand (også kaldet DEFAULT tilstanden).
De fleste færdige programmer undertrykker RESTORE tasten af samme årsag som STOP funktionen - altså for at undgå tab af værdifulde oplysninger.

CLR/HOME

CLR betyder CLeaR (slet). Home refererer til skærmens øverste venstre position, som også kaldes HOME positionen. Nedtrykkes HOME tasten, flyttes markøren til HOME positionen. Bruges SHIFT sammen med CLR/HOME tasten, slettes skærmindholdet og markøren går til HOME positionen.

COMMODORE TASTEN (C=)

C= tasten har flere funktioner, bl. a. følgende:

1. C= tasten sætter Dem i stand til at skifte mellem UPPER/LOWER-CASE DISPLAY MODE (der viser bogstaver og de karakterer, som står foroven på tasterne) og UPPER-CASE/GRAPHIC DISPLAY MODE (der viser store bogstaver og de grafiske symboler, som står på forsiden af tasterne). Skift mellem de to tilstande fås ved samtidig at nedtrykke C= tasten og SHIFT.
2. C= tasten giver også mulighed for at benytte et andet farvesæt på otte kulører. For at få disse farver holdes C= tasten nedtrykket, mens der trykkes på nummertasterne (fra 1 til 8) i øverste tastrække.
3. Hvis C= tasten holdes nedtrykkes, mens der tændes for computeren, vil der blive startet i C64 mode.

FUNKTIONSTASTER

De fire store taster, som findes umiddelbart til højre for det almindelige tastatur og lige ovenfor det numeriske tastatur, kaldes funktionstasterne.
Tasterne er foroven mærket F1, F3, F5 og F7 og på forsiden F2, F4, F6 og F8.
Disse taster kan programmeres, d.v.s. at de kan instrueres om at udføre bestemte opgaver eller funktioner. Derfor kaldes de PROGRAMMERBARE FUNKTIONSTASTER. I C128 mode er visse funktioner af tasterne forprogrammeret i forbindelse med opstart. Tasterne kan til enhver tid omprogrammeres; se afsnit 5 for detaljer.
For at få funktionerne på F2, F4, F6 og F8 (markeret på tasternes forside) frem, skal SHIFT nedtrykkes samtidig med den aktuelle funktionstast.

UDSKRIVNING AF GRAFISKE TEGN

For at udskrive det grafiske symbol på en tasts højre side, skal SHIFT nedtrykkes samtidig med, at der trykkes på den ønskede taste. OBS! Den 'højre' grafiske karakter kan kun udskrives, når man befinder sig i uppercase/graphic mode.
For at udskrive det grafiske symbol på tasternes venstre side, skal C= tasten være nedtrykket samtidig med, at tasten med det ønskede grafiske tegn nedtrykkes. Det 'venstre' grafiske tegn kan udskrives både i upper-case/graphic og upper/lowercase mode.

REGLER FOR SKRIVNING AF BASIC PROGRAMMER

De kan indtaste og bruge BASIC programmer uden nogen kendskab til BASIC.
De må imidlertid være meget omhyggelig med indtastningen, idet en skrivefejl kan betyde, at computeren ikke vil anerkende Deres oplysninger. Følgende retningslinier vil hjælpe med at undgå fejl ved skrivning eller kopiering af en programliste:

1. Antal mellemrum mellem ord er ikke kritisk; d.v.s. at indtastning af FORT=1TO10 er det samme som indtastning af FOR T=1 TO 10. Imidlertid må et BASIC ord ikke selv opdeles af mellemrum. (Se BASIC leksikon bag i denne bog for en liste over BASIC ord).
2. Enhver karakter kan indtastes omgivet af anførelsestegn. Imidlertid har visse karakterer specielle funktioner, når de skrives omgivet af anførelsestegn. Disse funktioner forklares senere i denne vejledning.
3. Vær forsigtig med skilletegn. Kommaer, koloner og semikoloner har også, som forklaret senere i dette afsnit, specielle funktioner.
4. Nedtryk altid RETURN tasten (i denne vejledning vist RETURN) efter at en linie er færdigskrevet.
5. Skriv aldrig mere end 160 tegn i en enkelt programlinie. Husk at det svarer til fire skærmlinier i 40 kolonners format, eller to linier i 80 kolonners format. Se afsnit 8 for flere detaljer.
6. Pas på ikke at forveksle bogstavet I med tallet 1 og bogstavet O med tallet 0.
7. Computeren ignorerer alt hvad der følger efter bogstaverne REM i en programlinie. REM er en forkortelse af REMARK

(bemærkning). REM instruktionen kan bruges til indsættelse af kommentarer i Deres program for verbalt at forklare, hvad der sker på dette sted i programmet.

Følg disse retningslinier, når De indtaster de eksempler og programmer, der findes i dette afsnit.

KOM IGANG - PRINT KOMMANDOEN

PRINT kommandoen giver computeren besked på at vise information på skærmen. De kan udskrive tal og bogstaver, men anvendelsesreglerne er forskellige, som beskrevet i følgende afsnit.

UDSKRIVNING AF TAL

For at få udskrevet tal, skal De simpelthen bruge PRINT kommandoen efterfulgt af de(t) tal, De ønsker skrevet på skærmen. Prøv at taste følgende på Deres Commodore 128:

PRINT 5

Tryk derefter på RETURN tasten. Læg mærke til at kun tallet 5 (uden ordet PRINT) vises på skærmens næste linie.

Indtast nu følgende og tryk på RETURN:

PRINT 5,6

I denne PRINT kommando fortæller kommaet Commodore 128, at De ønsker at få udskrevet mere end eet tal. Hvis computeren finder kommaer i en talrække bliver hvert tal, som følger efter et komma, udskrevet 10 mellemrum til højre for det foregående tal. Hvis De ikke ønsker disse ekstra mellemrum, skal De i stedet for komma bruge semikolon (;) i Deres PRINT instruktioner. Semikolon fortæller computeren, at tallene skal anbringes med kun 3 mellemrum. Indtast disse eksempler og se, hvad der sker:

PRINT 5;6 RETURN

PRINT 100;200;300;400;500 RETURN

BRUG AF SPØRGSMÅLSTEGN SOM FORKORTELSE FOR PRINT

Som forkortelse for PRINT kommandoen kan De bruge et

spørgsmålstegn (?). I virkeligheden kan de fleste BASIC kommandoer forkortes. Forkortelserne for BASIC kommandoerne kan findes i Appendiks K i denne vejledning.
Resten af eksemplerne i dette afsnit anvender ? symbolet i stedet for ordet PRINT.

UDSKRIVNING AF TEKST

Da De nu ved, hvordan man udskriver tal, er det på tide at lære, hvorledes man udskriver tekster. Det er i virkeligheden meget simpelt - der er meget færre regler for udskrivning af tekst end for tal. De indtaster blot PRINT kommandoen og skriver derefter, hvad De ønsker at se på skærmen, omgivet af anførelsestegn (""). BASIC's navn for et sæt af tegn omgivet af anførelsestegn er en STRENG. Anførelsestegn fås ved samtidig at nedtrykke SHIFT og taltasten 2. Prøv følgende eksempler:

? "COMMODORE 128" RETURN

? "4*5" RETURN

Læg mærke til, at computeren, når De trykke på RETURN, placerer det som står i anførelsestegnene, på skærmen, nøjagtigt som De indtastede det. Læg også mærke til forskellen mellem det forrige eksempel og dette:

? 4*5 RETURN

De kan PRINTe alt, hvad De ønsker, på skærmen ved brug af PRINT kommandoen, med anførelsestegn. De kan kombinere tekst og kalkulationer i en PRINT kommando.

? "4*5= "4*5 RETURN

Se nu hvorledes computeren udskriver det, De skrev i anførelsestegn, foretager kalkulationen og udskriver resultatet. Det har ingen betydning, om tekst eller kalkulation kommer først. I virkeligheden kan De anvende begge flere gange i een PRINT kommando.

?4*(2+3)" ER DET SAMME SOM "4*5 RETURN

Læg mærke til, at selv mellemrum indenfor anførelsestegnene anføres på skærmen. Tast:

? " KOM HER" RETURN

UDSKRIVNING I FORSKELLIGE FARVER

Commodore 128 er i stand til at vise 16 forskellige farver på skærmen. De kan meget let ændre disse farver. Alt hvad De skal gøre er at holde CTRL tasten nedtrykket og vælge en taltast mellem nul og otte på tastaturets øverste række. Læg mærke til, at markøren skifter farve, når De trykker på de forskellige tal. Alle efterfølgende tegn vises i den farve, De har valgt. Hold Commodore tasten (C=) nede og tryk på en taltast mellem nul og otte, og yderligere otte farver kan vælges på skærmen.
Herunder vises de farver, der kan fås i C128 mode, for såvel 40 som 80 kolonners format. Tabellen viser også tastsekvensen (CTRL + taltast eller C= tast + taltast) som skal bruges for en given farve.

Farvekode	Farve	Farvekode	Farve
1	Sort	9	Orange
2	Hvid	10	Brun
3	Rød	11	Lyserød
4	Cyanblå	12	Mørkegrå
5	Purpur	13	Mellemgrå
6	Grøn	14	Lysegrøn
7	Blå	15	Lyseblå
8	Gul	16	Lysegrå

Farvenumre i 40 kolonners format

Farvekode	Farve	Farvekode	Farve
1	Sort	9	Mørk purpur
2	Hvid	10	Mørkegul
3	Mørkerød	11	Lyserød
4	Lys cyanblå	12	Mørk cyanblå
5	Lys purpur	13	Mellemgrå
6	Mørkegrøn	14	Lysegrøn
7	Mørkeblå	15	Lyseblå
8	Lysegul	16	Lysegrå

Farvenumre i 80 kolonners format

BRUG AF MARKØRTASTER SAMMEN MED PRINT KOMMANDOEN

Når De taster markørtasterne omgivet af anførelsestegn, vises på skærmen nogle grafiske tegn, som repræsenterer disse taster. Disse tegn vil ikke blive udskrevet på skærmen, når De taster RETURN. Prøv at indtaste et ?
(spørgsmålstegn) og eet anførelsestegn ("); tryk derefter på 'markør ned' tasten 10 gange og indtast derefter ordet HERNEDE, og afslut med et anførelsestegn. Linien skulle se således ud:

? "QQQQQQQQQQHERNEDE"

Tryk nu på RETURN. Commodore 128 udskriver 10 blanke liner og på ellevte linie skriver den "HERNEDE". Som dette eksempel viser, kan De få computeren til at udskrive hvor som helst på skærmen, De ønsker det.

PROGRAM MODE FUNKTIONER

Nu, hvor De er i stand til at kommunikere i DIREKTE mode med Deres Commodore 128, er det på tide at påbegynde næste trin: skrivning af et program.
En program er kun et sæt nummererede BASIC instruktioner, som giver computeren besked om, hvad De ønsker, den skal udføre. I et BASIC program kaldes disse instruktioner programinstruktioner eller programlinier.

LINIENUMRE

Linierne i et program nummereres, så computeren kan vide, i hvilken rækkefølge De ønsker dem udført. Som linienumre kan benyttes ethvert tal fra 0 til 63999. Læg mærke til, at der ikke kan bruges komma eller punktum i linienumre.

Mange af de kommandoer, De har lært at bruge i DIREKTE mode, kan let ændres til programinstruktioner. Indtast f. eks. følgende:

10 ? "COMMODORE 128" RETURN

Læg mærke til, at computeren ikke viste COMMODORE 128, da De trykkede på RETURN, som da De brugte PRINT kommandoen i DIREKTE mode. Dette er fordi det tal, 10, som står foran

PRINT symbolet (?) fortæller computeren, at De er i færd med at indtaste et BASIC program. Når De indtaster en linie med foranstillet linienummer, ved computeren, at den skal vente på, at De giver den besked på, hvad den derefter skal udføre. Computeren gemmer blot den nummererede instruktion og afventer det næste input fra Dem.

Prøv nu at taste RUN og tryk på RETURN. Computeren udskriver ordene COMMODORE 128 på skærmen. Dette er ikke det samme som at bruge PRINT kommandoen i DIREKTE mode. Hvad der her skete var, at DE NETOP HAR SKREVET OG UDFØRT DERES FØRSTE BASIC PROGRAM. Programmet findes stadigvæk i computerens hukommelse, så De kan køre det igen, hvis De ønsker det.

SKÆRMFREMVISNING AF DERES PROGRAM - LIST KOMMANDOEN

Deres program ligger stadig i Commodore 128's hukommelse. Slet nu skærmen ved samtidig at nedtrykke SHIFT og CLR/HOME tasterne. Skærmen tømmes. På dette tidspunkt vil De muligvis gerne se programmet udskrevet, for at være sikker på, at det stadig er i hukommelsen. BASIC har en kommando til dette formål, LIST kommandoen.

Tast LIST og tryk på RETURN. På skærmen fremkommer:

```
10 PRINT "COMMODORE 128"
READY.
```

Når som helst De ønsker at se alle linier i Deres program, skal De blot taste LIST efterfulgt af tryk på RETURN.
En vigtig anvendelse af LIST er at kunne kontrollere eventuelle ændringer, der er lavet i en linie eller et program. Som svar på kommandoen vil computeren fremvise den ændrede version af linien, linierne eller programmet.
Herunder findes reglerne som gælder for LIST kommandoen.

REGLER FOR BRUG AF LIST KOMMANDOEN

Hvis kun linie X skal vises, tastes LIST X RETURN. (X erstattes af det ønskede linienummer.

Skal linierne fra nummer X til programslut vises, tastes LIST X- RETURN.

Skal linierne fra programstart til linie X vises, tastes LIST -X RETURN .

For at se fra linie X t.o.m. linie Z, tastes LIST X-Z RETURN .

EN SIMPEL LØKKE - GOTO KOMMANDOEN

Udover af at kunne fastlægge rækkefølgen af Deres kommandoer til computeren, har et programs linienumre også et andet formål. De virker som referencer for computeren, såfremt De ønsker at få kommandoen i en bestemt linie udført senere i Deres program. De bruger GOTO kommandoen til at fortælle computeren, at den skal gå til denne linie og udføre det, som står i linien. Tast nu:

20 GOTO 10

Når De trykker på RETURN efter at have skrevet linie 20, tilføjes linien det program, som ligger i computerens hukommelse.

Læg mærke til, at vi gav den første linie nummer 10 og den næste nummer 20. Det er meget nyttigt at nummerere programlinierne med spring på 10 (d.v.s. 10, 20, 30, 40, etc.), da det så senere er muligt at tilføje nye linier. Se endvidere RENUMBER og AUTO kommandoerne i BASIC ordbogen.

Skriv nu RUN og tryk på RETURN og De vil se ordene COMMODORE 128 blive skrevet ned over Deres skærm. For at standse udskrivningen kan De trykke på RUN/STOP tasten, som findes i tastaturets venstre side.
De to linier, De har indtastet, udgør et simpelt program, som vil gentage sig selv i det uendelige, fordi anden linie konstant vil vise tilbage til programmets første linie. Programmet vil køre i det uendelige, med mindre De standser det eller slukker for strømmen.

Skriv nu LIST RETURN . På skærmen vil fremkomme:

10 PRINT "COMMODORE 128"

20 GOTO 10

READY

Deres program findes stadig i hukommelsen. De kan køre det igen, hvis De ønsker det. Dette er en væsentlig forskel mellem

PROGRAM mode og DIREKTE mode, Er en kommando først udført i DIREKTE mode, findes den ikke længere i computerens hukommelse. Læg mærke til, at selv om De brugte ? symbolet for PRINT instruktionen, har computeren nu oversat det til den fulde kommando ordlyd. Dette sker, når De LISTer en hvilken som helst kommando, De har forkortet ved indtastningen af programmet.

SLETNING AF COMPUTERENS HUKOMMELSE - NEW KOMMANDOEN

Hvis De ønsker at starte forfra eller at slette et BASIC program, som findes i computerens hukommelse, skal De blot taste NEW RETURN. Denne kommando sletter computerens hukommelse, det areal hvori programmer lagres.

BRUG AF FARVER I ET PROGRAM

For at vælge farve i et program, skal De medtage farvevalget i en PRINT instruktion. Slet Deres computers hukommelse ved at taste NEW og trykke på RETURN. Indtast derefter følgende, og pas på at lave mellemrum mellem det enkelte bogstav:

10 PRINT "S P E K T R U M" RETURN

Indtast nu igen linie 10, men hold denne gang CTRL tasten nede og tryk på 1-tallet direkte efter at have tastet det første anførelsestegn. Slip CTRL tasten og tast "S". Hold nu CTRL nede igen og tryk på 2-tallet. Slip CTRL og tast "P". Hold derefter CTRL nede og tryk på 3-tallet. Fortsæt på denne måde, til De har skrevet alle bogstaver i ordet SPEKTRUM og valgt en farve mellem hvert enkelt bogstav. Nedtryk SHIFT og 2-tallet for at få det sidste anførelsestegn og tryk til sidst på RETURN. Computeren viser nu ordet SPEKTRUM på skærmen, med hvert bogstav i forskellig farve. Skriv nu LIST og tryk på RETURN. Læg mærke til de grafiske karakterer, som viser sig i PRINT instruktionen i linie 10. Disse karakterer fortæller computeren, hvilken farve det enkelte bogstav skal have. De grafiske tegn vises ikke, når Commodore 128 udskriver SPEKTRUM i forskellige farver.

Farvevalgskaraktererne, kaldet kontrolkaraktererne, i linie 10 i PRINT instruktionen fortæller Commodore 128, at der skal skif-

tes farve. Computeren udskriver så de tegn, der efterfølger, i den nye farve, indtil en ny kontrolkarakter mødes. Mens karakterer i anførelsestegn normalt udskrives nøjagtig som de er indtastet, vises kontrolkarakterer kun i forbindelse med program udLISTning.

PROGRAMÆNDRINGER

De følgende afsnit vil hjælpe Dem ved indtastning af Deres programmer og med, hvorledes De foretager ændringer og tilføjelser til dem.

SLETNING AF EN PROGRAMLINIE

Begynd med at bruge LIST kommandoen for at vise det netop indtastede program. Tast så 10 RETURN. Derved slettede De programmets linie 10. LIST programmet og se selv. Hvis den oprindelige linie 10 stadig står på skærmen, så flyt markøren op, så markøren står et eller andet sted på denne linie. Hvis De nu trykker på RETURN, vil linie 10 igen indsættes i computerens hukommelse.

GENTAGELSE AF EN PROGRAMLINIE

Hold SHIFT nedtrykket og tryk samtidig på CLR HOME tasten, som sidder øverst til højre på tastaturet. Derved slettes skærmbilledet. LIST nu programmet og læg mærke til, at linie 10 igen er med. Flyt markøren op, så den blinker over 0 i 10. Tast så et 5 tal og tryk på RETURN. De har derved fået linie 10 kopieret. Kopilinien har fået nummer 15. Skriv LIST og tryk på RETURN for at se programmet med den duplicerede linie.

UDSKIFTNING AF EN PROGRAMLINIE

En hel linie kan udskiftes ved at indtaste det gamle linienummer efterfulgt af en ny linie tekst og derefter trykke på RETURN. Den gamle version af linien vil blive fjernet fra hukommelsen og erstattet af den nye linie så snart De trykker på RETURN.

ÆNDRING AF EN PROGRAMLINIE

Forestil Dem, at De ønsker at tilføje et eller andet midt i en

linie. De flytter simpelthen markøren til det tegn eller mellemrum, der følger umiddelbart efter det sted, hvor De ønsker at indsætte det nye. Derefter holder De samtidig SHIFT og INST/DEL nedtrykkede, til der er plads nok til, at De kan indsætte de ønskede tegn.

Prøv dette eksempel. Slet først computerens hukommelse ved at taste NEW RETURN. Skriv derefter:

10 ? "JEG ELSKER MIN 128" RETURN

Lad os nu sige, at De ønsker at tilføje ordet COMMODORE foran tallet 128.
Flyt markøren, så den blinker over 1 tallet. Hold SHIFT og INST/DEL nedtrykkede, til der er plads nok til at skrive COMMODORE (husk at der også skal være plads til et mellemrum efter E). Skriv så ordet COMMODORE.

Ønsker De at slette noget i en linie (incl. ekstra blanke mellemrum), flyt så markøren til den karakter, som følger efter de tegn, De vil fjerne.
Tryk derefter INST/DEL tasten. Markøren vil flytte sig mod venstre og tegn eller mellemrum vil blive fjernede, så længe tasten holdes nedtrykket.

MATEMATISKE OPERATIONER

De kan bruge PRINT kommandoen for at få udført kalkulationer som addition og subtraktion. De skal blot indtaste kalkulationen efter PRINT kommandoen. Prøv nogle af disse:

ADDITION OG SUBTRAKTION

PRINT 6+4 RETURN
PRINT 50-20 RETURN
PRINT 10+15-5 RETURN
PRINT 75-100 RETURN
PRINT 30+40,55-25 RETURN
PRINT 30+40;55-25 RETURN

Læg mærke til, at den fjerde kalkulation (75-100) som resultat giver et negativt tal. Læg også mærke til, at De, med en enkelt PRINT kommando, kan give computeren besked på at udføre mere end een kalkulation. I Deres kommando kan De bruge enten et komma eller et semikolon, afhængig af hvor De ønsker resultatet udskrevet.

MULTIPLIKATION OG DIVISION

Find asterisk tasten (*) i højre side af Deres tastatur. Det er dette symbol, Commodore 128 bruger i forbindelse med multiplikation. Skråstregen (/) ved siden af højre SHIFTe tast, bruges som divisionssymbol.

Prøv disse eksempler:

PRINT 5*3 RETURN
PRINT 100/2 RETURN

POTENSOPLØFTNING

eller eksponentation betyder at opløfte et tal til en potens. Den lodrette piltast (↑), som er placeret ved siden af asterisken på tastaturet, bruges som symbol herfor. Hvis De ønsker at foretage potensopløftning af et tal, bruges PRINT kommandoen, fulgt af tallet, derefter ↑ og tilsidst potensen.
Ønsker De for eksempel at udregne kvadratet på 3, skal De indtaste:

PRINT 3↑2 RETURN

BEREGNINGSRÆKKEFØLGE

De har nu set, hvorledes De kan kombinere addition og subtraktion i samme PRINT kommando. Hvis De kombinerer multiplikation eller division med additions- eller subtraktionsoperationer, får De måske ikke det forventede resultat. Indtast for eksempel:

PRINT 4+6/2 RETURN

Hvis De forestillede Dem, at De dividerede 10 med 2, blev De formentlig overrasket, da computeren svarede med tallet 7. Grunden til, at De fik dette resultat, er, at multiplikations- og divisionsoperationer udføres af computeren før additioner og subtraktioner. Det betyder intet i hvilken rækkefølge operationerne er indtastet. Denne fremgangsmåde er almindeligt benyttet indenfor al databehandling. Multiplikation og division siges at have højere prioritet end addition og subtraktion.

Eksponentation, eller potensopløftning, har højere prioritet end de fire andre matematiske funktioner. Hvis De for eksempel indtaster:

PRINT 16/4↑2 RETURN

vil Commodore 128 svare med et ettal (1), fordi den før divisionen foretages beregner kvadradet på 4.

BRUG AF PARANTESER FOR DEFINITON AF OPERATIONS-RÆKKEFØLGE

Ved at omslutte en operation i paranteser () i en PRINT kommando, kan De fortælle Commodore 128, at De ønsker denne operation udført først. Hvis De, i det første eksempel ovenfor, ønsker at computeren skal addere, før den dividerer, skal De indtaste:

PRINT (4+6)/2 RETURN

Dette giver Dem det korrekte resultat, nemlig 5.
Hvis De, i det andet eksempel, ønsker, at computeren skal dividere før den beregner kvadradet, taster De:

PRINT (16/4)↑2 RETURN

De får nu det korrekte svar, 16.
Hvis De ikke anvender paranteser, vil computeren udføre kalkulationerne i henhold til de tidligere nævnte regler. Har alle operationer i en kalkulation samme prioritet, bliver de udført fra venstre mod højre. Tast for eksempel:

PRINT 4*5/10*6 RETURN

Da operationerne i dette eksempel udføres i rækkefølge fra venstre mod højre, bliver resultatet 12. Ønsker De at dividere 4*5 med 10*6, skal De i stedet indtaste:

PRINT (4*5)/(10*6) RETURN

og resultatet bliver nu .333333333

KONSTANTER, VARIABLER OG STRENGE

KONSTANTER

Konstanter er numeriske værdier, som er permanente: d.v.s. at de ikke skifter værdi på grund af hændelser i programmet. For eksempel er tallet 3 en konstant, som alle andre tal. Indtast dette program, hvor instruktionen illustrerer, hvorledes Deres computer bruger konstanter:

```
10 PRINT 3
```

Uanset hvor mange gange de udfører denne linie, vil svaret være 3.

VARIABLER

Variabler er værdier, som kan ændre sig på grund af beregninger eller programinstruktioner. En del af computerens BASIC hukommelse er reserveret til de karakterer (tal, bogstaver og symboler), De bruger i Deres program.
Tænk på hukommelsen som et antal lagerhylder i computeren, hvor der gemmes informationer om Deres program; denne del af hukommelsen kaldes variabellageret. Indtast følgende program:

```
10 X = 5
20 ?X
```

Kør nu programmet med RUN og se, hvorledes computeren skriver et 5-tal på Deres skærm. De har i linie 10 fortalt computeren, at bogstavet X i resten af programmet skal repræsentere værdien 5. Bogstavet X kaldes en variabel, fordi værdien af X varierer afhængig af den værdi, der sættes efter lighedstegnet. Vi kalder dette en 'tildelings instruktion', fordi der nu findes en lageradresse (hylde) i computerens hukommelse, og denne adresse har fået tildelt værdien 5. Lighedstegnet fortæller computeren, at ligegyldigt hvad der kommer på højre side af lighedstegnet, skal dette tildeles til en lagerhylde (hukommelses adresse) mærket X, som står til venste for =.

Variabelnavnet på venstre side af = tegnet kan være enten et eller to bogstaver, eller et bogstav og et tal (bogstavet skal i så tilfælde stå først). Navnene kan være længere, men computeren tager kun hensyn til de to første karakterer. Dette betyder

at navnene PA og PART refererer til samme lageradresse. Ord, som bruges som BASIC kommandoer (LOAD, RUN, LIST etc.) eller funktioner (INT, ABS, SQR, etc.) kan ikke bruges som navne i Deres programmer. Er De i tvivl om et ord er et BASIC nøgleord, så prøv af se i BASIC 7.0 i kapitel 5. Læg mærke til, at = tegnet i tildelings instruktioner ikke er det samme som det matematiske symbol, som betyder "lig med", men snarere betyder at en variabel skal oprettes og at den skal have tildelt en værdi.

I det programeksempel, De netop har indtastet, kaldes X en konstant, fordi den altid repræsenterer værdien 5. For at gemme et resultat i en æske, kan De indsætte kalkulationer efter lighedstegnet. I en PRINT instruktion kan De blande tekst med konstanter for at identificere dem. Tast nu NEW for at slette hukommelsen og prøv så dette program:

```
10 A = 3*100
20 B = 3*200
30 ? "A ER LIG MED "A
40 ? "B ER LIG MED "B
```

Der findes nu to variabler, mærket A og B, i computerens hukommelse, og de indeholder henholdsvis værdierne 300 og 600. Hvis De, senere i programmet, ønsker at ændre en konstants værdi, skal De blot indtaste en ny tildelingsinstruktion i programmet. Prøv at tilføje disse programlinier og kør det igen:

```
50 A = 900*30/10
60 B = 95+32+128
70 GOTO 30
```

Det vil igen være nødvendigt at trykke på STOP for at standse programmet.

LIST nu programmet og undersøg, hvad computeren udfører. Først tildeler den bogstavet A den værdi, der står på højre side af = tegnet i linie 10.
Det samme gør den med bogstavet B i linie 20. Derefter udskriver den meddelelserne i linierne 30 og 40, som giver Dem værdierne af A og B. Til sidst tildeler den, i linierne 50 og 60, A og B nye værdier. De gamle værdier fjernes og kan kun genskabes, hvis computeren atter udfører linierne 10 og 20 (hvilket ikke sker i dette program). Når computeren sendes til linie 30

for at udskrive værdierne i A og B vil den skrive de nye værdier, som blev beregnet i linierne 50 og 60. Linierne 50 og 60 tildeler A og B de samme værdier og linie 70 sender computeren tilbage til linie 30.
Dette kaldes en endeløs løkke (loop), fordi linier 30 til 70 udføres igen og igen, til De trykker på RUN/STOP for at stoppe programmet. Senere i dette og i de to følgende kapitler gennemgås andre løkkemetoder.

STRENGE

En streng er en karakter eller en gruppe karakterer, omsluttet af anførelsestegn. Disse karakterer lagres i computerens hukommelse på næsten samme måde som numeriske variabler. De kan også bruge navne for at repræsentere strenge, nøjagtig som De bruger navne for at repræsentere tal. Hvis De sætter et dollartegn ($) efter strengnavnet, oplyser dette computeren om, at der er tale om navnet på en strengvariabel og ikke en numerisk variabel.

Slet computerens hukommelse og indtast nedenstående program:

```
10 A$ = "COMMODORE"
20 X = 128
30 B$ = "COMPUTER"
40 Y = 1
50 ? "MIN " A$;X;B$" ER NUMMER "Y
```

Her kan De se, hvorledes De kan skrive numeriske og streng variabler i samme instruktion. Prøv at eksperimentere med variabler i Deres egne korte programmer.

Efter at et program er kørt, kan De, i DIREKTE mode, udskrive værdien af en variabel i programmet. Tast ? A$;B$;X;Y efter kørsel af programmet og De vil se, at de tre 'æsker' stadig findes i computerens hukommelse.

Hvis De ønsker at nulstille dette areal i BASIC's hukommelse, men stadig ønsker at bevare programmet intakt, skal De bruge CLR kommandoen. Indtast CLR RETURN og alle konstanter, variabler og strenge slettes. Men taster De nu LIST, vil De se, at programmet endnu findes i hukommelsen. Den tidligere om-

talte NEW kommando sletter både programmet samt variablerne.

PROGRAMEKSEMPEL

Her følger et programeksempel, som indeholder mange af de teknikker og kommandoer, der er gennemgået i dette afsnit.

Programmet beregner gennemsnittet af tre variabler (X, Y, og Z) og udskriver deres værdier og deres gennemsnit på skærmen. De kan redigere programmet og ændre kalkulationerne i linie 10 til 30 ved at ændre variablernes værdier. Linie 40 adderer variablerne og dividerer med 3 for at beregne deres gennemsnit. Læg mærke til anvendelsen af paranteser, som fortæller computeren, at den skal addere tallene før den udfører divisionen.

> TIP: Når som helst De anvender mere end eet sæt paranteser i en instruktion, er det en god ide at tælle antallet af venstreparanteser og højreparanteser for at være sikker på, at der er lige mange.

```
10 X=46
20 Y=72
30 Z=114
40 A=(X+Y+Z)/3
50
60 ? "GENNEMSNITTET AF";X;Y;"OG" Z"ER "A
70 GOTO 90
90 END
```

LAGRING OG GENANVENDELSE AF PROGRAMMER

Når De har redigeret Deres program, ønsker De måske at gemme det permanent, så De senere vil være i stand til at genindlæse og bruge det. For at kunne dette, må De have enten en Commodore diskettestation eller Datassette båndstation.

De vil lære adskillige kommandoer, som sætter Dem i stand til at foretage kommunikation mellem Deres computer og diskettestation/båndstation. Disse kommandoer består af et kommandoord efterfulgt af flere parametre. Parametere er bogstaver, ord

eller symboler i en kommando, som giver computeren nøjagtige oplysninger, som f. eks. et filnavn, eller en numerisk variabel, som angiver et enhedsnummer. Hver kommando kan have forskellige parametere. For eksempel indeholder parametre i en disketteformatterings-kommando diskettens navn, identifikationskode plus flere andre oplysninger.
Parametere anvendes i næsten alle BASIC kommandoer; nogle er variabler som ændres og andre er konstanter. Herunder vises de parametere, som indeholder oplysninger til C128 og diskettestationen:

DISKHÅNDTERINGS PARAMETERE

diskettenavn	-	et identifikationsnavn på højst 16 tegn
filnavn	-	et identifikationsnavn på højst 16 tegn
i.d. nummer	-	en id-kode på to tegn
drevnummer	-	skal være 0 for enkelt diskettestation 0 eller 1 for dobbelt diskettestation
enhedsnummer	-	et forud bestemt nummer for en ydre enhed.
	-	For eksempel har Commodore diskette-
	-	stationen nr. 8

FORMATTERING AF EN DISKETTE - HEADER KOMMANDOEN

For at kunne lagre programmer på en ny (blank) diskette, skal den først forberedes til at kunne modtage data. Dette kaldes 'formattering' af en diskette. Husk, at diskettestationen skal være tændt, før en diskette isættes.

Formatteringsprocessen inddeler disketten i sektioner, som kaldes spor og sektorer. En indholdsfortegnelse, kaldet Directory, dannes. Hver gang De lagrer et program på en diskette, bliver programmets navn føjet til indholdsfortegnelsen.
Commodore 128 har to slags formatteringskommandoer. Den ene kan bruges i C128 mode, og den anden kan bruges såvel i C64 som C128 mode. Følgende afsnit gennemgår formatkommandoerne for C128 mode. Se kapitel 3, C64 mode, for yderligere oplysninger om C64 programmering og diskkettehåndtering.

Kommandoen, som formatterer en diskette kaldes HEADER kommandoen. Den findes i en lang og en kort form. For at formattere en blank diskette SKAL den lange form bruges, som følger:

HEADER "diskettenavn" [,Ii.d."] [,Ddrevnummer] [,[ON]Uenhedsnummer]

Efter ordet HEADER indtastes en navn for disketten, i anførelsestegn. De kan bruge et hvilket som helst navn på op til 16 karakterers længde. Vælg diskettenavne, som fortæller noget om, hvad der lagres på disketten.
Diskettenavnet skal efterfølges af et komma og bogstavet "I". Derefter en id-kode på to tegn efterfulgt af et komma. Id-koden behøver ikke at bestå af tal; bogstaver kan også benyttes. De kan selv bestemme, hvad De ønsker, f. eks. 01, 76, A1, B3 etc.

Hvis De har en enkelt diskettestation, tastes nu blot RETURN, idet Commodore 128 automatisk forstår, at der dermed menes drev nummer 0 på enhed nummer 8. Hvis De har mere end een diskettestation eller et dobbeltdrev, kan disse parametere specificeres.

Næste parameter i kommandoen vælger drevnummeret. Tast "D" og, hvis De har en enkelt diskettestation, 0 efterfulgt af et komma. Dobbeltdrev har numrene 0 og 1. Enhedsnummer parameteren begynder med U, så tryk på "U" efterfulgt af det forud bestemte drevnummer, som er 8.

Her er et eksempel på den lange HEADER kommando:

HEADER "POSTER",IA1,D0,U8 RETURN

Denne kommando formatterer disketten, kalder indholdsfortegnelsen POSTER, id-nummeret A1, på drev 0, enhed 8.
Hvis ikke andet angives, vil standardværdierne 0 og 8 blive brugt for henholdsvis drev og enhed. Dette er en acceptabel form af den lange kommando:

HEADER "MINDISK",I23 RETURN

HEADER kommandoen kan også bruges til at fjerne alle data på en brugt diskette, så disketten kan genbruges, som var den ny. Pas på, at De ikke sletter en diskette, som indeholder data, som De senere vil bruge.

Den kortere form af HEADER kommandoen kan kun bruges, hvis disketten tidligere har været formatteret.

Den korte form sletter indholdsfortegnelsen, fjerner alle data, men fastholder det tidligere id-nummer. Kommandoen kan se således ud:

HEADER "PROGRAMMER" RETURN

DSAVE OG SAVE KOMMANDOERNE

For at lagre Deres programmer på diskette eller bånd kan DSAVE og SAVE kommandoerne anvendes.

LAGRING PÅ EN DISKETTE MED SAVE

I C128 mode kan programmer lagres på diskette ved anvendelse af een af de følgende kommandoer:

DSAVE "PROGRAMNAVN" RETURN
SAVE "PROGRAMNAVN",8 RETURN

Begge kommandoer kan bruges. Husk at ordet "DSAVE" kan blive vist på skærmen ved at nedtrykke funktionstast F5, eller De kan selv indtaste det.
PROGRAMNAVN kan være ethvert navn, De måtte ønske. De kan bruge bogstaver, tal og/eller symboler - op til 16 karakterer ialt. Vær opmærksom på, at programnavnet skal være omgivet af anførelsestegn. De kan ikke lagre to programmer med samme navn på disketten. Prøver De på det, vil det andet program ikke blive accepteret; disketten vil beholde det første.
I det andet eksempel angiver 8 tallet, at der bruges diskettestation til lagring af programmet. Denne angivelse er unødvendig, når DSAVE bruges.

LAGRING PÅ KASSETTEBÅND MED SAVE

Hvis De bruger en Datassette til programlagring, skal De sætte en blankt bånd i båndstationen, tilbagespole båndet og indtaste:

SAVE "PROGRAMNAVN" RETURN

De skal indtaste ordet SAVE efterfulgt af programnavnet, som kan være op til 16 karakterer langt.

> OBS! Mens programmet lagres, vil Deres skærm være blank, men når processen er overstået, fremkommer skærmbilledet igen.

Til forskel fra lagring på diskette kan De på bånd lagre to programmer med samme navn. Når det indlæses til computeren, vil det først fundne program på båndet blive indlæst, så undgå sammenfaldende navne.

Når et program er lagret med SAVE, kan det indlæses med LOAD og når som helst udføres med RUN.

INDLÆSNING FRA DISKETTE MED LOAD

Indlæsning af et program består simpelthen i kopiering af programmet på disketten til computerens hukommelse. Findes der allerede et BASIC program i hukommelsen, vil dette blive slettet.

For at indlæse Deres program fra en diskette, kan De anvende een af følgende kommandoer i C128 mode.

DLOAD "PROGRAMNAVN" RETURN
LOAD "PROGRAMNAVN",8 RETURN

Husk at De i C128 mode kan bruge F2 funktionstasten for at få DLOAD frem på skærmen, eller De kan selv taste ordet.
I eksempel 2 viser 8-tallet igen, at computeren skal hente programmet fra en diskettestation. Som ved DSAVE, forudsætter DLOAD, at enhedsnummeret er 8. Vær omhyggelig med at stave programnavnet nøjagtigt, som da det blev lagret, ellers vil computeren svare "FILE NOT FOUND".

Når programmet er indlæst tastes RUN for at få det udført. Commodore 128 har en speciel version af RUN kommandoen, som både indlæser og udfører programmet med een kommando. Tast RUN efterfulgt af programnavnet (filnavnet) i anførelsestegn:

RUN "PROGRAMNAVN" RETURN

INDLÆSNING FRA KASSETTEBÅND

For indlæsning fra kassettebånd tastes:

LOAD "PROGRAMNAVN" RETURN

Hvis De ikke kender programnavnet, kan De taste:

LOAD RETURN

og det næste program på båndet vil blive fundet. Mens Datassetten leder efter programmet, er skærmen blank. Når programmet er fundet, vises på skærmen:

FOUND PROGRAMNAVN

For at indlæse programmet, skal De trykke på Commodore tasten (C=). I C128 mode skal mellemrumstangenten nedtrykkes for at finde det næste program.
Til at finde startpositionen for et program kan De bruge tælleren på Datassetten. For at finde et program skal De så fremspole båndet fra 000 til den aktuelle startposition og taste:

LOAD RETURN

I dette tilfælde behøver De ikke at angive programnavnet; Deres program vil automatisk indlæses, fordi det er det næste på båndet.

ANDRE DISKETTE RELATEREDE KOMMANDOER

VERIFIKATION AF ET PROGRAM

For at verificere at et program er blevet korrekt lagret eller indlæst, kan følgende kommando anvendes i C128 mode.

DVERIFY "PROGRAMNAVN" RETURN

Hvis programmet på disketten er identisk med det, der findes i computeren, vil skærmen vise "OK".

VERIFY kommandoen kan også bruges med båndprogrammer. Her taster De:

VERIFY "PROGRAMNAVN" RETURN

Indtast ikke komma og enhedsnummer.

VISNING AF DISKETTENS INDHOLD PÅ SKÆRM

I C128 mode kan De se en liste (directory) over de programmer, der findes på Deres diskette ved anvendelse af een af følgende kommandoer:

DIRECTORY RETURN

Dette udlister directory's indhold. Den nemmeste måde er at trykke på F3 funktionstasten. Når De trykker på F3, viser C128 "DIRECTORY" og udfører kommandoen.

For flere oplysninger om SAVE og LOAD, eller andre diskette-operationer, bedes De se i brugervejledningerne til Deres diskettestation eller Datassette, samt i kapitel 5, BASIC 7.0.

****

De ved nu noget om programmeringssproget BASIC og nogle elementære programmeringsregler. Næste afsnit bygger videre på disse koncepter, introducerer nye og mere kraftfulde kommandoer, funktioner og teknikker, som De kan bruge ved programmering i BASIC.

AFSNIT 4
AVANCERET BASIC PROGRAMMERING

Dette afsnit introducerer avancerede BASIC kommandoer, funktioner og programmeringsteknikker, som kan bruges såvel i C64 som C128 mode.

Disse kommandoer og funktioner sætter Dem i stand til at kommunikere mere direkte med Commodore 128. De kan programmere repeterende operationer ved hjælp af avancerede løkker og sammenbyggede løkker; håndtere værditabeller; forgrene eller hoppe til et andet afsnit i et program og også vende tilbage fra dette afsnit; tildele en kvantitet forskellige værdier -- og mere. Eksempler og programeksempler viser, hvorledes disse BASIC koncepter arbejder.

IF - THEN INSTRUKTIONEN

Nu, hvor De er klar over, hvordan værdierne af variabler ændres, vil næste trin bestå i at få computeren til at foretage beslutninger baseret på disse opdaterede værdier. Dette gøres ved anvendelse af IF - THEN instruktionen (HVIS - SÅ). De giver computeren besked på at udføre en kommando, men kun hvis en betingelse er opfyldt (er sand), f. eks. IF X = 5. Den kommando, De ønsker, computeren skal udføre, når betingelsen er sand, kommer efter ordet THEN i instruktionen. Slet Deres computers hukommelse ved at taste NEW RETURN , og indtast derefter dette program:

```
10 J = 0
20 J = J+1
30 ? J,"COMMODORE 128"
40 IF J = 5 THEN GOTO 60
50 GOTO 20
60 END
```

Det er nu ikke længere nødvendigt at trykke på STOP tasten for at afslutte et sådant løkke-program. IF - THEN instruktionen fortæller computeren, at den skal blive ved med at udskrive "COMMODORE 128" og forøge J, til (J=5) er sandt. Når IF betingelsen bliver falsk, springer computeren til næste programlinie, uanset hvad der følger efter ordet THEN.

Læg mærke til END kommandoen i linie 60. Det er god praksis at lægge END instruktionen i slutningen af Deres program. Derved får computeren at vide, hvor den skal standse udførelse af instruktioner.

Herunder vises en liste med de symboler, der kan bruges i IF instruktioner samt deres betydning:

SYMBOL	BETYDNING
=	**LIG MED**
>	**STØRRE END**
<	**MINDRE END**
< >	**FORSKELLIG FRA**
> =	**STØRRE END ELLER LIG MED**
< =	**MINDRE END ELLER LIG MED**

Vær opmærksom på, at disse sammenligningskriterier arbejder på den måde, som er normalt ved matematisk behandling af tal. Se om emnet 'strengbehandling' i kapitel 5, BASIC 7.0.

Afsnit 5 gennemgår nogle kraftfulde udvidelser af IF-THEN konceptet, som f. eks. BEGIN, BEND og ELSE.

BRUG AF KOLON (:)

Et andet nyttigt værktøj ved programmering er kolon (:). De kan bruge kolon for at adskille to eller flere BASIC kommandoer i samme linie.

Instruktioner efter et kolon i en linie vil blive udført i rækkefølge, fra venstre mod højre. I en programlinie kan skrives så mange instruktioner, som kan være indenfor 160 karakterer. Dette svarer til fire fulde skærmlinier i 40 kolonners format og to linier i 80 kolonners format . Dette giver en glimrende mulighed for at få fuldt udbytte af THEN delen af IFTHEN instruktionen. De kan få computeren til at udføre adskillige kommandoer, hvis Deres IF betingelse er sand. Slet computerens hukommelse og indtast følgende program:

```
10 N = 1
20 IF N<5 THEN PRINT N;"MINDRE END 5":GOTO 40
30 ? N;"STØRRE END ELLER LIG MED 5"
40 END
```

Nu skal De ændre linie 10, så der står N = 20, og køre programmet igen.
Læg mærke til, at De kan give computeren besked om at udføre mere end een instruktion, når N er mindre end 5. De kan indsætte netop den eller de instruktioner, De ønsker, efter THEN kommandoen. Husk, at GOTO 40 ikke vil blive udført, før N⟨5 er sandt. Enhver kommando som ønskes udført, hvad enten den anførte betingelse opfyldes eller ej, skal stå på en separat linie.

LØKKER - FOR-NEXT KOMMANDOEN

Kan De huske programeksemplet i IF - THEN eksemplet? Vi fik computeren til at udskrive COMMODORE fem gange ved at forøge variablen J med enheder på 1, indtil værdien af J blev lig med 5; der sluttede programmet. I BASIC er der en enklere metode til at gøre dette. Man kan lave en FOR-NEXT løkke, som herunder:

```
10 FOR J=1 TO 5
20 ? J,"COMMODORE"
30 NEXT J
40 END
```

Kør dette program og sammenlign resultatet med resultatet af IF-THEN programmet -- de er ens! I virkeligheden er computerens virkemåde stort set den samme i de to programmer. FOR-NEXT løkken er et meget kraftfuldt programmeringsværktøj. De kan give computeren besked på hvormange gange en programudførelse skal gentages. Lad os følge, hvad computeren trin for trin foretager sig med ovennævnte program.

Først tildeler computeren variablen J en værdi på 1. 5-tallet i FOR instruktionen i linie 10 giver computeren besked på at udføre alle instruktioner mellem FOR instruktionen og NEXT instruktionen, indtil J er lig 5.
I dette tilfælde er der kun een instruktion - PRINT instruktionen.

Efter at computeren har tildelt J værdien 1, fortsætter den programudførelsen af PRINT instruktionen. Når computeren når til NEXT J instruktionen bliver J forøget og sammenlignet med 5. Har J ikke nået værdien 5, vender computeren tilbage til PRINT instruktionen. Efter 5 udførelser af denne løkke vil J være lig

med 5. Her går computeren videre til den instruktion, som følger lige efter NEXT instruktionen og fortsætter herfra. I dette tilfælde er der tale om END kommandoen, så programudførelsen standser.

TOMME LØKKER (LOOPS) FOR FORSINKELSE

Før De fortsætter, er det praktisk, at De ved noget om løkker og hvorledes de anvendes til at få computeren til at gøre, hvad De ønsker. De kan bruge en løkke til at få computeren til at arbejde langsommere - De har jo nu set den hastighed, hvormed computeren udfører kommandoer. Se om De kan forudse, hvad dette program gør, før De kører det.

```
10 A$="COMMODORE 128"
20 FOR J=1 TO 20
30 PRINT
40 FOR K=1 TO 1500
50 NEXT K
60 PRINT A$
70 NEXT J
80 END
```

Skete der, hvad De ventede? Løkken i linie 40 og 50 giver computeren besked på at tælle til 1500, før den udfører resten af programmet. Dette kaldes en 'forsinkelses-løkke' og kan være nyttigt for Dem i Deres programmer. En løkke der således ligger indeni i hovedløkken kaldes 'nested loop' eller 'kædet løkke'. En kædet løkke kan være meget nyttig, hvis De ønsker, at computeren skal udføre en given opgave flere gange. Afsnit 5 gennemgår en avanceret måde, hvorpå man kan indsætte forsinkelser ved brug af den nye BASIC 7.0 kommando, SLEEP.

STEP KOMMANDOEN

De kan give computeren besked på at forøge Deres tæller i enheder på 10, 0.5 eller andre størrelser. Dette gøres ved at bruge en STEP kommando i forbindelse med FOR instruktionen. Ønsker De f. eks. at computeren skal tælle til 100 i trin på 10, skal De indtaste:

```
10 FOR X=0 TO 100 STEP 10
20 ? X
30 NEXT
```

Læg mærke til, at De ikke behøver at medtage X i NEXT instruktionen, hvis De kun udfører een løkke ad gangen -- dette gennemgås senere i afsnittet.
Læg også mærke til, at De ikke kun kan forøge tælleren -- De kan også formindske den. Prøv for eksempel at ændre linie 10 i ovenstående program til:

```
10 FOR X=100 TO 0 STEP -10
```

Computeren vil nu tælle baglæns fra 100 til 0, i trin på 10. Hvis De ikke bruger en STEP kommando i en FOR instruktion, vil computeren automatisk forøge tælleren med 1.

I FOR-NEXT kommandoen indgår følgende:

FOR	-	ordet fortæller, at her skal løkken begynde
X	-	tællervariabel; enhver talvariabel kan benyttes
1	-	startværdi; kan være et positivt eller negativt tal
TO	-	forbinder startværdi og slutværdi
100	-	slutværdi; kan være et positivt eller negativt tal
STEP	-	indikerer, at en anden trinværdi end 1 vil blive brugt
2	-	trinværdi; kan være et positivt eller negativt tal

INPUT KOMMANDOEN

TILDELING AF EN VÆRDI TIL EN VARIABEL

Slet computerens hukommelse ved at taste NEW og trykke på RETURN, og kør derefter følgende program.

```
10 K=10
20 FOR I=1 TO K
30 ? "Commodore"
40 NEXT
```

I dette program kan De ændre værdien af K i linie 10 for at få computeren til at udføre løkken, så mange gange, De ønsker. Dette skal De naturligvis gøre under indtastningen, før De kører programmet. Men hvad nu hvis De, under kørslen af programmet, ønskede at fortælle computeren, hvor mange gange, løkken skulle udføres?

Med andre ord, De vil gerne være i stand til at ændre variablen K's værdi, hver gang De kører programmet, vel at mærke uden at ændre selve programmet. Vi kalder dette for at kunne arbejde 'interaktivt' med computeren. De kan lade computeren spør-

ge Dem, hvor mange gange løkken ønskes udført. For at gøre dette skal De bruge INPUT kommandoen. Udskift f. eks. linie 10 i programmet med:

```
10 INPUT K
```

Når De nu kører programmet, vil computeren vise et ? på skærmen, for at fortælle Dem, at den venter på, at De indtaster en værdi for K. Tast 15 og tryk på RETURN. Computeren vil så udføre løkken 15 gange.

SPØRGSMÅL PÅ SKÆRMEN (PROMPTS)

I en INPUT instruktion kan De også lade computeren udskrive en meddelelse om, hvilken variabel den forventer at modtage. Udskift linie 10 med:

```
10 INPUT "INDTAST EN VÆRDI FOR K";K
```

Husk at meddelelsen skal omgives af anførelsestegn. Vær også opmærksom på, at De skal bruge semikolon mellem sidste anførelsestegn og variablen K. En sådan meddelelse kan bestå af hvad som helst, men INPUT instruktionen må højst fylde 160 karakterer, eller mindre, som enhver anden BASIC kommando.

INPUT instruktionen kan også anvendes med strengvariabler. De regler som gælder for numeriske variabler, gælder også for strenge. Glem ikke at bruge $-tegnet ved angivelse af strengvariabler. Slet computerens hukommelse med NEW/RETURN og indtast følgende program:

```
10 INPUT "HVAD ER DIT NAVN";N$
20 ? "HEJ ";N$
```

Kør nu programmet med RUN. Når computeren spørger "HVAD ER DIT NAVN?", skal du indtaste navnet og derefter trykke på RETURN.

Når først en variabels værdi (numerisk eller streng) har været INPUTtet i et program, kan De hvor som helst i programmet referere til den med variabelnavnet. Indtast ?N$ -- De vil se, at computeren kan huske Deres navn!

GET KOMMANDOEN

Der findes andre BASIC kommandoer, som De kan bruge i Deres program, for at arbejde interaktivt med computeren. Een hedder GET og ligner INPUT meget. Slet computerens hukommelse og indtast følgende program:

```
10 GET A$
20 IF A$="" THEN 10
30 ? A$
40 END
```

Når De taster RUN og trykker på RETURN, sker der tilsyneladende intet.
Grunden hertil er, at computeren forventer, at De nedtrykker en eller anden taste. GET kommandoen fortæller computeren, at denne skal kontrollere tastaturet for at finde ud af, hvilken tast, der er blevet nedtrykket.

GET kommandoen er meget vigtig, da De kan bruge den til at programmere en tast på tastaturet. Nedenstående eksempel programmerer Q-tasten til at give en skærmmeddelelse. Indtast og kør programmet. Tryk derefter på Q og se hvad der sker.

```
10 ?"TRYK PÅ Q FOR AT SE MEDDELELSE"
20 GET A$
30 IF A$="" THEN 20
40 IF A$="Q" THEN 60
50 GOTO 20
60 FOR I=1 TO 25
70 ? "NU KAN JEG BRUGE INSTRUKTIONEN"
80 NEXT
90 END
```

Læg mærke til, at hvis De prøver at trykke på andet end Q, vil computeren ikke udskrive meddelelsen, men gå tilbage til linie 20 for at modtage et andet tegn.

Afsnit 5 gennemgår hvorledes GETKEY instruktionen bruges. Den er en ny kraftfuld instruktion i BASIC 7.0, som kan bruges til opgaver af ovenstående art.

PROGRAMEKSEMPEL

Nu, hvor De ved, hvorledes man bruger FOR-NEXT løkken og INPUT kommandoen, kan De lave dette program meget stærkere. Slet computerens hukommelse og indtast følgende program:

```
10 T=0
20 INPUT "HVOR MANGE TAL";N
30 FOR J=1 TO N
40 INPUT "INDTAST ET TAL";X
50 T=T+X
60 NEXT
70 A=T/N
80 PRINT
90 ? "DE HAR";N"TAL IALT";T
100 ? "GENNEMSNIT = ";A
110 END
```

De har nu mulighed for at fortælle computeren, hvor mange tal De ønsker at beregne gennemsnit af, og De kan ændre tallene, hver gang De kører programmet, uden at skulle ændre selve programmet.

Lad os gennemgå programmet linie for linie:

Linie 10 tildeler T værdien 0, T vil blive den løbende total af tallene.

Linie 20 lader Dem bestemme, hvor mange tal, der skal gennemsnitsberegnes på.

Linie 30 fortæller computeren hvor mange løkker, der skal udføres.

Linie 40 lader Dem indtaste de tal, som skal gennemsnitsberegnes.

Linie 50 adderer hvert tal til den løbende total.

Linie 60 giver computeren besked på at gå til linie 30, forøge tælleren (J) og begynde en ny løkke.

Linie 70 dividerer totalen med det antal tal, De indtastede (N) efter at løkken er blevet udført N gange.

Linie 80 giver en blank linie på skærmen.

Linie 90 udskriver meddelelsen, som giver Dem antallet af tal og deres total.

Linie 100 udskriver tallenes gennemsnit.

Linie 110 fortæller computeren, at programmet er slut.

READ-DATA KOMMANDOEN

Der findes en anden, endnu mere kraftfuld metode til at fortælle computeren, hvilke tal eller karakterer, den skal bruge i Deres program. De kan bruge READ instruktionen i programmet for at få computeren til at hente et tal eller en karakter (eller flere) fra DATA instruktionen. Hvis De eksempelvis ønsker, at computeren skal beregne gennemsnittet på fem tal, kan De bruge READ og DATA instruktionerne således:

```
10 T=0
20 FOR J=1 TO 5
30 READ X
40 T=T+X
50 NEXT
60 A=T/5
70 ? "GENNEMSNIT =";A
80 END
90 DATA 5,12,1,34,18
```

Når De kører programmet, vil computeren udskrive GENNEMSNIT = 14 på skærmen. Programmet anvender variablen T til optælling af en løbende total, og beregner gennemsnittet på samme måde som INPUT programmet. READ-DATA programmet finder imidlertid de tal, der skal beregnes på, i en DATA linie. Læg mærke til linie 30, READ X. Denne READ kommando fortæller computeren, at der findes en DATA instruktion i programmet. Den finder DATA linien og bruger det første tal som aktuel værdi for variablen X. Næste gang løkken udføres, vil det næste tal i DATA instruktionen blive brugt som værdi for variablen, o.s.v.

I en DATA instruktion kan De angive et hvilket som helst tal, men ikke en beregning. DATA instruktionen kan placeres hvor som helst i programmet selv efter END instruktionen. Det er fordi, computeren egentlig aldrig udfører DATA instruktionen; den refererer kun til den. Pas på at adskille Deres dataindivider med kommaer.

Hvis De har mere end een DATA instruktion i programmet, vil computeren referere til den, som følger nærmest efter den READ instruktion, som udføres netop nu. Computeren benytter en pointer (pegepind) for at huske sig selv på, hvilket nummer, den sidst læste. Når computeren har læst det første tal i DATA

instruktionen, peger pointeren på det andet tal. Når computeren atter kommer til READ instruktionen, tildeler den den værdi, pointeren indikerer, til variabelnavnet i READ instruktionen.

De kan bruge så mange READ og DATA instruktioner i Deres program, som De har brug for, men pas på, at der er data nok i DATA instruktionerne, som computeren kan læse. Prøv lige at fjerne et af tallene fra DATA instruktionen i det sidste program og kør det igen. Computeren svarer med meddelelsen ?OUT OF DATA ERROR IN 30. Der skete det, at da computeren udførte løkken femte gang, var der ikke flere data at læse. Det er hvad fejlmeldingen fortæller Dem. At der lægges for meget i DATA instruktionen, tager computeren overhovedet ikke notits af.

RESTORE KOMMANDOEN

De kan bruge RESTORE kommandoen i et program til at genstille datapointeren til det første dataindivid, hvis dette ønskes. Udskift linie 80 END i programmet med:

```
80 RESTORE
```

og tilføj

```
85 GOTO 10
```

Kør nu programmet. Programmet vil køre endeløst og bruge den samme DATA instruktion. (Obs! Hvis computeren giver en OUT OF DATA ERROR meddelelse er det fordi, De har glemt at genindsætte det tal, De før fjernede fra DATA instruktionen).

Som tidligere nævnt, kan De bruge DATA instruktioner til at tildele værdier til strengvariabler. Der gælder her samme regler som for numeriske data. Slet computerens hukommelse og indtast følgende program:

```
10 FOR J=1 TO 3
20 READ A$
30 ? A$
40 NEXT
50 END
60 DATA COMMODORE,C128,COMPUTER
```

Hvis READ instruktionen henter en strengvariabel, kan De placere bogstaver eller tal i DATA instruktionen. Læg imidlertid mærke til, at da computeren læser en streng, vil tal blive lagret som en karakterstreng, ikke som en værdi, der kan manipuleres med. Tal lagret som strenge kan udskrives, men ikke bruges i beregninger. Det er heller ikke muligt at angive bogstaver i en DATA instruktion, hvis READ instruktionen forventer en talvariabel.

BRUG AF OMRÅDER (ARRAYS)

De har nu set, hvorledes man ved brug af READ-DATA kan tildele en variabel mange værdier. Hvad nu, hvis De ønsker, at computeren skal kunne huske alle data i DATA instruktionen, i stedet for at ændre indholdet af en variabel? Hvis De for eksempel ønsker at kunne genkalde det tredie tal, eller den anden streng med karakterer?
Hver eneste gang De tildeler en variabel en ny værdi, sletter computeren den gamle værdi i variablens 'kasse' i hukommelsen og indsætter den nye værdi istedet. De kan give computeren besked på at reservere en række 'kasser' i hukommelsen og heri gemme enhver værdi, De i Deres program tildeler denne variabel. En sådan række af 'kasser' kaldes et array (område).

TABELOMRÅDER

Hvis området indeholder alle de værdier, variablen X fik tildelt i READDATA eksemplet, kaldes det X-området. Den første værdi X tildeles af programmet navngives X(1), den næste X(2) o.s.v. Dette kaldes tabelvariabler. Tallet i paranteserne kaldes tabelnumre. Som tabelnummer kan bruges en variabel eller en beregning. Her følger en ny version af gennemsnits-beregningsprogrammet, idet der nu anvendes tabelvariabler.

```
5 DIM X(5)
10 T=0
20 FOR J=1 TO 5
30 READ X(J)
40 T=T+X(J)
50 NEXT
60 A=T/5
70 ? "GENNEMSNIT =";A
80 END
90 DATA 5,12,1,34,18
```

Læg mærke til, at der ikke er mange ændringer. Linie 5 indeholder den eneste nye instruktion. Den fortæller computeren, at den skal oprette fem kasser til X området i hukommelsen. Linie 30 er ændret, så computeren, hver gang den udfører en løkke, tildeler en værdi fra DATA instruktionen til den plads i X området, som svarer til løkke tælleren (J). Linie 40 udfører det samme som før, men De må benytte en tabelvariabel for at gøre det.

Efter at De har kørt programmet, skal De, hvis De ønsker at genkalde det tredie tal, taste ?X(3) RETURN . Computeren husker alle tal i X området.

De kan oprette strengområder for at gemme strengvariablers karakterer på samme måde. Prøv at opdatere COMMODORE C128 COMPUTER READ-DATA programmet, så computeren vil huske disse elementer i A$ området.

```
5 DIM A$(3)
10 FOR J=1 TO 3
20 READ A$(J)
30 ? A$(J)
40 NEXT
50 END
60 DATA COMMODORE,C128,COMPUTER
```

TIP: De behøver ikke DIM instruktionen i Deres program, med mindre området, De bruger, har mere end 10 elementer. Se DIMENSIONERING AF OMRÅDER.

DIMENSIONERING AF OMRÅDER (ARRAYS)

Områder kan bruges sammen med kædede løkker, så computeren kan håndtere data på en mere avanceret måde. Forestil Dem, at De har et skema med 10 rækker og 5 tal i hver række. Tænk Dem, at De ønsker at finde gennemsnittet af de fem tal i hver række. De kunne oprette 10 områder og få computeren til at beregne gennemsnittet af de fem tal i hvert af disse. Dette er ikke nødvendigt. De kan anbringe alle tallene i et todimensionalt område.
Dette område vil have samme størrelse som skemaet, De ønsker at arbejde med - ti rækker med fem kolonner. DIM instruktionen for dette område (som vi kalder X) skal være:

```
10 DIM X(10,5)
```

Dette fortæller computeren, at den skal reservere plads i hukommelsen til et todimensionalt område kaldet X. Computeren reserverer plads til 50 tal.
Det er ikke nødvendigt at udfylde et område med så mange tal, som det er DIMensioneret til, men computeren vil stadig reservere plads til alle positioner i området.

PROGRAMEKSEMPEL

Det bliver nu meget let at referere til et hvilket som helst tal i skemaet ved blot at angive tallets kolonne og række position. Se nedenstående kort. Find det tredie element i den tiende række (1500). I Deres program henviser De til dette tal med X(10,3). Efterfølgende program indlæser tallene fra skemaet til et todimensionelt område (X) og beregner gennemsnittet af tallene i den enkelte række.

	Kolonne				
Række	**1**	**2**	**3**	**4**	**5**
1	**1**	**3**	**5**	**7**	**9**
2	**2**	**4**	**6**	**8**	**10**
3	**5**	**10**	**15**	**20**	**25**
4	**10**	**20**	**30**	**40**	**50**
5	**20**	**40**	**60**	**80**	**100**
6	**30**	**60**	**90**	**120**	**150**
7	**40**	**80**	**120**	**160**	**200**
8	**50**	**100**	**150**	**200**	**250**
9	**100**	**200**	**300**	**400**	**500**
10	**500**	**1000**	**1500**	**2000**	**2500**

```
10 DIM X(10,5),A(10)
20 FOR R=1 TO 10
30 T=0
35 FOR C = 1 TO 5
40 READ X(R,C)
50 T=T+X(R,C)
60 NEXT C
70 A(R)=T/5
80 NEXT R
90 FOR R=1 TO 10
100 PRINT "RÆKKE NR";R
110 FOR C=1 TO 5
120 PRINT X(R,C):NEXT C
130 PRINT "GENNEMSNIT = ";A(R)
140 FOR D=1 TO 1000:NEXT
150 NEXT R
160 DATA 1,3,5,7,9
170 DATA 2,4,6,8,10
180 DATA 5,10,15,20,25
190 DATA 10,20,30,40,50
200 DATA 20,40,60,80,100
210 DATA 30,60,90,120,150
220 DATA 40,80,120,160,200
230 DATA 50,100,150,200,250
240 DATA 100,200,300,400,500
250 DATA 500,1000,1500,2000,2500
260 END
```

SUBRUTINER

GOSUB-RETURN KOMMANDOEN

Den eneste måde, hvorpå De endnu har kunnet fortælle computeren, at den skulle gå til en anden del af programmet, har været ved hjælp af GOTO kommandoen. Hvad nu, hvis De, fra et bestemt sted i programmet, ønsker at computeren skal hoppe til en anden programdel, udføre instruktionerne heri og derefter returnere til det sted, den kom fra og fortsætte programudførelsen herfra.

Den programdel, som computeren således hopper til, kaldes en subrutine.
Slet computerens hukommelse og indtast efterfølgende program.

```
10 A$="SUBRUTINE":B$="PROGRAM"
20 FOR J=1 TO 5
30 INPUT "INDTAST ET TAL";X
40 GOSUB 100
50 PRINT B$:PRINT
60 NEXT
70 END
100 PRINT A$:PRINT
110 Z=X↑2:PRINT Z
120 RETURN
```

Dette program vil tage de tal, De indtaster, udregne kvadratet på dem, og udskrive resultatet. Den anden meddelelse fortæller Dem, hvornår computeren udfører enten subrutinen eller hovedprogrammet. Linie 40 får computeren til at hoppe til linie 100, udføre den og efterfølgende instruktioner indtil den møder en RETURN kommando. RETURN instruktionen giver computeren besked på at gå tilbage til den linie i programmet, der følger umiddelbart efter GOSUB kommandoen, og fortsætte derfra. Subrutiner kan anbringes hvor som helst i et program -- selv efter END instruktionen. Husk også, at GOSUB og RETURN kommandoerne altid skal bruges sammen i et program (som FOR og NEXT & IF og THEN), i modsat fald vil computeren give en fejlmelding.

ON GOTO/GOSUB KOMMANDOEN

Der findes en anden måde til at få computeren til at hoppe

til en anden del i programmet (dette kaldes branching - (forgrening)). Ved at bruge ON instruktionen, kan De få computeren til at afgøre, til hvilken del af programmet, der skal forgrenes, på basis af en kalkulation eller tastaturinput. ON instruktionen bruges sammen med enten GOTO eller GOSUB-RETURN kommandoerne, afhængig af hvad programmet skal udføre. En variabel eller kalkulation skal anføres efter ON kommandoen. Efter GOTO eller GOSUB kommandoen, skal der være en liste med linienumre. Indtast følgende program for at se, hvorledes ON kommandoen virker:

```
10 ? "INDTAST ET TAL MELLEM ET OG FEM"
20 INPUT X
30 ON X GOSUB 100,200,300,400,500
40 END
100 ? "DERES TAL VAR 1":RETURN
200 ? "DERES TAL VAR 2":RETURN
300 ? "DERES TAL VAR 3":RETURN
400 ? "DERES TAL VAR 4":RETURN
500 ? "DERES TAL VAR 5":RETURN
```

Hvis værdien af X er 1, forgrener computeren til første linienummer i listen. Hvis X er 2 forgrenes til andet nummer i listen o.s.v.

BRUG AF HUKOMMELSES ADRESSER

BRUG AF PEEK OG POKE FOR RAM/ROM TILGANG

Hvert område i computerens hukommelse har en speciel funktion. Der findes for eksempel et meget stort område, hvori Deres programmer og deres tilhørende variabler lagres. Det er denne del af hukommelsen som slettes, når De bruger NEW kommandoen. Andre områder er ikke så store, men har meget specielle funktioner. Der findes bl. a. et hukommelsesområde, som styrer computerens musikmuligheder.

Der findes to BASIC kommandoer - PEEK og POKE - som kan bruges for tilgang til og manipulation med computerens hukommelse. Denne anvendelse af PEEK og POKE kommandoerne kan være et meget betydende programmeringsværktøj, da indholdet af computerens hukommelsesområder afgør, nøjagtigt hvad computeren skal gøre på et bestemt tidspunkt.

BRUG AF PEEK

Med PEEK kommandoen kan man få computeren til at oplyse, hvilken værdi der er lagret i en bestemt adresse i hukommelsen (værdien i en adresse kan være en værdi fra 0 til 255). PEEK kan bruges for at få værdien af enhver adresse i hukommelsen (RAM el. ROM) i DIREKTE eller PROGRAM mode. Indtast:

P = PEEK(2594) RETURN
? P RETURN

Når De trykker på RETURN efter første linie, tildeles variablen P den værdi der findes i hukommelsesadressen 2594. Den udskriver denne værdi, når De trykker på RETURN efter at have indtastet ? P kommandoen. Adresse 2594 afgør, hvorvidt taster som mellemrum og CRSR repeteres, når de holdes nedtrykkede. Tallet 128 i denne adresse betyder, at tasterne er repeterende, når de holdes nede. Prøv at holde mellemrumstangenten nedtrykket og se, hvordan markøren flytter sig på tværs af skærmen.

BRUG AF POKE

POKE kommandoen bruges til at ændre værdien i en RAM adresse. Tast:

POKE 2594,96 RETURN

Computeren gemmer værdien efter kommaet (96) i den adresse, der er angivet foran kommaet (2594). Værdien 96 i adressen 2594 fortæller computeren, at taster som mellemrum og CRSR ikke skal være repeterende. Prøv nu at holde mellemrumstangenten nede og se på markøren - der sker næsten intet! Markøren flyttes een plads mod højre, men den repeterer ikke. For at stille computeren tilbage til dens normale tilstand, tastes:

POKE 2594,128 RETURN

De kan ikke ændre værdierne i samtlige adresser - værdierne i ROM kan læses, men ikke ændres.

OBS! Disse eksempler forudsætter at De er i bank 0. Se beskrivelsen af BANK kommandoen i kapitel 5, BASIC 7.0 for detaljer.

BASIC FUNKTIONER

HVAD ER EN FUNKTION

En funktion er en forud bestemt operation i BASIC sproget, som generelt forsyner Dem med en enkelt værdi. Når funktionen giver værdien, siges det, at den returnerer den. For eksempel er kvadratrodsfunktionen SQR en matematisk funktion, som returnerer værdien af et angivet tal, når dette tal opløftes til anden potens.

Der findes to slags funktioner:

NUMERISKE - returnerer et enkelt tal som resultat. Numeriske funktioner rækker fra at beregne matematiske værdier til at specificere den numeriske værdi i en hukommelses adresse.
STRENG - returnerer et resultat, som er en karakter.

Følgende indeholder beskrivelse af nogle af de mest anvendte funktioner.
En komplet liste over funktioner kan findes i kapitel 5, BASIC 7.0.

HELTALSFUNKTIONEN (INT)

Hvad nu, hvis De ønsker at få afrundet et tal til det nærmeste heltal? For det første vil De skulle benytte heltalsfunktionen INT (Integer=heltal).
Denne funktion fjerner alt, hvad der findes efter decimalpunktet. Prøv at indtaste disse eksempler:

? INT(4.25) RETURN
? INT(4.75) RETURN
? INT(SQR(50)) RETURN

Hvis De ønsker oprunding til det nærmeste heltal, skulle det andet eksempel give værdien 5. I virkeligheden bør De oprunde ethvert tal med en decimalværdi over 0.5. For at gøre dette, må De lægge 0.5 til tallet før De bruger INT funktionen. På denne måde vil tal med decimaler over 0.5 øges med 1, før de nedrundes med INT funktionen. Prøv til en begyndelse dette:

? INT(4.75+0.5) RETURN

Før INT funktionen blev udført, adderede computeren 0.5 til

4.75, så den derefter nedrundede 5.25 til 5 som resultat. Hvis De ønsker at få afrundet en kalkulations resultat, gøres det således:

? INT((100/6)+.05) RETURN

Divisionen i eksemplets inderste paranteser kan erstattes af enhver anden kalkulation.

Hvad nu, hvis De ønsker at afrunde tal til nærmeste 0.01 -- som kroner og ører. Først må De, i stedet for at addere 0.5 til Deres tal, addere 0.005 og derefter multiplicere med 100. Lad os sige, at De vil afrunde 2.876 til nærmeste 0.01. Brug denne metode, begynd med:

? (2.876 + 0.005)*100 RETURN

Brug nu INT funktionen for at slippe af med alt efter decimalpunktet (som jo er flyttet to pladser mod højre, da der blev ganget med 100). Tilbage har De nu:

? INT ((2.2876 + 0.005)*100) RETURN

som giver Dem en værdi på 288. Alt, hvad der nu mangler, er at dividere med 100 for at få værdien 2,88, som er det ønskede resultat. Ved anvendelse af denne teknik, kan kalkulationer som den følgende afrundes til nærmeste 0.01:

? INT (((2.876+1.29+16.1-9.534)+0.005)
***100)/100** RETURN

FREMSTILLING AF TILFÆLDIGE (RANDOM) TAL

Der er endnu en funktion, De skal lære, før De begynder at udvikle Deres egne programmer, eller begynder at bruge de koncepter om grafik og musik, som gennemgås i de to sidste afsnit. RND funktionen giver computeren besked på at generere et tilfældigt tal. Alle genererede tal er nicifrede tal i decimal form og ligger mellem 0.000000001 og 0.999999999. Tast:

? RND(0) RETURN

Ved at multiplicere det tilfældigt genererede tal med 6 opnås, at de genererede tals område udvides til større end 0 og mindre end 6. For at få 6 med blandt de genererede tal, lægger vi 1 til resultatet af RND(0)*6.
Dette giver området 1⟨X⟨7. Hvis vi anvender INT funktionen

for at eliminere decimalpladserne, vil kommandoen generere et heltal fra 1 til 6. Denne fremgangsmåde kan benyttes for at få computeren til at fremstille tal mellem 1 og 6, som en slags terningkast. Indtast:

```
10 R = INT(RND(1)*6+1)
20 ? R
30 GOTO 10
```

Hvert genereret tal svarer til et terningkast. For at simulere kast af to terninger, kan to af denne slags kommandoer bruges. I beskrivelsen af DO/LOOP instruktionen i afsnit 5 fortælles om en anden måde at generere tilfældige tal på.

ASC OG CHR$ FUNKTIONERNE

Hver karakter, som Commodore 128 kan udskrive (incl. grafiske tegn) har et bestemt nummer. Dette tal kaldes en karakterstrengs-kode (CHR$) og Commodore 128 har 255 af disse. Sammenhængende hermed findes to meget nyttige funktioner. Den første er ASC funktionen. Indtast:

? ASC("Q") RETURN

Computerens svar er 81. 81 er karakterstrengs-koden for Q tasten. Udskift Q i ovenstående kommando med èn hvilken som helst anden tast for at se dennes kodetal.

Den anden funktion er CHR$ funktionen. Tast:

? CHR$(81) RETURN

Læg mærke til, at computeren nu svarer med et Q. I virkeligheden er CHR$ funktionen det modsatte af ASC funktionen. Begge refererer til tabellen over karakterstreng-koder i computerens hukommelse. CHR$ værdier kan bruges til at programmere funktionstaster med. Se appendiks E for fuld oversigt over ASC og CHR$ koder. Afsnit 5 giver mere information om denne anvendelse af CHR$.

KONVERTERING AF STRENGE OG TAL

De kan af og til have behov for at udføre beregninger på numeriske karakterer, der er lagret som strengvariabler i Deres program. Andre gange kan De måske ønske at udføre strengoperationer på tal. Der findes to BASIC funktioner, som De kan bruge

for at konvertere variabler fra numerisk til strengtype og vice versa.

VAL Funktionen

VAL funktionen giver den numeriske værdi af en strengvariabel. Slet computerens hukommelse og indtast dette program:

```
10 A$ = "64"
20 A = VAL(A$)
30 ? "VÆRDIEN AF";A$"" ER";A
40 END
```

STR$ Funktionen

STR$ funktionen konverterer numerisk variable til strengvariabler. Slet computerens hukommelse og indtast dette program:

```
10 A = 65
20 A$ = STR$(A)
30 ? A"ER VÆRDIEN AF";A$
```

KVADRATRODS FUNKTIONEN

Kvadratrodsfunktionens navn er SQR. For for eksempel at finde kvadratroden af tallet 50, tastes:

```
? SQR(50) RETURN
```

På denne måde kan roden af ethvert positivt tal uddrages.

ABSOLUT VÆRDI FUNKTIONEN

Absolut værdi funktionen (ABS) er meget nyttig i forbindelse med behandling af negative tal. De kan bruge denne funktion for at få den positive værdi af ethvert tal -- positivt eller negativt. Prøv disse eksempler:

```
? ABS (-10) RETURN
? ABS (5)" ER LIG MED " ABS (-5) RETURN
```

STOP OG CONTINUE KOMMANDOERNE

Der findes en anden måde, hvorpå De kan arbejde interaktivt med Deres computer. De kan få den til at stoppe et program

og derefter til at fortsætte, når De er klar til det. Programmet skal så indeholde en STOP kommando. STOP instruktionen kan placeres et vilkårligt sted i programmet.

Når computeren afbryder programmet, kan De ved brug af DIRECT mode kommandoer finde ud af, nøjagtigt hvad der sker i programmet (d.v.s. at De kan finde værdierne af løkke-tælleren eller andre variabler). Dette er meget nyttigt hvis De foretager fejlsøgning eller korrektion af Deres program. Slet nu computerens hukommelsen og indtast følgende program:

```
10 X=INT(SQR(630))
20 Y=(.025*80)↑2
30 Z=INT(X*Y)
40 STOP
50 FOR J=0 TO Z STEP Y
60 ? "STOP OG FORTSÆT"
70 NEXT
80 END
```

Kør nu programmet med RUN. Computeren svarer med "BREAK IN 40". På dette tidspunkt har computeren beregnet værdierne af X, Y og Z. Hvis De ønsker at finde ud af, hvad resten af programmet forventes at udføre, kan De bede computeren om at udskrive (PRINTe) X;Y;Z. De vil ofte, når De foretager fejlsøgning eller korrektion i et stort program (eller et lille komplekst) ønske at kende værdier af en variabel på et bestemt sted i programmet.

Når De har den ønskede oplysning, kan De taste CONT (for CONTINUE) og trykke på RETURN. Computeren vil så fortsætte programmet, idet den begynder med instruktionen, som følger efter STOP kommandoen.

****

Meningen med dette og foregående afsnit har været at gøre Dem bekendt med BASIC programmeringssproget og dets muligheder. De sidste fire afsnit i dette kapitel beskriver kommandoer, som kun kan benyttes i C128 mode.
Visse Commodore 128 mode kommandoer tilføjer muligheder, som ikke findes i C64 mode. Andre C128 mode kommandoer udfører det samme som en C64 kommando, men på en enklere måde. Syntaksen for alle Commodore 7.0 kommandoerne vises i BASIC 7.0 ordbogen bagest i denne bog.

AFSNIT 5

NOGLE BASIC KOMMANDOER OG TASTATUROPERATIONER UDELUKKENDE FOR C128 MODE

INTRODUKTION

Dette afsnit introducerer nogle kraftfulde BASIC kommandoer og instruktioner, som De formentlig aldrig før har set, selv om De skulle være en øvet BASIC programmør. Hvis De er fortrolig med BASIC programmering, har De formentlig oplevet mange situationer, hvor De kunne have haft brug for disse kommandoer og instruktioner. Dette afsnit beskriver konceptet vedr. den enkelte kommando og giver eksempler på, hvorledes de bruges i et program. Fuldstændig liste over og forklaringer til de enkelte kommandoer og instruktioner findes i BASIC 7.0 ordbogen bagest i denne vejledning.
Dette afsnit beskriver endvidere anvendelsen af de specialtaster, som kan bruges i C128 mode.

AVANCEREDE LØKKER

DO/LOOP INSTRUKTIONEN

DO/LOOP instruktionen giver flere avancerede måder til at fremstille løkker, end det er tilfældet med GOTO, GOSUB eller FOR/NEXT instruktionerne.
DO/LOOP instruktionen bibringer BASIC sproget en meget kraftfuld og specialiseret teknik, som normalt kun findes i strukturerede programmeringssprog. I denne forklaring vil vi blot gennemgå nogle få muligheder for anvendelse af DO/LOOP.

Hvis De ønsker at fremstille en uendelig løkke, begynder De med en DO instruktion, indtaster derefter den eller de linie(r), som specificerer, hvad computeren skal udføre, og afslutter med en LOOP instruktion, således:

```
100 DO
110 PRINT "GENTAGELSE"
120 LOOP
```

Nedtrykning af RUN/STOP tasten afbryder programmet.

Det, som følger efter DO instruktionen, udføres indtil programmet når til LOOP instruktionen (linie 120); derfra gås tilbage til DO instruktionen i linie 100. Alle instruktioner, som måtte findes mellem DO og LOOP vil blive udført i det uendelige.

UNTIL

En endnu nyttigere teknik er at kombinere DO/LOOP med UNTIL instruktionen.

UNTIL instruktionen stiller en betingelse, som styrer løkken. Løkken vil fungere uendeligt, med mindre UNTIL's betingelse opfyldes.

```
100 DO: INPUT "KAN DE LIDE DERES
    COMPUTER?"+;A$
110 LOOP UNTIL A$="JA"
120 PRINT "DET ER JEG GLAD FOR"
```

DO/LOOP instruktionen bruges ofte for at gentage en hel rutine uendeligt indenfor et program, som i det følgende:

```
10 PRINT "PROGRAMMET FORTSÆTTER TIL DE
   TASTER 'SLUT'"
20 DO UNTIL A$="SLUT"
30 INPUT "GRADER FAHRENHEIT";F
40 C=(5/9)*(F-32)
50 PRINT F;" GRADER FAHRENHEIT ER DET SAMME
   SOM ";C;" GRADER CELSIUS"
60 INPUT "IGEN ELLER SLUT";A$
70 LOOP
80 END
```

DO/LOOP kan også bruges som en tæller, hvor UNTIL instruktionen bruges for at angive et bestemt antal gentagelser.

```
10 N=2*2
20 PRINT "TO FORDOBLEDE ENS TAL"; N
30 DO UNTIL X=25
40 X=X+1
50 N=N*2
60 PRINT "FORDOBLET";X+1;"GANGE...";N
70 LOOP
80 END
```

Læg mærke til, at hvis De udelader tællerinstruktionen (UNTIL X=25 delen i linie 30) vil tallet fordobles i det uendelige, til en OVERFLOW fejl fremkommer.

WHILE

WHILE instruktionen virker på samme måde, men løkken gentages kun, hvis betingelsen er opfyldt, som i denne rettede udgave af det sidste korte program:

```
100 DO: INPUT "KAN DE LIDE DERES
    COMPUTER?"+;A$
110 LOOP WHILE A$ <> "JA"
120 PRINT "DET ER JEG GLAD FOR"
```

EXIT

EXIT instruktionen kan anbringes indenfor en DO/LOOP sætning. Når en EXIT instruktion opdages, hopper programmet til næste instruktion, som følger efter LOOP instruktionen.

ELSE klausulen med IF-THEN

ELSE klausulen giver mulighed for at give computeren besked på, hvorledes den skal reagere, hvis vilkåret i IF-THEN instruktionen er falsk. I stedet for at fortsætte til næste programlinie, vil computeren udføre kommandoen eller forgrene til den programlinie, som angives i ELSE klausulen. Hvis De for eksempel ønsker, at computeren skal udskrive kvadradet på et tal, kan De bruge ELSE på følgende måde:

```
10 INPUT "INDTAST ET TAL TIL OPLØFTNING";N
20 IF N <100 THEN PRINT N*N:ELSE 40
30 END
40 ? "TALLET SKAL VÆRE <100":GOTO 10
```

Læg mærke til at De skal anføre et kolon mellem IF-THEN instruktionen og ELSE klausulen.

BEGIN/BEND sekvensen med IF-THEN

BASIC 7.0 tillader, at De kan udnytte IF-THEN vilkåret endnu bedre.
BEGIN/BEND sekvensen tillader Dem at medtage et antal programlinier, som vil blive udført, hvis IF vilkåret er sandt, i stedet for en enkel funktion eller GOTO. Kommandoen er konstrueret således:

```
IF vilkår THEN BEGIN:
(programlinier):
BEND:ELSE
```

Vær opmærksom på, at der skal være et kolon mellem BEGIN og instruktioner til computeren, og igen mellem sekvensens sidste kommando og ordet BEND.

BEGIN/BEND kan benyttes uden en ELSE klausul eller kan bruges efter ELSE klausulen, hvis THEN kun efterfølges af en enkelt kommando. Prøv dette:

```
10 INPUT A
20 IF A< 100 THEN BEGIN: ? "DERES TAL VAR";A
30 SLEEP 2:REM FORSINKELSE
40 FOR X = 1 TO A
50 ? "DETTE ER ET EKSEMPEL PÅ BEGIN/BEND"
60 NEXT X
70 ? "DET ER NOK":BEND:ELSE ?"FOR MANGE"
80 END
```

Dette program spørger efter et tal fra brugeren. Hvis tallet er mindre end 100, udføres instruktionerne mellem nøgleordene BEGIN og BEND, sammen med eventuelle instruktioner på samme linie som BEND (bortset fra ELSE). Meddelelsen "DERES TAL VAR N" viser sig på skærmen. Linie 30 er en forsinkelsesløkke, som bruges til at fastholde meddelelsen på skærmen så længe, at den er let at læse. Derefter bruges en FOR/NEXT løkke til at vise en meddelelse med brugerens angivne tal. Er tallet større end 100, ses der bort fra THEN vilkåret og ELSE vilkåret (udskrivning af "FOR MANGE") udføres.
ELSE nøgleordet skal være på samme linie som BEND.

SLEEP KOMMANDOEN

Læg mærke til SLEEP kommandoen i linie 30 i sidste program. SLEEP giver en lettere, mere nøjagtig måde til at indsætte og fastlægge en tidsforsinkelse i programforløbet. SLEEP kommandoens format er:

SLEEP n

hvor n angiver antallet af sekunder, i området fra 0 til 65535, der ønskes som forsinkelse i programmet. Kommandoen i linie 30 giver en forsinkelse på 2 sekunder.

FORMATTERING AF UDDATA

PRINT USING KOMMANDOEN

Forestil Dem, at De skriver et salgsprogram, som beregner et beløb i kroner. Totalsalget divideret med antallet af salgsfolk giver gennemsnitssalget. Men en sådan beregning kunne resul-

tere i et beløb indeholdende fire eller fem decimalpladser! De kan formattere dette resultat, så kun to decimalpladser bliver vist. Kommandoen, som bruges hertil, hedder PRINT USING.

PRINT USING lader Dem danne et format for Deres uddata, idet mellemrum, kommaer, decimalpunkter og dollartegn kan anvendes. Nummertegn (#) bruges til at repræsentere mellemrum eller karakterer i det viste resultat. For eksempel:

PRINT USING "#######.##";A

instruerer computeren om at udskrive værdien af A i det specificerede format, med to pladser for øre og fem pladser for kroner. Hvis De ønsker et komma foran de sidste tre kronepladser, som i kr. 1,000.00 skal der medtages et komma i PRINT USING instruktionen. Husk, at De kan formattere uddata med mellemrum, kommaer, decimalpunkter og dollartegn. Der kan bruges flere andre specielle karakterer med PRINT USING, se BASIC ordlisten for mere information.

PUDEF KOMMANDOEN

Hvad nu, hvis det ikke drejer sig om kroner og øre ? Der findes en kommando, som ligner PRINT USING, som sætter Dem i stand til at redefinere de tegn, som typisk indeholdes i PRINT USING formatet. Det drejer sig om PUDEF instruktionen, som tillader Dem at ændre de karakterer, som er defineret i PRINT USING -- kommaer, mellemrum, decimaltegn, dollartegn -- til andre karakterer på tastaturet.

PUDEF kommandoen har fire positioner, men det er ikke nødvendigt at omdefinere alle fire. Kommandoen ser således ud:

PUDEF " , . $"
1 2 3 4

- position 1 er fyldkarakteren. En blank vil vises, hvis positionen ikke omdefineres.
- position 2 er kommakarakteren. Standardværdi er komma.
- position 3 er decimalpunktet. Standardværdi er punktum.
- position 4 er dollartegnet

Hvis De vil skrive et program, hvor De i stedet for dollartegn f. eks. vil have @-tegnet, kan det gøres således:

```
10 PUDEF " @"
20 PRINT USING "#$#####.##";X
```

PROGRAMEKSEMPEL

Dette program bruger nogle af de nye kommandoer og instruktioner i Commodore 7.0 BASIC til at beregne rente og låneafdrag. Det definerer en minimumsværdi for lånet ved brug af ELSE sammen med IF/THEN instruktionen, og et krone/øre format for tallene med PRINT USING instruktionen.

```
10 INPUT "LÅNEBELØB I KRONER";A
20 IF A<100 THEN 70: ELSE P=.15
30 I=A*P
40 ?"TOTALYDELSE UDGØR";
50 PRINT USING "#######.##"; A+I
60 GOTO 80
70 ?"DER YDES IKKE LÅN UNDER 100 KR."
80 END
```

GETKEY

De har lært, hvorledes man bruger INPUT og GET kommandoerne for indtastning af DATA i et programforløb. En anden måde, hvorpå der kan indlæses data, mens et program udføres, er ved brug af GETKEY instruktionen. GETKEY instruktionen accepterer kun een tast ad gangen. GETKEY efterfølges af en strengvariabel (for eksempel A$). Enhver tast, som bliver nedtrykket, bliver tildelt til denne strengvariabel. GETKEY er værdifuld, fordi den sætter Dem i stand til at indtaste data med eet tegn ad gangen, uden at skulle trykke på RETURN efter hvert tegn. GETKEY instruktionen kan kun bruges i programmer.

Her er et eksempel på anvendelse af GETKEY i et program:

```
1000 PRINT "VÆLG A, B, C, D, E ELLER F"
1010 GETKEY A$
1020 PRINT A$;"BLEV NEDTRYKKET."
```

Computeren venter til en tast nedtrykkes; når det sker, bliver tastens værdi tildelt variablen A$ og udskrevet i linie 1020. Følgende program viser GETKEY i flere komplekse og nyttige versioner; som svar på et spørgsmål med flere svarmuligheder og

endvidere med spørgsmål på, om spørgsmålet skal gentages. Hvis det afgivne svar er forkert, har brugeren mulighed for at prøve igen ved at nedtrykke "J" tasten (linie 80). Den tast som nedtrykkes som svar, ud fra flere muligheder, tildeles variablen A$, medens "PRØV IGEN" tildeles B$, ved anvendelse af GETKEY instruktionerne i linierne 60 og 90. IF/THEN instruktioner bruges til løkker i programmet for at få passende reaktioner fra computeren på forskelligt input fra tastaturet.

```
10 PRINT "HVEM SKREV 'RAVNEN'?"
20 PRINT "A. EDGAR ELLEN POE"
30 PRINT "B. EDGAR ALLEN POE"
40 PRINT "C. IGOR ALLEN POE"
50 PRINT "D. ROB. RAVEN"
60 GETKEY A$
70 IF A$="B" THEN 150
80 PRINT "FORKERT. VIL DE PRØVE IGEN? (J ELLER N)"
90 GETKEY B$
100 IF B$="J" THEN PRINT "A,B,C ELLER D?":GOTO 60
110 IF B$="N" THEN 140
120 PRINT "TAST 'J' ELLER 'N' - PRØV IGEN"
130 GOTO 90
140 PRINT "DET RIGTIGE SVAR ER B"
145 GOTO 160
150 PRINT "RIGTIGT!!"
160 END
```

GETKEY ligner GET meget, bortset fra at GETKEY automatisk vil vente på, at en tast bliver nedtrykket.

PROGRAMMERINGS VÆRKTØJER

I de foregående afsnit har De lært at foretage ændringer i Deres programmer og at rette skrivefejl med INST/DEL. BASIC indeholder andre kommandoer og funktioner, som hjælper Dem med at lokalisere aktuelle programfejl, og kommandoer, som De kan bruge til at få programmeringen til at gå lettere.

Indtastning af programmer

AUTO

C128 BASIC indeholder en facilitet for automatisk nummerering. De bestemmer intervallet mellem linienumrene. Lad os sige, at De vil nummere Deres program på den mest anvendte måde,

med intervaller på 10. Før De begynder at programmere, indtaster De i DIREKTE mode:

AUTO 10 RETURN

Computeren vil automatisk nummerere Deres program med 10 som nummerinterval. Når De trykker på RETURN, viser næste linienummer sig og markøren er korrekt placeret for indtastning af næste instruktion. De kan vælge nummerering med et hvilket som helst interval; De kan vælge 5 eller selv 50.
Angiv blot tallet efter ordet AUTO og tryk på RETURN. Autonummerering afbrydes ved at taste AUTO uden talangivelse, og trykke på RETURN.

RENUMBER

Hvis De skriver et program og senere tilføjer instruktioner, kan det nogle gange give problemer med nummereringen. Med RENUMBER kommandoen kan De ændre linienumrene til et lige interval for en del af, eller hele programmet. RENUMBER kommandoen har flere mulige parametre, som vist nedenfor i paranteser:

RENUMBER ((ny startlinie)(,(forøgelse)(,gammel startlinie)))

Den nye startlinie angiver det tal, som den første programlinie vil have efter RENUMBER kommandoen er brugt. Hvis intet angives, bruges standardværdien 10. Forøgelse er springet mellem linienumrene og også her er standardværdien 10. Den gamle startlinie er det tal, hvorfra omnummerering skal ske. Dette gør det muligt, at omnummerere en del af et program, i stedet for hele programmet. For eksempel fortæller

RENUMBER 40,,80

computeren, at programmet skal omnummereres med start i linie 80, med en forøgelse på 10. Linie 80 bliver til linie 40.
Læg mærke til, at denne kommando, som AUTO, kun kan bruges i DIREKTE mode.

DELETE

De ved, hvorledes programlinier kan slettes ved at indtaste linienummeret og derefter trykke på RETURN. Dette kan være besværligt, hvis De ønsker at slette en programsektion. DELETE

kommandoen kan spare tid, da De kan angive et antal programlinier, som skal slettes samtidig. For eksempel:

DELETE 10-50

Sletter linie 10, 50 og alle mellemliggende linier. Brugen af DELETE svarer til brugen af LIST, ved at De kan specificere et område op til en given linie, eller blot en enkelt linie, som i disse eksempler:

DELETE -120 sletter alle linier t.o.m. linie 120
DELETE 120- sletter linie 120 og alt herefter
DELETE 120 sletter kun linie 120

LØSNING AF PROBLEMER VED PROGRAMMERING

Når et program ikke virker som forventet, vises en fejlmelding. Imidlertid giver fejlmeldingen kun en svag ide om, hvad der er galt, og De forstår stadig ikke problemet. Commodore 128 har mange måder, hvorpå den kan hjælpe Dem med at løse problemet.

HELP

Commodore 128 har en HELP kommando, som angiver, i hvilken linie et problem findes. For at aktivere HELP kommandoen skal De blot nedtrykke den specielle HELP tast på tastaturet.

Indtast følgende instruktion. Den indeholder en bevidst fejl, så tast den blot som den er:

10 ? 3;4:5;6

Hvis De kører dette eenlinies program med RUN, skriver computeren 3 og 4 som forventet, men svarer derefter "SYNTAX ERROR IN 10". Lad os lade, som om De ikke kan få øje på fejlen (et kolon i stedet for et semikolon mellem 4 og 5). Tryk på HELP. (De kan også taste HELP RETURN .) Computeren viser linien igen, men 5;6 er oplyst for at vise, hvor fejlen i linien er.

FEJLSØGNING - MED TRAP (fælde) KOMMANDOEN

Hvis en fejl optræder i et program, vil programmet normalt standse. På dette tidspunkt kan De trykke på HELP tasten for at finde fejlen. De kan imidlertid bruge BASIC 7.0 TRAP kommandoen for at indbygge en fejlsøgningsrutine i Deres program.

TRAP kommandoen adviserer Dem om at finde og rette en fejl, og fortsætter derefter programudførelsen. Fælden (trap) sættes typisk i første linie:

5 TRAP 100

fortæller cmputeren, at hvis en fejl optræder, skal den gå til en bestemt programlinie (i dette tilfælde linie 100). Linie 100 anføres i slutningen af programmet og opstiller her en konsekvens. Ingen linier udføres, MED MINDRE der er en fejl. Hvis en fejl optræder, bliver linien med TRAP instruktionen virksom, og styringen dirigeres til en andel del af programmet. De kan bruge disse instruktioner til at fange uforudsete fejl under inddatering af data, gentage udførelsen, eller returnere til tekstmode fra grafikmode, blot for at nævne nogle muligheder. Hvis De kører det sidste DO/LOOP eksempel uden UNTIL instruktion, kan De få en OVERFLOW fejl og programmet 'går ned'. Ved tilføjelse af to linier kan De undgå dette. Den ene linie skal stå i programmets begyndelse og den anden i slutningen. For dette eksempel skal De tilføje følgende to linier:

```
5 TRAP 100
100 IF N>1 THEN END
```

Selv om N er blevet meget større end een igennem programforløbet, vil instruktionen ikke blive brugt, før en fejl viser sig. Når tallet giver "overflow" (d.v.s. bliver større end computeren kan klare), træder TRAP instruktionen i funktion. Da N nu naturligvis er større end een, styres programmet til END (i stedet for at 'gå ned'). Linie 100 kunne have fået computeren til at udføre mange andre ting ved brug af andre mulige parametre.

Herunder vises et eksempel i hvilket fejlsøgning bruges for at undgå, at der bliver divideret med 0.

```
10 TRAP 1000
100 INPUT "JEG KAN DIVIDERE MED ETHVERT TAL.
ANGIV MIG ET TAL SOM SKAL DIVIDERES";D
110 INPUT "HVAD SKAL JEG DIVIDERE MED?";B
120 A=D/B
130 PRINT D;"DIVIDERET MED";B;"GIVER";A
140 END
1000 IF B=0 THEN PRINT "DET KAN JEG IKKE"
1100 INPUT "TAST ET ANDET TAL";B:RESUME 120
```

Læg mærke til RESUME i linie 1100. Dette fortæller compute-

ren, at den skal gå tilbage til den angivne linie (i dette tilfælde 120) og fortsætte. Afhængig af den fejl, der blev opdaget, kan fortsættelse være mulig eller umulig.
For yderligere oplysninger om fejlsøgning bedes De se funktionerne ERR$, EL og ER, i kapitel 5, Basic 7.0 ordforklaring.

GENNEMGANG AF PROGRAMMER - TRON OG TROFF KOMMANDOERNE

Når et problem viser sig i et program, eller De ikke får det forventede resultat, kan det være nyttigt at arbejde sig metodisk gennem programmet og gøre nøjagtigt, som computeren ville gøre. Denne proces kaldes 'tracing'. Tegn variabelkasser og opdater værdierne i relation til programinstruktionerne. Udfør beregningerne og udskriv resultaterne efter hver instruktion.

Tracing kan for eksempel vise Dem, at De har brugt GOTO med en forkert linienummerangivelse, eller beregnet et resultat, som aldrig er blevet gemt i en variabel. Mange programfejl kan lokaliseres ved at man forestiller sig, at man selv er computeren og kun udføre een instruktion ad gangen. Deres C128 kan udføre en slags trace ved brug af kommandoerne TRON og TROFF (TRace ON og TRace OFF). Mens programmer udføres, med TRACE ON, vil computeren udskrive linienumrene i den rækkefølge, de bliver udført. De vil på denne måde være i stand til at se, hvorfor Deres program ikke giver de forventede resultater.

Indtast et eller andet kort program af dem, vi tidligere har brugt, eller lav selv et. For at aktivere trace mode, indtastes TRON i DIREKTE mode. Når De kører programmet, læg så mærke til, hvorledes linienumre vises i paranteser før resultater vises på skærmen. Prøv at følge linienumrene og se, hvor mange trin det er nødvendigt for computeren at gå for at komme til et bestemt sted. TRON vil være mere interessant, hvis De vælger et program med mange forgreninger, som f. eks. GOTO, GOSUB og IF-THEN linienumre. Tast TROFF for at afbryde trace mode, før De fortsætter.

De behøver ikke at 'trace' et helt program. De kan indlægge TRON i et program, som en linie prioriteret til det programafsnit, som giver problemer.Indsæt ordet TROFF som en programlinie efter den besværlige sektion. Når De kører programmet, vil kun linierne mellem TRON og TROFF blive anført før resultaterne.

SKÆRMVINDUER

Vinduer er et specificeret skærmområde, som De definerer som Deres arbejdsareal. Alt hvad De indtaster (linier, programlister etc.) vil være indenfor for det angivne vindues rammer og vil ikke influere på det, som ligger udenfor. Commodore 128 tilbyder to måder, hvorpå vinduer kan dannes: WINDOW kommandoen og ESCAPE tast funktionen.

Commodore 128 BASIC indeholder en kommando, som kan spare Dem tid ved angivelse af vinduer, WINDOW kommandoen. Det kommando-format, som skal bruges, ser således ud:

WINDOW øverste venstre kolonne, øverste venstre række, nederste højre kolonne, nederste højre række, (CLR option)

De to første tal efter WINDOW specificerer kolonne- og rækkenumrene fra hvor, De ønsker, at vinduets øverste venstre hjørne skal være. De to næste tal angiver koordinaterne for nederste højre hjørne. Husk at skærmbilledets størrelse (40 eller 80 kolonner) er bestemmende for vinduets størrelse. De kan også indsætte en CLEAR option med denne kommando. Tilføjes et 1-tal i kommandoen slutning, slettes skærmbilledet, som i dette eksempel:

WINDOW 10,10,20,20,1

Her følger et programeksempel, der danner fire vinduer på skærmen, enten i 40 eller 80 kolonners format.

```
10 PRINT "!SHIFT/CLR!":REM  SLETTER SKÆRMEN
20 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
30 B$=A$+A$+A$
40 FOR I = 1 TO 25  : PRINT B$:NEXT : REM UDFYLDER SKÆRM MED TEGN
50 WINDOW 1 ,1 ,8 ,20 : REM DEFINER VINDUE1
60 PRINT "!RED!!RVS ON!";
65 FOR I = 1 TO 25:PRINT"         ";:NEXT
70 REM FOREGÅENDE LINIE UDFYLDER VINDUE1 MED RØDT
80 WINDOW 15,15,39,20,1 :REM DEFINER VINDUE2
90 PRINT "!GRN!"; B$;A$     :REM UDFYLDER VINDUE MED KARAKTERER
100 WINDOW 30,1,39,22,1 :REM DEFINER VINDUE3
110 PRINT "!BLU!": LIST     :REM VÆLGER GRØN OG LIST I VINDUE
120 WINDOW 5,5,33,18,1  :REM DEFINER VINDUE 4 OVENFOR DE TRE ANDRE
130 PRINT "!YEL!":PRINTA$:LIST: REM FARVESKIFT - A$ OG LIST I VINDUE
```

I PRINT-sætningerne er der i anførelsestegn anført nogle tekster, f.eks. som i linie 10,!SHIFT/CLR!. Dette skal opfattes på den måde, at de angivne taster skal nedtrykkes. Teksten skal altså ikke skrives.

OPRETTELSE AF VINDUE MED ESC TASTEN

For at fastlægge et vindue, følges denne fremgangsmåde:

1) Flyt markøren til den skærmposition, hvor De ønsker at vinduets øverste venstre hjørne skal være.
2) Tryk på ESC tasten og slip den igen. Tryk derefter på T.
3) Flyt markøren til den position, hvor De ønsker, at vinduets nederste højre hjørne skal være.
4) Tryk på ESC, slip den og tryk så på B. Deres vindue er fastlagt.

Ved brug af ESC tasten kan De manipulere med vinduet og teksten i det. Skærmredigerings funktioner, som indsættelse og fjernelse af tekst, scrolling og ændring af vinduets størrelse, kan udføres ved at nedtrykke ESC efterfulgt af tryk på en anden tast. For at få en speciel funktion, nedtrykkes den aktuelle tast efter nedtrykning af ESC:

TAST	FUNKTION
A	Automatisk indsættelses mode
B	Placerer vinduets nederste højre hjørne i markørens nuværende position.
C	Nulstil automatisk indsættelse
D	Fjern aktuel linie
E	Får markøren til at holde op med at blinke
F	Får markøren til at blinke
G	Starter klokke (CTRL/G)
H	Standser klokke
I	Indsæt en linie
J	Flyt til begyndelsen af aktuel linie
K	Flyt til slutningen af aktuel linie
L	Begynd scrolling
M	Slut scrolling
N	Tilbagestilling til normalt skærmbillede
O	Nulstil indsæt, quote, reverse og blink mode
P	Slet alt på linien til venstre for markøren

Q Slet alt på linie til højre for markøren
R Gør skærmbilledet mindre
S Skifter til blokmarkør
T Indstil vinduets øverste venstre hjørne
U Skift til understregsmarkør (_)
V Flyt skærmen een linie op
W Flyt skærmen een linie ned
X Skift mellem 40 og 80 kolonner
Y Sæt standard tabulator stop
Z Slet alle tabulator stop

Eksperimenter med ESCape funktionerne. De vil sikkert finde visse funktioner mere nyttige end andre. Bemærk at De kan bruge INST/DEL til at udføre tekstredigering indeni i et vindue.

Når et vindue er dannet, sendes al skærmuddata til den "kasse", De har defineret. Ønsker De at slette vinduesområdet, skal SHIFT og CLEAR/HOME nedtrykkes samtidigt. Et vindue fjernes ved at trykke CLEAR/HOME to gange. Vinduet slettes og markøren går HOME. Vinduer er særdeles nyttige under skrivning, udlistning og kørsel af programmer, fordi de tillader, at man kan arbejde på et skærmområde, mens resten af skærmen forbliver, som den er.

2 MHz PROGRAMUDFØRELSE

FAST og SLOW KOMMANDOERNE

2 MHz operation mode tillader Dem at udføre 'ikke-grafiske' programmer i 80 kolonners format med den dobbelte hastighed. Der kan skiftes mellem normal og hurtig udførelse ved brug af FAST og SLOW kommandoerne.

FAST kommandoen sætter Commodore 128 i 2 MHz mode. Denne kommandos format er:

FAST

SLOW kommandoen sætter Commodore 128 tilbage til 1 MHz mode. Kommandoen er:

SLOW

FUNKTIONSTASTER

De fire taster på Commodore 128 tastaturet på højre side ovenfor det numeriske tastatur er specielle funktionstaster, som kan spare Dem for tid ved at udføre funktioner med blot et enkelt anslag. Den første tast er mærket F1/F2, den anden F3/F4, den tredie F5/F6 og den sidste F7/F8. Ved blot at trykke på tasterne kan funktionerne 1, 3, 5 og 7 umiddelbart fås. For at bruge funktioner 2, 4, 6 og 8 skal SHIFT nedtrykkes samtidig med den aktuelle funktionstast.

Her er standardfunktionerne for hver tast:

F1	F2	F3	F4
GRAPHIC	**DLOAD"**	**DIRECTORY**	**SCNCLR**
F5	F6	F7	F8
DSAVE"	**RUN**	**LIST**	**MONITOR**

TAST 1 går ind i et af de grafiske modes, når De tilføjer nummeret på det grafiske område og trykker på RETURN. GRAPHIC kommandoen er nødvendig for afgivelse af grafiske kommandoer som CIRCLE og PAINT. Flere oplysninger om GRAPHIC i afsnit 6.

TAST 2 skriver DLOAD" på skærmen. De skal nu blot indtaste programnavnet og det afsluttende anførelsestegn og nedtrykke RETURN for at indlæse et program fra diskette, i stedet for selv at skulle skrive DLOAD.

TAST 3 viser DIRECTORY (indholdsfortegnelse) over de filer, som findes på disketten i diskettestationen.

TAST 4 sletter skærmbilledet med kommandoen SCNCLR.

TAST 5 skriver DSAVE" på skærmen. De skal nu blot indtaste programnavnet og nedtrykke RETURN for at gemme et program på diskette.

TAST 6 RUN - starter det aktuelle program.

TAST 7 viser en LISTe af det aktuelle program på skærmen.

TAST 8 giver Dem adgang til maskinsprogsmonitoren. Se endvidere appendiks J for forklaring om monitoren.

OMPROGRAMMERING AF FUNKTIONSTASTERNE

De kan omdefinere eller programmere enhver af disse taster til at udføre en funktion, som passer til Deres ønsker. Ved anvendelse af KEY kommandoen er dette meget let at gøre. Omdefinering kan foretages via et program eller fra direkte mode.

En situation, hvor De kunne ønske at omdefinere en funktionstast, kunne være, hvis De ofte skal bruge den samme kommando, og vil spare tid, i stedet for ustandseligt at skulle indtaste kommandoen. De nye definitioner slettes, når computeren slukkes. De kan omdefinere så mange taster så ofte, De ønsker det.

Hvis De ønsker at omprogrammere F7 funktionstasten, så den bringer Dem til tekst mode fra højopløsnings- eller flerfarvegrafik mode, kan De benytte KEY kommandoen på følgende måde:

KEY 7,"GRAPHIC 0" + CHR$(13)

CHR$(13) er ASCII karakterkoden for RETURN. Så når De nedtrykker F7 tasten efter at have omdefineret den, sker der det, at kommandoen "GRAPHIC 0" skrives automatisk og gives til computeren med RETURN. Hele kommandoer eller serier af kommandoer kan programmeres på een tast.

ANDRE TASTER, SOM KUN BRUGES I C128 MODE

HELP

Hvis De laver en fejl i et program, viser computeren Dem en fejlmeddelelse for at gøre Dem opmærksom på, hvad der er gået galt. Disse fejlmeddelelser forklares nærmere i appendiks A bagest i denne brugervejledning. De kan få mere assistance ved fejl ved at bruge HELP tasten. Efter at have fået en fejlmelding nedtrykker De blot HELP tasten, og hjælpefunktionen lokaliserer Deres fejl. Når HELP nedtrykkes, oplyses den fejlramte linie på skærmen (ved 40 kolonner), eller understreget (ved 80 kolonner). For eksempel:

?SYNTAX ERROR IN LINE 10	Dette er fejlmeddelelsen
HELP	De trykker på HELP
10 PRONT "COMMODORE"	Den fejlramte linie vises oplyst ved 40 kolonners uddata, eller understreget ved 80 kolonners uddata.

NO SCROLL

Tryk på denne tast standser rulning af teksten på skærmen, når markøren når ned til bundlinien. Denne tast kan låses på samme måde som SHIFT/LOCK tasten, så rulning er slået fra, til der atter trykkes på tasten.

CAPS LOCK

Denne tast tillader Dem udelukkende at indtaste store bogstaver uden at skulle nedtrykke SHIFT tasten. Taster med tal og symboler ændres ikke. CAPS LOCK tasten fastlåses, når den nedtrykkes og et fornyet tryk udløser den.

40/80 DISPLAY

40/80 tasten sætter hoved (standard) skærmformat til enten 40 eller 80 kolonner. Den valgte skærm viser alle meddelelser og uddata ved opstart eller hvis RESET eller RUN/STOP/RESTORE benyttes. Denne tast kan kun bruges til indstilling af skærmformatet før computeren tændes eller 'resettes'. Der kan ikke skiftes mode med denne tast efter at computeren er tændt, med mindre De bruger RUN STOP/RESET eller RUN STOP/RESTORE. Afsnit 8 indeholder forklaring om 40/80 kolonners modes.

ALT

ALT tasten gør, at programmer kan tildele en speciel betydning til en given tast eller sæt af taster.
Med mindre den omdefineres med et specielt applikationsprogram, har det ingen effekt at holde ALT tasten og andre taster nedtrykket.

TAB

Denne tast virker på samme måde som tabulationstasten på en skrivemaskine.
Den kan bruges for at sætte tabulatorstop på skærmen og til at flytte markøren mellem disse stop.

LINE FEED

Nedtrykning af denne tast flytter markøren til begyndelsen af næste linie.

YDERLIGERE OPLYSNINGER

Dette afsnit dækker kun nogle af de koncepter, taster og kommandoer, som gør Commodore 128 til en speciel maskine. Yderligere forklaring om BASIC sproget kan findes i BASIC 7.0 ordforklaringen bagest i vejledningen.

AFSNIT 6

FARVE, ANIMATION OG SPRITE-GRAFIK INSTRUKTIONER SPECIELT FOR C128

I C128 mode har Deres Commodore 128's BASIC 7.0 mange nye og kraftfulde kommandoer og instruktioner, som gør det lettere at fremstille avanceret grafisk programmering. Hvert af de to skærmformater i C128 mode (40 eller 80 kolonner) styres af en separat microprocessor chip. 40 kolonners chippen kaldes Video Interface Controller, eller VIC. 80 kolonners chippen kaldes 8563'eren. VIC chippen giver 16 farver og styrer hele den højt detaljerede grafik, som kaldes bit-mapped grafik. 80 kolonners chippen giver også 16 farver, viser kun karakterer og tegngrafik. Derfor skal alle detaljerede grafiske programmer i C128 mode fremstilles i 40 kolonners formatet.

GRAFISKE MULIGHEDER

Som en del af C128 mode's fremragende grafiske muligheder, tilbyder Commodore 128:

- 13 specielle grafiske kommandoer
- 16 farver
- Seks forskellige display modes
- Otte programmerbare og flytbare objekter, kaldet SPRITES
- Kombineret grafik/tekst display

Alle disse muligheder er integreret for at give et enestående grafiksystem, som er let at bruge.

Her følger en kort forklaring på hver enkelt grafisk kommando:

BOX - tegner rektangler på højopløsnings skærmen
CHAR - viser karakterer på højopløsnings skærmen
CIRCLE - tegner cirkler, ellipser og andre geometriske figurer
COLOR - vælger farver for kant, forgrund, baggrund og karakterer
DRAW - viser linier og punkter på højopløsnings skærmen
GRAPHIC - vælger et skærmdisplay (tekst, bit-map, delt-skærm)
PAINT - farvelægger arealer på højopløsnings skærmen
SCALE - giver relative størrelser af figurer på skærmen
SPRDEF - åbner for redigering af sprites
SPRITE - danner, farvelægger, sætter sprite prioritet og udvider sprites.
SPRSAV - gemmer en tekststreng-variabel i et sprite lagerområde og vice versa
SSHAPE - lagrer en del af højopløsnings skærmens udseende i en tekststreng-variabel

De fleste af disse kommandoer gennemgås i eksemplerne i dette afsnit. Se kapitel 5, Basic 7.0 ordforklaring, for detaljerede formater og oplysninger om alle grafiske kommandoer og funktioner, inclusive dem som ikke omtales i dette afsnit.

GRAFISK PROGRAMMERING PÅ C128

Dette afsnit gennemgår et trin-for-trin grafisk programeksempel. Efterhånden som De lærer den enkelte grafiske kommando, kan De føje den til et program, som De vil opbygge, mens De læser dette afsnit. Når De er færdig, har De et komplet grafisk program.

VALG AF FARVER

Det første trin i grafisk programmering på C128 består i at vælge farver for skærmens baggrund, forgrund og kant. For valg af farver tastes:

COLOR source, farve

hvor source er den del af skærmen, De vil farvelægge, farve er den farvekode for den ønskede farve. Se figur 1 om sourcenumre og figur 2 om farvekoder i 40 kolonners format og figur 3 for 80 kolonners format farvenumre.

NUMMER	SOURCE
0	Baggrund (VIC)
1	Forgrund - grafisk skærm (VIC)
2	Forgrundsfarve 1 - flerfarveskærm (VIC)
3	Forgrundsfarve 2 - flerfarveskærm (VIC)
4	Kantfarve - 40 kolonner
5	Tekstfarve for tekstskærm (40 eller 80)
6	Baggrundsfarve - 80 kolonner

(figur 1)
Source numre

FARVEKODE	FARVE	FARVEKODE	FARVE
1	Sort	9	Orange
2	Hvid	10	Brun
3	Rød	11	Lyserød
4	Cyanblå	12	Mørkegrå
5	Purpur	13	Mellemgrå
6	Grøn	14	Lysegrøn
7	Blå	15	Lyseblå
8	Gul	16	Lysegrå

(figur 2)
Farvekoder i 40 kolonners format

FARVEKODE	FARVE	FARVEKODE	FARVE
1	Sort	9	Mørk purpur
2	Hvid	10	Mørkegul
3	Mørkerød	11	Lyserød
4	Lys cyanblå	12	Mørk cyanblå
5	Lys purpur	13	Mellemgrå
6	Mørkegrøn	14	Lysegrøn
7	Mørkeblå	15	Lyseblå
8	Lysegul	16	Lysegrå

(figur 3)
Farvekoder i 80 kolonners format

SKÆRMBILLED TYPER

Deres Commodore 128 har flere forskellige måder, hvorpå den kan vise information på skærmen; parameteren "source" i COLOR kommandoen refererer til forskellige modes af skærmbilledet. Video displayets typer ligger i tre kategorier.
Den første er tekstskærm, som kun viser karakterer, som bogstaver, tal, specialtegn og de grafiske tegn, der findes på forsiden af de fleste C128 taster. C128 kan vise tekst i både 40 kolonners og 80 kolonners skærmformat.

Den anden kategori display mode bruges til højopløst grafik, som billeder og indviklede tegninger. Denne billedtype omfatter standard højopløsnings mode og flerfarve højopløsnings mode. Højopløsning tillader Dem at styre hver enkelt individuel skærmprik, eller pixel (billedelement). Dette giver muligheder for at lave meget detaljerede billeder og anden computerkunst. Disse grafiske billedformater virker kun i 40 kolonners format. 80 kolonners display er beregnet til brug i tekst-mode.

Forskellen mellem tekst- og højopløsnings mode ligger i den måde, hvorpå den enkelte skærmtilstand adresserer og lagrer information. Tekstskærmen kan kun manipulere med hele karakterer, som hver dækker et areal på 8 gange 8 pixel på skærmen. Den mere kraftfulde højopløsnings mode kan kontrollere hver eneste pixel på skærmen.

Den tredie billedtype, delt skærm eller 'split-screen', er en blanding af de to første typer. Delt skærm viser een skærmdel som tekst og en anden som højopløsning (enten standard eller flerfarve). C128 er i stand til det, fordi den bruger to separate dele af hukommelsen til lagring af de to skærme; een del for teksten og den anden del for den grafiske skærm.

Indtast følgende korte program:

```
10 COLOR 0,1: REM BAGGRUNDSFARVE - SORT
20 COLOR 1,3: REM GRAFISK FORGRUNDSFARVE -
   RØD
30 COLOR 4,1: REM KANTFARVE - SORT
```

Dette programeksempel farver baggrunden sort, forgrunden rød og kanten sort.

VALG AF GRAFISK MODE

Det næste trin består i at vælge den passende grafiske mode. Formatet af GRAPHIC kommandoen er følgende:

40 GRAPHIC mode [,c] [,s] eller GRAPHIC CLR

hvor mode er et tal mellem 0 og 5, c er enten 0 eller 1, og s er en værdi mellem 0 og 25. Figur 4 viser de modsvarende værdier:

Mode	Beskrivelse
0	40 kolonners standard tekst
1	Standard højopløsning
2	Standard højopløsning - delt skærm
3	Flerfarve højopløsning
4	Flerfarve højopløsning - delt skærm
5	80 kolonners tekst

Figur 4 Grafiske modes

Parameteren CLR står for CLEAR (slet). Figur 4 forklarer de værdier, som bruges med CLEAR parameteren.

C-værdi	Beskrivelse
0	Slet ikke grafisk skærm
1	Slet grafisk skærm

Figur 5

Når De skal køre Deres program, vil De skulle slette skærmen for eventuelt indhold, så sæt C lig med 1 i GRAPHIC kommandoen. Når De kører programmet næste gang, ønsker De måske at bevare skærmbilledet, istedet for at skulle tegne det hele igen. I så fald sættes C lig med 0.

Parameteren S specificerer hvor starten på tekstskærm i split-screen mode skal være, på linien efter det anførte linienummer. Hvis parameteren S udelades og man vælger split-screen grafisk mode (2 eller 4) vises tekstdelen i rækkerne 20 t.o.m. 25; resten af skærmen er i højopløsning. Parameteren S sætter Dem i stand til at kunne ændre tekstskærmens startlinie til en vilkårlig linie på skærmen. Et nul som parameter indikerer, at der ikke skal bruges delt skærm, men kun tekst.

Den sidste GRAPHIC kommando parameter er GRAPHIC CLR. Hvis De først angiver en højopløsnings grafikkommando, reserverer Commodore 128 et område på 9K til Deres grafikskærm. 8K reserveres til data i Deres 'bit map' og 1K bruges til farvedata. Da 9K er en ret stor del af hukommelsen, kunne De måske have lyst til at bruge dette område i andre sammenhænge senere i programmet. Dette er formålet med GRAPHIC CLR. Denne kommando reorganiserer hukommelsen og giver Dem de 9K hu-

kommelse tilbage, som blev brugt til højopløsnings-skærmen, så det kan bruges til andre formål.

Kommandoens format ser således ud:

GRAPHIC CLR

Når dette format anvendes, skal alle andre grafikkommando parametre udelades.

Føj følgende kommando til Deres program. Den sætter C128 i standard højopløsnings mode og allokerer en 8K højopløsnings skærm (og 1K med farvedata) så De kan fremstille grafik.

40 GRAPHIC 1,1

Det andet ettal sletter højopløsnings-skærmen. Hvis De ikke ønsker skærmen slettet, skal det erstattes med et 0, eller fuldstændig udelades.

OBS! Hvis De nogensinde går i stå i højopløsnings-mode, og ikke kan komme tilbage til tekstskærmen, kan De nedtrykke RUN/STOP og RESTORE tasterne samtidigt, eller trykke på ESC tasten efterfulgt af X, for at returnere til 80 kolonners skærmen. Selv om det kun er muligt at vise grafik med VIC (40 kolonner) chippen, kan man stadig skrive grafiske programmer i 80 kolonners formatet. Hvis De har en Commodore 1902 monitor, og De ønsker at betragte Deres grafiske program, mens det kører, skal De vælge 40 kolonners uddata ved at sætte kontakten på monitoren på 40 kolonners uddata.

FREMVISNING AF GRAFIK PÅ SKÆRMEN

Da De nu har valgt en grafisk mode og de ønskede farver, er De næsten klar til at fremvise grafik på skærmen. Begynd med en cirkel.

Denne instruktion tegner en trekant øverst på skærmen. De fire talsæt angiver X og Y koordinaterne for trekantens hjørner. Læg mærke til, at den første og sidste koordinat er den samme, idet De skal afslutte tegningen samme sted, som De begyndte. Med denne udgave af DRAW intruktionen kan De tegne stort set enhver geometrisk figur, som trapezer, parallelogrammer og polygoner.

DRAW instruktionen har endvidere en tredie mulighed.

De kan tegne eet punkt ad gangen ved at angive start X og Y værdierne på denne måde:

150 DRAW 1,160,160

Denne instruktion tegner en prik nedenunder cirklen.
Som De kan se, tilbyder DRAW instruktionen mange fremragende muligheder, som sætter Dem i stand til at frembringe et ubegrænset antal computertegninger på Deres skærm.

PAINT KOMMANDOEN - FARVELÆGNING AF OPRIDSEDE OMRÅDER

Med DRAW instruktionen kan De tegne figuromrids på skærmen. Hvad nu hvis De ønsker at udfylde områderne indenfor de tegnene linier? Her får De brug for PAINT (male) instruktionen. PAINT instruktionen gør nøjagtigt, hvad navnet antyder – den udfylder, eller maler, angivne områder med farve.
Nøjagtigt som en maler dækker et stykke lærred med oliemaling, dækker PAINT skærmområder med een af 16 farver. Indtast for eksempel:

160 PAINT 1,150,97

Linie 160 maler den cirkel, De har tegnet i linie 60. PAINT instruktionen udfylder et angivet område, til den når grænsefladen til samme farve (eller en hvilken som helst anden end en baggrundsfarve). Når Commodore 128 holder op med at male, efterlader den pixelmarkøren på det sted, hvor den begyndte at male (i dette tilfælde punktet 150,97).

Her er to PAINT instruktioner mere:

180 PAINT 1,50,25
200 PAINT 1,225,125

Linie 180 udfylder trekanten og linie 200 udfylder den tomme kasse.

* VIGTIGT UDFYLDNINGS RÅD: Hvis De som udgangspunkt for en PAINT instruktion vælger et punkt, som allerede er farvelagt, vil Commodore 128 ikke udfylde det område, som er omkranset af Deres figur. De må vælge et startpunkt som ligger indenfor omridset af den figur, De ønsker farvelagt. Startpunktet kan ikke ligge på figurens omridslinie.

VISNING AF KARAKTERER PÅ EN HØJOPLØSNINGS-SKÆRM - CHAR KOMMANDOEN

Indtil nu har dette eksempel arbejdet i standard højopløsningsmode.
Højopløsnings-mode bruger en skærm helt forskellig (i hukommelsen) fra tekst mode – den mode, i hvilken man indtaster programmer eller tekst.
Hvis man går ind i højopløsnings-mode og prøver på at indtaste karakterer på skærmen, sker der intet. Dette skyldes, at de tegn, De indtaster, betragtes som inddata til tekstskærmen, og det er højopløsnings-skærmen, De kan se. Af og til er det nødvendigt at vise karakterer på højopløsningsskærmen, hvis man f. eks. er i gang med at oprette skemaer og grafer.
CHAR kommandoen er beregnet specielt til dette formål. Med CHAR kommandoen er det muligt at vise standardkarakterer på en højopløsnings-skærm, som det fremgår af følgende:

220 CHAR 1, 11,24,"GRAFISK EKSEMPEL"

Dette viser teksten "GRAFISK EKSEMPEL", begyndende på linie 25, kolonne 12. CHAR kommandoen kan også bruges i tekstmode, men er egentlig lavet primært til højopløsning.

ÆNDRING AF STØRRELSEN AF GRAFISKE FIGURER - SCALE KOMMANDOEN

Commodore 128 har en anden grafisk instruktion, som tilbyder ekstra styrke til Deres grafiske system. SCALE instruktionen åbner mulighed for at formindske den grafik, som ses på skærmen. Dette betyder, at der kan vises mere grafik på skærmen end normalt. SCALE instruktionen tilføjer også en anden facilitet, som kan forklares med følgende.

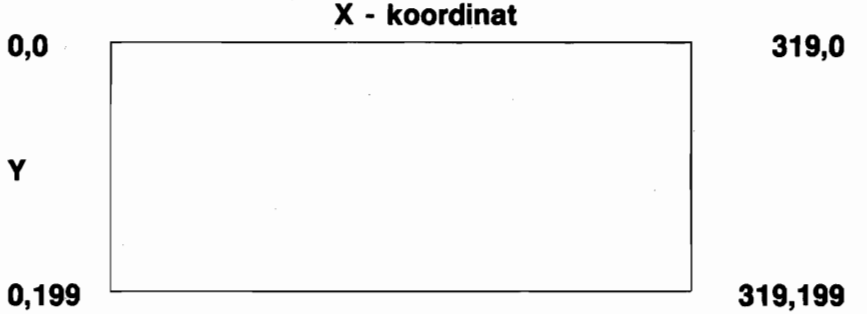
TEGNING AF EN CIRKEL - CIRCLE KOMMANDOEN

Til tegning af en cirkel benyttes CIRCLE instruktionen således:

60 CIRCLE 1,160,100,40

Dette frembringer en cirkel midt på skærmen. CIRCLE instruktionen har ni parametre, med hvilke De kan fremstille forskellige cirkeltyper og geometriske figurer. Ved for eksempel at ændre tallene i CIRCLE instruktionen i linie 60, kan De opnå forskelligt cirkelomfang eller variationer i figuren (f.eks. en oval). CIRCLE instruktionen tilfører kraft og muligheder til programmering af C128 grafik i BASIC. Betydningen af tallene i CIRCLE instruktionen står forklaret i kapitel 5, BASIC 7.0 ordforklaring.

På Deres Commodore 128 skærm bliver det punkt, hvor X=0 og Y=0, i skærmens øverste venstre hjørne kaldt for HOME positionen. I standard geometri er punktet, hvor X og Y er lig 0, nøjagtig i centrum af en grafisk figur.
Figur 6 viser placeringen af X og Y skærmkoordinaterne og C128 skærmens fire hjørner.



Figur 6

Her er tallenes betydning:

- 1 er farveområdet (i dette tilfælde forgrunden)
- 160 er start X (den horisontale) koordinaten.
- 100 er start Y (den vertikale) koordinaten.
- 40 er radius.

TEGNING AF EN KASSE - BOX KOMMANDOEN

Prøv nu med en kasse, indtast følgende:

80 BOX 1,20,60,100,140,0,1

Dette tegner en kasse til venstre for cirklen. Oplysninger om, hvad talangivelserne i kasse-instruktionen betyder, kan findes i BASIC ordforklaringen bagest i vejledningen. BOX instruktionen har syv parametre, hvormed De kan udvælge og modificere forskellige kassetyper. Skift nu forgrundsfarve og tegn en kasse til højre for cirklen med disse instruktioner:

90 COLOR 1,9:REM SKIFT AF FORGRUNDSFARVE
100 BOX 1,220,60,300,140,0

Eksperimenter med BOX instruktionen i linie 80 for at fremstille forskellige firkanter.

TEGNING AF LINIER, PUNKTER OG ANDRE FIGURER - DRAW KOMMANDOEN

De er nu i stand til at vælge grafisk mode og farver, samt til at tegne cirkler og kasser på skærmen. Med en anden grafikinstruktion, DRAW, kan De tegne linier på skærmen, som var den et stykke papir. Følgende instruktion tegner en linie under kasserne og cirklen.

120 DRAW 1,20,180 TO 300,180

Tallene betyder følgende: 1 angiver det område, hvorpå der skal tegnes (i dette tilfælde forgrunden) Tallet 20 er X koordinaten (vandret) og 180 er Y (lodret) koordinaten. Tallet 300 er det sted, hvor X koordinaten slutter og 180 er slutpunktet for Y koordinaten. En tegnet linie kan slettes ved at ændre område (1) i DRAW instruktionen med et 0. Linien tegnes så på baggrunden, hvilket ser ud som om linien slettes. Prøv at bruge forskellige koordinater og andre områder, som kan bruges i forbindelse med DRAW instruktionen.

DRAW instruktionen kan udformes på en anden måde, så De kan tegne en linie, ændre retning og tegne en anden, så linierne er sammenhængende og danner en geometrisk figur. Prøv for eksempel denne instruktion:

130 DRAW 1,230,0 TO 30,0 TO 40,40 TO 230,0

I standard højopløsnings-mode har 40 kolonners skærmen 320 vandrette og 200 lodrette koordinater. I flerfarve højopløsnings-mode indeholder 40 kolonners skærmen kun halvt så megen vandret opløsning som standard højopløsnings-mode, nemlig 160 gange 200. Denne reduktion i opløsning kompenseres der for ved muligheden af at bruge yderligere en farve, således at der kan bruges tre farver indenfor en matrix på 8 gange 8 karakterer.
Standard højopløsnings-mode kan kun vise to farver indenfor en matrix på 8 gange 8 karakterer.

Når De bruger SCALE instruktionen, har både standard højopløsnings-mode og flerfarve højopløsnings-mode koordinater, som er proportionale med hinanden. Skalaen rækker fra 0 til et maksimum på 1023 vandrette koordinater.
Dette gælder uanset, om De er i standard højopløsnings-mode eller flerfarve højopløsnings-mode.

For at skalere skærmbilledet tastes:

SCALE 1,x,y

og skærmkoordinaterne rækker fra 0 til 65535 uanset om De er i standard eller flerfarve højopløsnings-mode. Skaleringen afbrydes ved at taste:

SCALE 0

hvorved koordinaterne vender tilbage til deres normale værdier.

For at se, hvorledes SCALE influerer på Deres program, kan De tilføje denne linie:

50 SCALE 1,500,500

og tast RUN for at se, hvad der sker.

Kapitel 5 indeholder flere detaljer om SCALE kommandoen.

Obs! SCALE kommer efter GRAPHICS og har ingen indflydelse på CHAR.

FREMSTILLING AF ET GRAFISK PROGRAMEKSEMPEL

De har indtil nu lært adskillige grafiske instruktioner. Sæt nu programmet sammen i en helhed og se, hvorledes instruktionerne arbejder sammen. Herunder ses, hvorledes programmet ser

ud nu. Farveinstruktionerne i linierne 70, 110, 140, 170, 190 og 210 er tilføjet for at vise det enkelte objekt i afvigende farver.

```
10 COLOR 0,1 :REM VALG AF BAGGRUNDSFARVE
20 COLOR 1,3 :REM VALG AF FORGRUNDSFARVE
30 COLOR 4,1 :REM VALG AF KANTFARVE
40 GRAPHIC1,1 :REM VALG AF STD. HØJOPLØSNING
60 CIRCLE1,160,100,40,40 : REM CIRKEL
70 COLOR 1,6 :REM SKIFT FORGRUNDSFARVE
80 BOX1,20,60,100,140,0,1 :REM KASSE
90 COLOR 1,9 :REM SKIFT FORGRUNDSFARVE
100 BOX1,220,60,300,140,0 :REM KASSE
110 COLOR 1,8 :REM SKIFT FORGRUNDSFARVE
120 DRAW 1,20,180 TO 300,180 :REM TEGN EN LINIE
130 DRAW 1,230,0 TO 30,0 TO 40,40 TO 230,0 :REM TEGN TREKANT
140 COLOR 1,15 :REM SKIFT FORGRUNDSFARVE
150 DRAW 1,160,160 :REM TEGN ET PUNKT
160 PAINT 1,150,97 :REM FARVELÆG CIRKLEN
170 COLOR 1,5 :REM SKIFT FORGRUNDSFARVE
180 PAINT 1,50,25 :REM FARVELÆG TREKANT
190 COLOR 1,7 :REM SKIFT FORGRUNDSFARVE
200 PAINT 1,225,125 :REM FARVELÆG KASSE
210 COLOR 1,11 :REM SKIFT FORGRUNDSFARVE
220 CHAR1,11,24,"GRAFISK EKSEMPEL" :REM VIS TEKST
230 FOR I = 1 TO 5000:NEXT: GRAPHIC 0,1:COLOR 1,2
```

Linierne 10 til 30 vælger baggrundsfarve, forgrundsfarve og kantfarve.
Linie 40 vælger den grafiske mode.
Linie 50 placerer pixelmarkøren på koordinaterne 150,130.
Linie 60 tegner en cirkel.
Linie 80 tegner en farvet kasse.
Linie 100 tegner omridset af en kasse.
Linie 120 tegner en ret linie i bunden af skærmbilledet.
Linie 130 tegner en trekant.
Linie 150 tegner et enkelt punkt under cirklen.
Linie 160 maler cirklen.
Linie 180 maler trekanten.
Linie 200 maler den tomme kasse.
Linie 220 udskriver karaktererne "GRAFISK EKSEMPEL" nederst på skærmen.
Linie 230 forsinker programmet, så de kan se grafikken på skærmen skifte tilbage til tekst mode og farver karaktererne sorte.

Hvis De ønsker, at grafikken skal forblive på skærmen, skal De udelade GRAPHIC instruktionen i linie 230.

Her er nogle flere programeksempler, som benytter de grafik instruktioner, De netop har lært.

```
10 COLOR 0,1
20 COLOR 1,8
30 COLOR 4,1
40 GRAPHIC1,1
50 FOR I = 80 TO 240 STEP 10
60 CIRCLE1,I,100,75,75
70 NEXT
80 COLOR 1,5
90 FOR I = 80 TO 250 STEP 10
100 CIRCLE1,I,100,50,50
110 NEXT
120 COLOR 1,7
130 FOR I = 50 TO 280 STEP 10
140 CIRCLE1,I,100,25,25
150 NEXT
160 FOR I = 1 TO 7500:NEXT:GRAPHIC0,1:COLOR1,2
```

```
10 GRAPHIC 1,1
20 COLOR0,1
30 COLOR4,1
40 FOR I = 1 TO 50
50 Z = INT(((RND(1))*16)+1)* 1
60 COLOR1,Z
70 X = INT(((RND(1))*30)+1)*10
80 Y = INT(((RND(1))*20)+1)*10
90 U = INT(((RND(1))*30)+1)*10
100 V = INT(((RND(1))*20)+1)*10
110 DRAW,X,Y TO U,V
120 NEXT
130 SCNCLR
140 GOTO 40
```

```
10 COLOR4,7:COLOR0,7:COLOR1,1
20 GRAPHIC1,1
30 FOR I = 400 TO 1 STEP -5
40 DRAW 1,150,100 TO I,1
50 NEXT
60 FOR I = 1 TO 400 STEP 5
70 DRAW 1,150,100 TO 1,I
80 NEXT
90 FOR I = 40 TO 320 STEP 5
100 DRAW 1,150,100 TO I,320
110 NEXT
120 FOR I = 320 TO 30 STEP -5
130 DRAW 1,150,100 TO 320,I
140 NEXT
150 FOR I = 1 TO 7500:NEXT:GRAPHIC0,1:COLOR1,1
```

Indtast eksemplerne på Deres computer. Udfør dem med RUN og gem dem med SAVE, så De senere kan bruge dem. En af de bedste måder at lære programmering på er, at studere programeksempler og se, hvorledes instruktionerne udfører deres funktioner. De vil snart være i stand til at bruge grafiske instruktioner til at skabe spændende grafik på Deres Commodore 128.

Har De brug for yderligere oplysninger om nogen BASIC instruktion eller kommando, kan disse findes i BASIC 7.0 ordlisten bagest i vejledningen.
De er nu i besiddelse af et sæt grafiske kommandoer, som sætter Dem i stand til at skabe et uendeligt antal kunstneriske grafiske billeder. Men Commodore 128 grafikkens muligheder standser ikke her. Commodore 128 har et andet sæt SPRITE instruktioner, som gør sprite fremstilling og styring hurtig, enkel og enestående. For dem, som ikke er familiære med sprites, består sprites af flytbare grafiske objekter, som kan dannes i enhver form, og som kan bevæges på skærmen. Disse højniveau-instruktioner giver Dem mulighed for at oprette sprites uden brug af grafisk papir eller ekstra redigeringsudstyr. Disse instruktioner er nu vejen, den nye teknologi bag skabelse og styring af sprites. Gennemlæs næste afsnit og tag Deres første trin i at lære computer animation.

SPRITES: PROGRAMMERBARE FLYTBARE OBJEKT BLOKKE

De har allerede lært noget om nogle af Commodore 128's exceptionelle grafiske muligheder. De har lært at benytte det første sæt højniveau grafiske instruktioner til at tegne cirkler, kasser, streger og prikker. De har også lært at farve skærmen, skifte mellem grafiske tilstande, male objekter på skærmen og at skalere dem. Det er nu på tide at tage næste skridt i grafisk programmering - sprite animering.
Hvis De tidligere har arbejdet med Commodore 64, ved De allerede en del om sprites. For dem, som ikke kender C64 grafikken, defineres en sprite som et bevægeligt objekt, som kan gives enhver facon eller udseende. Sprites kan farves i op til tre af 16 forskellige farver. Sprites kan ovenikøbet være flerfarvede. Det bedste er næsten, at de kan flyttes på skærmen.
Sprites åbner døren til computer animation.

Her anføres det sæt sprite instruktioner, De vil lære om i dette afsnit:

MOVSPR
SPRDEF
SPRITE
SPRSAV
SSHAPE

OPBYGNING AF SPRITES

Første trin i sprite programmering er fastlæggelse af spritens udseende. Hvis De for eksempel ønsker at lave et rumskib eller en racervogn-sprite.

Før spriten kan farves eller flyttes må dens udseende først fastlægges.
Dette kan, i Commodore 128 mode, gøres på følgende tre måder:

1) Ved brug af SPRITE instruktioner i et program.
2) Ved anvendelse af SPRite DEFinations mode.
3) Ved anvendelse af samme metode som på Commodore 64.

BRUG AF SPRITE INSTRUKTIONER I ET PROGRAM

Denne metode er sandsynligvis den hurtigste og nemmeste må-

de til at skabe sprites i C128 mode. Den anvender indbyggede instruktioner, så De ikke har brug for andre værktøjer for oprettelse af sprites, til forskel fra de to andre metoder, der findes. Denne metode anvender det første sæt grafiske instruktioner, som De har lært i det første afsnit. Her er den generelle fremgangsmåde. Detaljer vil blive tilføjet, efterhånden som de behøves.

1) Tegn et billede med det første sæt grafiske instruktioner, De lærte i sidste afsnit, som DRAW, CIRKLE, BOX og PAINT. Sæt derefter billedets dimensioner til 24 pixel koordinater bredt gange 21 pixel koordinater højt, hvis De arbejder i standard højopløsnings-mode, eller 12 koordinater bredt gange 21 højt, hvis De foretrækker flerfarve højopløsnings-mode.
2) Brug SSHAPE instruktionen til at gemme billedets data i en strengvariabel.
3) Overfør billedets data fra strengvariablen til en sprite ved brug af SPRSAV instruktionen.
4) Klargør spriten, farvelæg den, vælg enten standard eller flerfarve mode og udvid den, alt med SPRITE instruktionen.
5) Bevæg spriten med MOVSPR instruktionen.

TEGNING AF SPRITENS UDSEENDE

Herunder anføres de aktuelle instruktioner, som udfører disse sprite operationer. Når De er færdig med dette afsnit, vil De have skrevet Deres første sprite program. De vil kunne køre programmet med RUN, så ofte De har lyst og lagre det med SAVE for fremtidig anvendelse.

Det første trin består i at tegne et billede (24 gange 21 pixels) på skærmen ved brug af DRAW, CIRCLE, BOX eller PAINT. Dette eksempel udføres i standard højopløsnings-mode og har sort baggrund. Her er de instruktioner, som sætter grafisk mode og farver baggrunden sort.

```
5 COLOR 0,1 :REM GIVER SORT BAGGRUNDSFARVE
10 GRAPHIC 1,1 :REM SÆTTER STD.HØJOPLØSNINGS MODE
```

De følgende instruktioner tegner et billede af en racerbil i skærmens øverste venstre hjørne. Disse instruktioner har De lært i sidste afsnit.

```
5 COLOR 0,1:COLOR 4,1
10 GRAPHIC 1,1
15 BOX 1,2,2,45,45
20 DRAW 1,17,10 TO 28,10 TO 26,30 TO 19,30 TO 17,10:REM KAROSSERI
22 DRAW 1,11,10 TO 15,10 TO 15,18 TO 11,18 TO 11,10:REM Ø.V.HJUL
24 DRAW 1,30,10 TO 34,10 TO 34,18 TO 30,18 TO 30,10:REM Ø.H.HJUL
26 DRAW 1,11,20 TO 15,20 TO 15,28 TO 11,28 TO 11,20:REM N.V.HJUL
28 DRAW 1,30,20 TO 34,20 TO 34,28 TO 30,28 TO 30,20:REM N.H.HJUL
30 DRAW 1,26,28 TO 19,28:REM KØLERGITTER
32 BOX 1,20,14,26,18,90,1:REM FRONTRUDE
35 BOX 1,150,35,195,40,90,1:REM VEJLINIE
37 BOX 1,150,135,195,140,90,1:REM VEJLINIE
40 BOX 1,150,215,195,220,90,1:REM VEJLINIE
42 DRAW 1,150,180 TO 300,180:DRAW 1,50,180 TO 50,190:REM MÅLSTREG
43 DRAW 1,300,180 TO 300,190:DRAW 1,50,190 TO 300,190:REM MÅLSTREG
44 CHAR 1,18,23,"M Å L":REM SKRIVER M Å L
```

Udfør nu programmet med RUN. De har netop tegnet en hvid racerbil i en kasse i skærmens øverste venstre hjørne. De har også tegnet en målstreg nederst på skærmen. Indtil nu er racerbilen kun et fast billede. Bilen er endnu ikke en sprite, men De har netop afsluttet første trin i sprite programmering.

LAGRING AF SPRITE DATA MED SSHAPE

Næste trin består i at lagre billedet i en tekststreng med SSHAPE instruktionen. Dette gøres således:

45 SSHAPE A$,11,10,34,30:REM GEMMER BILLEDET I EN STRENG

SSHAPE kommandoen lagrer skærmbilledet (bitmønstret) i en strengvariabel for senere brug, i henhold til de anførte skærmkoordinater.

Tallene 11, 10, 34, 30 er billedets koordinater. Koordinaterne skal placeres det rigtige sted, da SSHAPE instruktionen ellers ikke kan lagre Deres billeddata korrekt. Hvis SSHAPE instruktionen placeres på en tom skærmadresse, vil datastrengen også blive tom. Når De senere overfører den til en sprite, vil De opdage, at der ingen data findes. Vær altså sikker på at placere SSHAPE instruktionen direkte på den rigtige koordinat. Vær også sikker på, at De danner billedet med dimensionerne 24 gange 21 pixels, som er størrelsen af en enkelt sprite.

SSHAPE instruktionen overfører billedet af racerbilen til en datastreng, som computeren kan forstå. Datastrengen, A$, gemmer en streng bestående af nuller og ettaller i computerens hukommelse, som danner skærmbilledet. Som ved al computergrafik, har computeren en måde, hvorpå den kan visualisere grafik ved hjælp af bits i hukommelsen. Hver eneste prik på skærmen, kaldet en pixel, modsvares af en bit, som styrer den, i computerens hukommelse. Hvis denne bit har værdien 1, er pixelen på skærmen tændt og har den valgte forgundsfarves farve. Er styrebitten lig med 0, slukkes pixelen og har samme farve som baggrundsfarven.

LAGRING AF EN SPRITE'S BILLEDDATA

Deres billede er nu lagret i en streng. Næste trin består i at overføre billedets data fra datastrengen (A$) til en sprite, så De kan animere den.
Dette gøres med SPRSAV instruktionen på følgende måde:

```
50 SPRSAV A$,1:REM LAGRER DATASTRENGEN I SPRITE 1
55 SPRSAV A$,2:REM LAGRER DATASTRENGEN I SPRITE 2
```

Deres billeddata er nu overført til sprite 1 og sprite 2. Begge sprites har samme data, så de ser fuldstændig ens ud. De kan endnu ikke se dem, fordi de først skal tændes.

AKTIVERING AF SPRITES

SPRITE instruktionen tænder for en bestemt sprite (nummereret 1 til 8), farvelægger den, specificerer dens skærmprioritet, udvider spritens størrelse og afgør hvilken grafisk tilstand, der er valgt. Skærmprioritet er afgørende for, om spriten passerer foran eller bagved objekter på skærmen.
Sprites kan udvides til deres dobbelte størrelse i enten vandret eller lodret retning. Den valgte grafiske tilstand afgør om spriten er en standard højopløsnings-sprite eller en flerfarve-sprite. Her er de to instruktioner, der tænder for sprite 1 og 2.

```
60 SPRITE 1,1,7,0,0,0,0:REM TÆNDER FOR SPRITE 1
65 SPRITE 2,1,3,0,0,0,0:REM TÆNDER FOR SPRITE 2
```

Her er betydning af tallene i SPRITE instruktionerne:

```
SPRITE #,O,F,P,X,Y,M
```

- sprite nummer (1 til 8)
O - Tænd (O=1) eller Sluk (O=0)
F - 1 Farve (1 til 16)
P - Prioritet - Hvis P=0, er spriten foran objekter på skærmen
Hvis P=1, er spriten bagved objekter på skærmen
X - Hvis X=1, udvides spriten i vandret plan (X)
Hvis X=0, har spriten sin normale størrelse
Y - Hvis Y=1, udvides spriten i lodret plan (Y)
Hvis Y=0, har spriten sin normale størrelse
M - Hvis M=1, vises spriten i flerfarve-højopløsnings-mode
Hvis M=0, vises spriten i standard højopløsnings-mode

Som De kan se, er SPRITE instruktionen meget kraftfuld og styrer mange spritekvaliteter.

SPRITE BEVÆGELSE MED MOVSPR

Nu kan De se Deres sprite på skærmen. Nu mangler De kun at bevæge den.
MOVSPR instruktionen styrer bevægelsen af en sprite og giver Dem mulighed for at animere den på skærmen. MOVSPR instruktionen kan anvendes på to måder. For det første kan MOVSPR placere spriten på et udvalgt sted på skærmen ved bruge af lodrette og vandrette koordinater. Føj følgende instruktioner til Deres program:

```
70 MOVSPR 1,240,0:REM PLACERING AF SPRITE 1 -
X=240, Y=0
80 MOVSPR 2,120,0:REM PLACERING AF SPRITE 2 -
X=120, Y=0
```

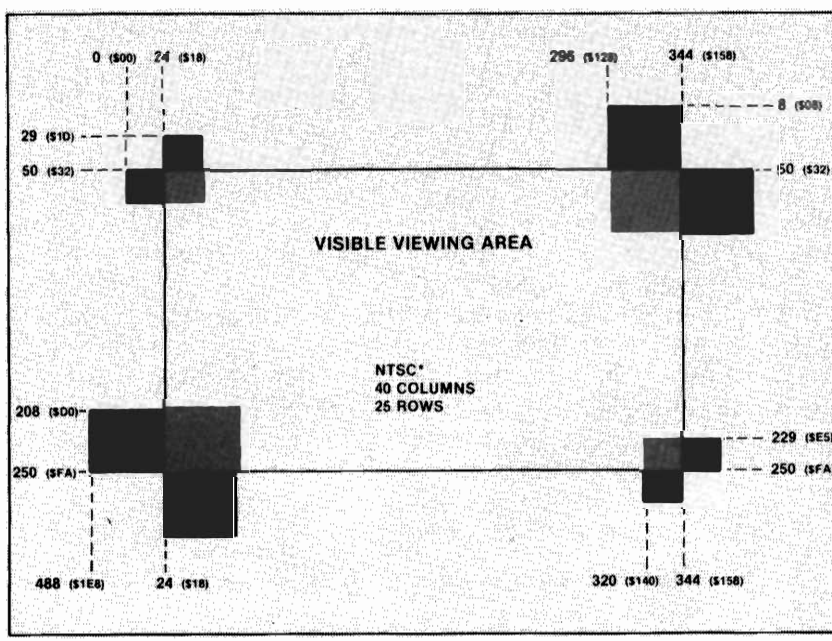
Linie 70 placerer sprite 1 i (pixel) kolonne 240, række 0. Linie 80 placerer sprite 2 i (pixel) kolonne 120, række 0. MOVSPR instruktionen kan også bruges til at flytte sprites i relation til deres oprindelige positioner. For eksempel placere sprite 1 og 2 i koordinaterne som i linie 70 og 80. De ønsker dem flyttet fra deres oprindelige position til en anden. Brug nedenfor viste instruktioner til at flytte spritene langs en specificeret rute på skærmen:

```
85 MOVSPR 1,180 # 6:REM FLYTTER SPRITE 1 FRA
TOP TIL BUND
90 MOVSPR 2,180 # 7:REM FLYTTER SPRITE 2 FRA
TOP TIL BUND
```

Det første tal i denne instruktion er sprite nummeret. Det næste tal er det antal grader, der skal flyttes (i urets retning) i relation til spritens oprindelige plads. Nummertegnet # indikerer, at spriten skal flyttes relativt til en startposition i stedet for en absolut position, som i linie 70 og 80. Det sidste tal angiver den fart, hvormed spriten skal flyttes langs dens rute på skærmen; farten angives med et tal fra 0 t.o.m. 15.

MOVSPR kommandoen kan udføres på to alternative måder. Se kapitel 5, BASIC 7.0, ordforklaring for disse.

Sprites anvender en koordinatplan helt forskellig fra højopløsnings koordinaterne. Højopløsnings koordinaterne går fra punkt 0,0 (øverste venstre hjørne) til 319,199 (nederste højre hjørne). De synlige sprite koordinater begynder i punkt 50,24 og slutter i punkt 250,344. Resten af sprite koordinaterne ligger udenfor skærmen og er ikke synlige, men spritene kan stadig følge dem. Disse koordinater udenfor skærmen gør, at spritene kan bevæge sig til og fra skærmen på en smart måde. Figur 7 illustrerer sprite koordinat planet og de synlige sprite positioner.



* North American television transmission standards for your home TV.

Udfør nu hele programmet med alle trin includeret. De har nu skrevet Deres første sprite program. De har lavet en racerbane med to racerbiler. Prøv at tilføje flere biler og flere objekter på skærmen. Eksperimenter med at tegne andre sprites og tag dem med på racerbanen. De er nu godt igang med sprite programmering. Brug fantasien og udtænk andre scener og objekter, som kan animeres. De vil snart være i stand til at fremstille alle mulige slags animerede computer "tegnefilm".

For at standse sprites, nedtrykkes RUN/STOP og RESTORE samtidig.

FREMSTILLING AF ET SPRITE PROGRAM

De har nu et sprite programeksempel, der virker. Her ses hele programmet:

```
5 COLOR 0,1:COLOR 4,1
10 GRAPHIC 1,1
15 BOX 1,2,2,45,45
20 DRAW 1,17,10 TO 28,10 TO 26,30 TO 19,30 TO 17,10:REM KAROSSERI
22 DRAW 1,11,10 TO 15,10 TO 15,18 TO 11,18 TO 11,10:REM Ø.V.HJUL
24 DRAW 1,30,10 TO 34,10 TO 34,18 TO 30,18 TO 30,10:REM Ø.H.HJUL
26 DRAW 1,11,20 TO 15,20 TO 15,28 TO 11,28 TO 11,20:REM N.V.HJUL
28 DRAW 1,30,20 TO 34,20 TO 34,28 TO 39,28 TO 30,20:REM N.H.HJUL
30 DRAW 1,26,28 TO 19,28:REM KØLERGITTER
32 BOX 1,20,14,26,18,90,1:REM FRONTRUDE
35 BOX 1,150,35,195,40,90,1:REM VEJLINIE
37 BOX 1,150,135,195,140,90,1:REM VEJLINIE
40 BOX 1,150,215,195,220,90,1:REM VEJLINIE
42 DRAW 1,150,180 TO 300,180:DRAW 1,50,180 TO 50,190:REM MÅLSTREG
43 DRAW 1,300,180 TO 300,190:DRAW 1,150,190 TO 300,190:REM MÅLSTREG
44 CHAR 1,18,23,"M Å L":REM SKRIVER M Å L
45 SSHAPE A$,11,10,34,31:REM GEMMER BILLEDET I EN STRENG
50 SPRSAV A$,1:REM LAGRER DATASTRENGEN I SPRITE 1
55 SPRSAV A$,2:REM LAGRER DATASTRENGEN I SPRITE 2
60 SPRITE 1,1,7,0,0,0,0:REM TÆNDER FOR SPRITE 1
65 SPRITE 2,1,3,0,0,0,0:REM TÆNDER FOR SPRITE 2
70 MOVSPR 1,240,70
80 MOVSPR 2,120,70
85 MOVSPR 1,180 £ 6
90 MOVSPR 2,180 £ 7
95 FOR I = 1 TO 5000:NEXT
99 GRAPHIC 0,1
```

Linie 5 farver skærmen sort.
Linie 10 sætter standard højopløsnings-mode.
Linie 15 tegner en kasse i skærmens øverste venstre hjørne. Derefter tegner linierne 20 t.o.m. 32 racerbilen.
Linierne 35 t.o.m. 44 tegner racerbanen og målstregen.
Linie 45 overfører racerbilens billeddata til en strengvariabel.
Linierne 50 og 55 overfører strengvariablens indhold til sprite 1 og 2.
Linierne 60 og 65 tænder for sprite 1 og 2.
Linierne 70 og 80 placerer spritene øverst i skærmbilledet. Endelig får linierne 85 og 90 spritene (de to biler) til at køre om kap, til de når målstregen.

I dette afsnit har De lært hvorledes man skaber sprites ved anvendelse af indbyggede C128 grafik instruktioner som DRAW og BOX. De har lært at styre sprites ved brug af Commodore 128 sprite instruktioner. Der findes yderligere to måder, hvorpå Commodore 128 kan fremstille sprites. Den ene metode er ved anvendelse af den indbyggede SPRite DEFinitions metode, som gennemgås i følgende afsnit. Den anden metode ligner den, som bruges på Commodore 64. Se herom i C64 Programmer's Reference Guide.

SPRDEF KOMMANDOEN - SPRite DEFinitions mode

Commodore 128 har en indbygget SPRite DEFinitions mode, som sætter Dem i stand til at fremstille sprites på Deres Commodore 128. De kender muligvis den måde, hvorpå der skabes sprites på en Commodore 64. Dertil kræves en separat 'sprite editor' eller at man tegner spriten på specielt grafisk papir og derefter indlæser de kodede spritedata og poker dem ind i en ledig sprite blok. Med den nye C128 sprite definitions kommando SPRDEF, kan De konstruere og redigere Deres egne sprites i et specielt arbejdsområde.

For at komme i SPRDEF mode tastes:

SPRDEF

og der trykkes på **RETURN**. Commodore 128 viser et sprite 'gitter' på skærmen og spørger samtidig:

SPRITE NUMBER?

Indtast et nummer mellem 1 og 8. Computeren udfylder 'gitte-

ret' og viser den modsvarende sprite i skærmens øverste højre hjørne. Fra nu af vil vi referere til sprite 'gitteret' som arbejdsområdet. Arbejdsområdets dimension er 24 karakterer bredt og 21 karakterer højt. Hver karakterposition indenfor arbejdsområdet svarer til 1 pixel i spriten, da en sprite er 24 pixels bred og 21 pixels høj. Mens De arbejder i arbejdsområdet i SPRDEF mode, har De adskillige redigeringskommandoer, De kan bruge. Her følger en oversigt over disse kommandoer:

SPRDEF MODE KOMMANDO OVERSIGT

CLR-tast	Sletter hele arbejdsområdet
M-tast	Tænder/slukker flerfarve-sprite
CTRL 1-8	Vælger sprite forgrundsfarve 1-8
C= 1-8	Vælger sprite forgrundsfarve 9-16
1 tast	Tænder pixels i baggrundsfarven
2 tast	Tænder pixels i forgrundsfarven
3 tast	Tænder områder i flerfarve 1
4 tast	Tænder områder i flerfarve 2
A tast	Tænder/slukker automatisk markørbevægelse
CRSR taster	Flytter markøren i arbejdsområdet
RETURN	Flytter markøren til næste linies begyndelse
HOME tast	Flytter markøren til arbejdsområdets øverste venstre hjørne.
X tast	Styrer horisontal udvidelse
Y tast	Styrer vertikal udvidelse
SHIFT/RETURN	Gemmer sprite fra arbejdsområde og returnerer til spørgsmål om SPRITE NUMMER
C tast	Kopierer een sprite til en anden
STOP tast	Slukker for tændt sprite og returnerer til SPRITE NUMMER uden at ændre spriten.
RETURN tast	(ved SPRITE NUMMER?) Går ud af SPRDEF mode

FREMSTILLING AF SPRITES I SPRite DEFinitions MODE

Her forklares den generelle fremgangsmåde til fremstilling af en sprite i SPRDEF mode:

1. Slet arbejdsområdet ved at nedtrykke SHIFT og CLR/HOME samtidigt.
2. Hvis De ønsker en flerfarve sprite nedtrykkes M tasten og en ekstra markør viser sig ved siden af den originale. Grunden til, at der er to markører, er den, at flerfarve mode tæn-

der to pixels i stedet for een i standard sprite mode. Det er derfor, at flerfarve mode kun har halv så stor horisontal opløsning som standard højopløsnings mode.

3. Vælg en farve til Deres sprite. For valg af farverne mellem 1 og 8 holdes CONTROL tasten nedtrykket, mens der trykkes på en tast mellem 1 og 8. Farvenumre mellem 9 og 16 vælges ved at holde C= tasten nedtrykket og samtidig trykke på en af tasterne 1 - 8.
4. De er nu klar til at begynde fremstilling af Deres sprite's omrids. Taltasterne 1 til 4 udfylder spriten og giver den skikkelse. For enkeltfarve sprite bruges 2 tasten til at udfylde en karakterposition indenfor arbejdsområdet. Hvad der er tegnet med 2 tasten kan slettes med et tryk på 1 tasten. Hvis De ønsker at udfylde een karakterposition ad gangen, trykkes på A tasten. De skal nu bevæge spriten manuelt ved anvendelse af markørtasterne. Hvis De ønsker, at markøren automatisk skal flytte mod højre, når De holder den nede, skal De ikke nedtrykke A tasten, da den allerede er sat til automatisk markørbevægelse. Efterhånden som De udfylder karakterpositioner i arbejdsområdet, kan De se de tilsvarende pixels blive tændt i den viste sprite. Ændring af spriten sker i samme øjeblik, som De redigerer i arbejdsområdet.
   I flerfarve mode udfylder 3 tasten to karakterpositioner i arbejdsområdet med flerfarve 1 farven, 4 tasten udfylder to karakterpositioner med flerfarve 2. De kan slukke (give pixelen baggrundsfarven) udfyldte arealer i arbejdsområdet med 1 tasten. I flerfarve mode slukker 1 tasten to karakterpositioner samtidigt.
5. Mens De konstruerer Deres sprite, kan de frit bevæge Dem rundt i arbejdsområdet uden at tænde eller slukke pixels ved at benytte RETURN, HOME og markørtasterne.
6. De kan når som helst udvide Deres sprite i såvel vandret som lodret plan. Lodret udvidelse fås ved at trykke på Y tasten, og vandret udvidelse ved at trykke på X tasten. Trykkes atter på disse taster, vendes tilbage til normal sprite størrelse.
   Når samme tast både kan tænde og slukke samme kontrol, kaldes dette for 'toggling', så X og Y tasterne 'toggler' vertikal og horisontal udvidelse af spriten.

7. Når De er færdig med at danne Deres sprite og tilfreds med dens udseende, gemmes den ved at holde SHIFT nedtrykket og trykke på RETURN. Commodore 128 lagrer den så i et passende sprite lagerområde. Den viste sprite, i skærmens øverste højre hjørne, slukkes og kontrollen overgives igen til SPRITE NUMMER?. Hvis De ønsker at fremstille en ny sprite, skal De indtaste et andet nummer og redigere den nye sprite på samme måde som den første. Ønsker De at se den originale sprite i arbejdsområdet igen, indtastes dennes nummer. Ønsker De at forlade SPRITE DEFINITION mode, skal De blot trykke på RETURN, når der spørges efter spritenummer.
8. En sprite kan kopieres til en anden med C tasten.
9. Hvis De ikke ønsker at lagre Deres sprite, skal De trykke på STOP. Commodore 128 slukker så spriten og vender tilbage til SPRITE NUMMER?.
10. De kommer ud af SPRDEF mode ved at nedtrykke RETURN, når der spørges efter sprite nummer på skærmen. De kan afbryde under følgende betingelser:

    Umiddelbart efter, at De har lagret Deres sprite (SHIFT/RETURN)
    Umiddelbart efter, at De har trykket på STOP.

Når De har fremstillet en sprite og derefter har forladt SPRDEF mode, er Deres sprite lagret i et område i Commodore 128's hukommelse. Da De nu er tilbage under kontrol af BASIC sproget, skal De hente Deres sprite for at kunne se den igen. Dette gøres ved brug af SPRITE kommandoen, som De tidligere har lært om. De har f. eks. fremstillet sprite 1 i SPRDEF mode. For at tænde den i BASIC, farve den blå og udvide den i begge retninger, skal De indtaste denne kommando:

SPRITE 1,1,7,0,1,1,0

Brug nu MOVSPR kommandoen for at bevæge spriten, som følger:

MOVSPR 1,45 # 5

De ved nu alt om SPRDEF mode. Først, opret spriten, gem sprite data og gå fra SPRDEF mode til BASIC. Tænd Deres sprite med SPRITE kommandoen. Flyt den med MOVSPR kommandoen. Når De er færdig med programmering, gemmes Deres sprite data i en binær fil med BSAVE kommandoen, som følger:

BSAVE "filnavn",B0,P3584 TO P4096

Ønsker de igen at bruge sprite data fra diskette, indlæses den tidligere gemte binære fil med BLOAD kommandoen, således:

BLOAD "filnavn" [,B0,P3584]

Angivelsen i skarpe paranteser kan udelades. BLOAD indlæser så data til den adresse, hvorfra de blev gemt.

De kender nu de nye metoder til fremstilling af sprites:
1) SSHAPE, SPRSAV, SPRITE, MOVSPR 2) SPRDEF MODE. Lav forsøg med disse metoder og bliv en mester i sprite animation.

Se LAGRING AF SPRITE DATA I BINÆRE FILER senere i dette afsnit for yderligere information.

SAMMENSÆTNING AF SPRITES

De har lært at fremstille, farvelægge, tænde for og animere sprites. De kunne muligvis komme ud for at ville skabe et billede, som er for detaljeret eller for stort til at kunne være i en enkelt sprite. I så fald kan De sammensætte to eller flere sprites, så billedet bliver større og mere detaljeret, end det kan lade sig gøre med en enkelt sprite. Ved at sammensætte sprites, kan de hver for sig bevæge sig uafhængigt af hinanden. Dette giver Dem meget større kontrol af animering end en enkelt sprite kan give.

Dette afsnit omfatter et eksempel på, hvorledes to sammensatte sprites bruges. Her er den generelle fremgangsmåde (algoritme) for, hvorledes et program med to eller flere sammensatte sprites skal skrives.

1) Tegn et billede på skærmen ved brug af Commodore 128 grafiske instruktioner som DRAW, BOX og PAINT, på nøjagtig samme måde, som De benyttede i racerbilsprogrammet i sidste afsnit. Gør denne gang billedet dobbelt så stort som en enkelt sprite ved at bruge dimensionerne 48 pixels bredt og 21 pixels højt.
2) Brug to SSHAPE instruktioner for at lagre spritene i to separate datastrenge. Positioner de første SSHAPE instruktions koordinater over 24 gange 21 pixel området af den første halvdel af det billede De tegnede. Positioner derefter de næ-

ste SSHAPE koordinater over det andet 24 gange 21 pixel område. Vær sikker på, at De lagrer hver halvdel af billedet i forskellige strenge. For eksempel gemmer den første SSHAPE instruktion den første billedhalvdel i A$, og den næste SSHAPE instruktion gemmer den anden billedhalvdel i B$

3) Overfør billeddataene fra hver datastreng til separate sprites med SPRSAV instruktionen.

4) Tænd for hver sprite med SPRITE instruktionen.

5) Placer spritene, så begyndelsen af den ene sprite starter på den pixel, der ligger op ad den, hvor den første sprite slutter. Det er på denne måde, spritene bindes sammen. Tegn for eksempel et billede 48 gange 21 pixels. Placer den første sprite (1, f.eks.) i lokation 100,100 med denne instruktion:

100 MOVSPR 1,100,100

hvor første tal er spritenummeret, andet tal er den vandrette koordinat og det tredie tal er den lodrette koordinat. Placer den anden sprite 24 pixels til højre for sprite 1 med denne instruktion:

200 MOVSPR 2,124,100

På dette tidspunkt vises de to sprites direkte ved siden af hinanden. De ligner nøjagtigt det billede, De tegnede i programmets begyndelse, ved anvendelse af DRAW, BOX og PAINT instruktionerne.

6) De kan nu igen flytte spritene på den måde, De ønsker, ved igen at bruge MOVSPR instruktionen. De kan flytte dem samme vej samtidigt eller i forskellige retninger. Som De lærte i sidste afsnit, gør MOVSPR instruktionen det muligt at flytte spritene til en bestemt skærmposition, eller til en lokation, som er relativ til spritenes oprindelige placering.

Følgende program er et eksempel på sammensætning af sprites. Programmet foregår i det ydre rum. Det tegner stjerner, en planet og et rumskib som Apollo. Rumskibet tegnes, gemmes derefter i to datastrenge, A$ og B$.
Rumskibets forreste del, kapslen, gemmes i sprite 1. Rumskibets bageste del, raketdelen, gemmes i sprite 2. Rumskibet flyver langsomt hen over skærmen to gange. Da det rejser så langsomt og er meget langt fra Jorden, skal de nødvendigvis lande med raketmotorerne vendt mod Jorden. Efter anden tur over

skærmen affyres raketterne og kapselen vender sikkert tilbage mod Jorden.

Her er programlisten:

```
5  COLOR 4,1:COLOR 0,1:COLOR 1,2:REM VÆLGER SORT KANT OG BGR., HVID FRGR.
10 GRAPHIC 1,1:REM SÆTTER HØJOPLØSNINGS MODE
17 FOR I = 1 TO 40
18 X=INT(RND(1)*320)+1:REM TEGNER STJERNER
19 Y=INT(RND(1)*200)+1:REM TEGNER STJERNER
21 DRAW 1,X,Y:NEXT     :REM TEGNER STJERNER
22 BOX 0,0,5,70,40,,1 :REM SLETTER KASSE
23 BOX 1,1,5,70,40     :REM KASSE I RUMSKIB
24 COLOR 1,8:CIRCLE 1,190,90,35,25:PAINT 1,190,95:REM PLANET
25 CIRCLE1,190,90,65,10:CIRCLE1,190,93,65,10:CIRCLE1,190,95,65,10:COLOR0,1
26 DRAW1,10,17TO 16,17TO 32,10TO 33,20TO 32,30TO 16,23TO 10,23TO 10,17
28 DRAW 1,19,24 TO 20,21 TO 27,25 TO 26,28:REM BUNDVINDUE
35 DRAW 1,20,19 TO 20,17 TO 29,13 TO 30,18 TO 28,23 TO 20,19:REM TOPVINDUE
38 PAINT 1,13,20:REM FARVELÆG RUMSKIB
40 DRAW 1,34,10 TO 36,20 TO 34,30 TO 45,30 TO 46,20 TO 45,10 TO 34,10:REM
SP1
42  DRAW 1,45,10  TO  51,12  TO  57,10  TO  57,17  TO  51,15  TO  46,17:REM ENG1
43 DRAW 1,46,22 TO 51,24 TO 57,22 TO 57,29 TO 51,27 TO 45,29:REM ENG2
44  PAINT  1,40,15:PAINT  1,47,12:PAINT  1,47,26:DRAW  0,45,30  TO  46,20  TO
45,10
45 DRAW 0,34,14 TO 44,14:DRAW 0,34,21TO 44,21:DRAW 0,34,28 TO 44,28
47 SSHAPE A$,10,10,33,32:REM GEMMER SPRITE I A$
48 SSHAPE B$,34,10,57,32:REM GEMMER SPRITE I B$
50 SPRSAV A$,1:REM SPR1 DATA
55 SPRSAV B$,2:REM SPR2 DATA
60 SPRITE 1,1,3,0,0,0,0: REM SÆTTER SPR1 ATTRIBUTER
65 SPRITE 2,1,7,0,0,0,0: REM SÆTTER SPR2 ATTRIBUTER
82 MOVSPR 1,150 ,150:REM SPR1´s ORGINAL POSITION
83 MOVSPR 2,172 ,150:REM SPR2´s ORGINAL POSITION
85 MOVSPR 1,270 £ 5 :REM FLYTTER SPR1 HEN OVER SKÆRMEN
87 MOVSPR 2,270 £ 5 :REM FLYTTER SPR2 HEN OVER SKÆRMEN
90 FOR I=1 TO 5950:NEXT:REM FORSINKELSE
92 MOVSPR 1,150,150:REM SPR1 I POSITION TIL RAKET AFFYRING
93 MOVSPR 2,174,150:REM SPR2 I POSITION TIL RAKET AFFYRING
95 MOVSPR 1,270 £ 10:REM DELER RAKET
96 MOVSPR 2, 90 £ 5 :REM DELER RAKET
97 FOR I=1TO 1200:NEXT:REM FORSINKELSE
98 SPRITE 2,0:REM RAKETTEN (SPR2) SLUKKES
99 FORI=1TO 20500:NEXT:REM FORSINKELSE
100 GRAPHIC 0,1:REM RETUR TIL TEKSTMODE
```

Her er en forklaring til programmet:
Linie 5 farver baggrunden sort og forgrunden hvid.
Linie 10 vælger standard højopløsnings-mode og sletter højopløsningsskærmen.
Linie 23 tegner en kasse til rumskibet i skærmens øverste venstre hjørne.
Linierne 17 t.o.m. 21 tegner stjernerne.
Linie 24 tegner og farvelægger planeten.
Linie 25 tegner cirkler rundt om planeten.
Linie 26 tegner kapselens omrids.
Linie 28 tegner vinduet i rumkapselens bund.
Linie 35 tegner rumkapselens øverste vindue.
Linie 40 tegner omridset af rumskibets raketdel.
Linierne 42 og 43 tegner raketmotorerne bagest i rumskibet.
Linie 44 farvelægger raketmotorerne og tegner et omrids af rakettens bagside i baggrundsfarven.
Linie 45 tegner streger i baggrundsfarve på rumskibets raketdel. Indtil nu, har De kun vist billeder på skærmen. De har ikke brugt nogen sprite instruktioner, så Deres rumskib er endnu ikke nogen sprite.
Linie 47 positionerer SSHAPE koordinaterne over den første halvdel (24 gange 21 pixels), rumskibets kapsel, og lagrer den i en datastreng, A$.
Linie 48 positionerer SSHAPE koordinaterne over den anden halvdel (24 gange 21 pixels) af rumskibet og lagrer den i datastrengen B$.
Linie 50 overfører data fra A$ til sprite 1.
Linie 55 overfører data fra B$ til sprite 2.
Linie 60 tænder sprite 1 og farver den rød.
Linie 65 tænder sprite 2 og farver den blå.
Linie 82 anbringer sprite 1 i koordinaten 150,150.
Linie 83 anbringer sprite 2 24 pixels til højre for sprite 1's startkoordinat. I virkeligheden sammensætter linierne 82 og 83 de to sprites.
Linierne 85 og 87 bevæger de samlede sprites hen over skærmen.
Linie 90 forsinker programmet. Denne gang er forsinkelse nødvendig for at de to sprites kan gennemføre de to rejser hen over skærmen. Hvis man udelader forsinkelsen, har spritene ikke tilstrækkelig tid til at flytte over skærmen.
Linierne 92 og 93 placerer spritene midt på skærmen og gør klar til, at rumskibet kan affyre raketterne.

Linie 95 driver sprite 1, rumkapselen, fremad. Tallet 10 i linie 95 angiver den hastighed, hvormed spriten skal bevæge sig. Farten kan ligge fra 0, som er stilstand, til 15, som er lynhurtigt.
Linie 96 bevæger den udbrændte del af rumskibet baglæns ud af skærmbilledet.
Linie 97 er en tidsforsinkelse, så raketdelen, sprite 2, får tid til at komme væk fra skærmen.
Når sprite 2 er borte, slukkes den af linie 98.
Linie 99 er atter en forsinkelse, så kapselen kan fortsætte bevægelsen tværs over skærmen.
Til sidst returnerer linie 100 til tekstmode.

Som De har set, kan sammensatte sprites være morsommere og mere interessante end enkelte sprites. De hovedpunkter, som skal huskes er:

1) Vær sikker på, at SSHAPE koordinaterne placeres på det rigtige sted på skærmen, så billeddata kan gemmes korrekt.
2) Vær opmærksom på at placere sprite koordinaterne i den rigtige lokation, når de sættes sammen med MOVSPR instruktionen. I dette eksempel blev sprite 2 placeret 24 pixels til højre for sprite 1.

Når De mestrer at sammensætte to sprites, prøv så med flere. Jo flere sprites De sætter sammen, des bedre detaljer og animation vil Deres programmer indeholde. Foretag eksperimenter. De vil snart selv blive fascineret over at lave Deres egne animerede "tegnefilm".

Commodore 128 har yderligere to SPRITE kommandoer, SPRCOLOR og COLLISION, som ikke er gennemgået i dette kapitel. Se i kapitel 5, BASIC 7.0, for at lære om disse kommandoer.

LAGRING AF SPRITE DATA I BINÆRE FILER

OBS! Den følgende forklaring forudsætter nogen kendskab til maskinsprog, hukommelseslokationer, binære filer og objektkode filer.

Commodore 128 har to nye kommandoer, BLOAD og BSAVE, som gør håndtering af spritedata let. "B" i BLOAD og BSAVE står for Binær. BSAVE og BLOAD kommandoerne lagrer og indlæser binære filer til og fra diskette. En binær fil består af enten en del af et maskinsprogsprogram eller en samling af data indenfor et specificeret adresseområde. De kender muligvis SAVE kommandoen i den indbyggede maskinkodemonitor. Hvis De benytter denne SAVE kommando, resulterer dette i, at filen på disketten bliver en binær fil.
En binær fil er lettere at arbejde med end en objektkode fil, da en binær fil kan indlæses uden forudgående formaliteter. En objektkode fil skal indlæses med en 'loader', som i Commodore 64 Assembler Udviklings System; og der benyttes SYSTEM kommandoen (SYS) for at udføre den. Når binære filer skal indlæses, skal man huske, at dette kan ske på een af to måder:

LOAD "binært filnavn",8,1
eller
BLOAD "binært filnavn",B0,Pstart

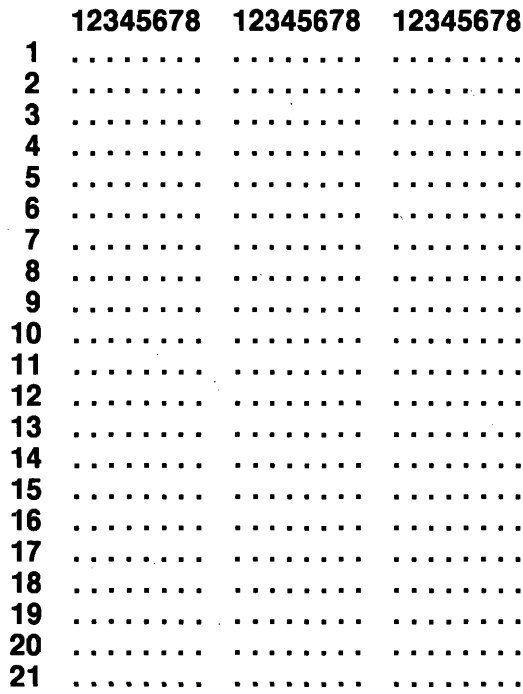
hvor start er 3584, hvis der indlæses spritedata filer.

Med den første metode skal der angives en ",1" sidst i kommandoen, idet computeren ellers behandler det som en BASIC programfil og indlæser den i begyndelsen af BASIC tekstområdet. ",1" gør computeren opmærksom på, at den binære fil skal indlæses til samme område, hvorfra den blev lagret.

De har muligvis spekuleret på, hvad dette har med sprites at gøre. Her er forklaringen. Commodore 128 har en udvalgt del af hukommelsen, rækkende fra decimaladresse 3584 ($0E00) til 4095 ($0FFF), hvori spritedata lagres.
Denne del af hukommelsen optager 512 bytes. Som De ved, er en sprite 24 pixels bred og 21 pixels høj. Hver pixel kræver een bit i hukommelsen.
Hvis bitten i en sprite er slukket (lig med 0), er den tilsvarende pixel på skærmen slukket og antager samme farve som baggrunden. Er en pixel i en sprite tændt (lig med 1), tændes den

tilsvarende pixel på skærmen i forgrundsfarven. Denne kombination af nuller og enere giver det billede, der ses på skærmen.

Da en sprite er 24 gange 21 pixel, og hver pixel kræver en byte af lageret i hukommelsen, bruger een sprite 63 bytes i hukommelsen. Se figur 8 for bedre forståelse af lagerkrav til en sprites data.

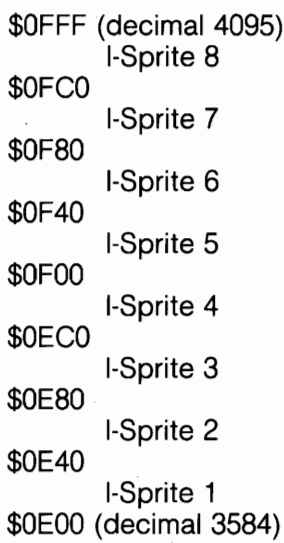


Hver række = 24 bits = 3 bytes

Figur 8

En sprite kræver 63 databytes. Hver spriteblok dannes i virkeligheden af 64 bytes; den ekstra byte benyttes ikke. Da Commodore 128 har 8 sprites og hver af dem består af en blok på 64 bytes, skal computeren bruge 512 bytes for at kunne repræsentere data for samtlige otte sprites.

Området, hvor alle otte spriteblokke findes, begynder i hukommelsesadresse 3584 ($0E00) og slutter i adresse 4095 ($0FFF). Figur 9 indeholder hukommelsens adresseområder, i hvilke hver individuel sprite's data er lagret.



Figur 9

BSAVE

Når De forlader SPRDEF mode, kan De gemme Deres spritedata i binære sprite filer. På denne måde kan De enkelt og nemt genindlæse enhver spritesamling til Commodore 128. Brug nedenstående kommando for at lagre Deres spritedata i en binær fil:

BSAVE "filnavn",B0,P3584 TO P4096

Det binære filnavn er det navn, De giver filen. B0 angiver, at De lagrer spritedata fra bank 0. Parameterne 'P3584 TO P4096' bestemmer, at De lagrer adresseområdet 3584 ($0E00) t.o.m. 4095 ($0FFF), som er det område, hvor alle sprite data findes.

Det er ikke nødvendigt at definere alle sprites, når De gemmer dem. De sprites, De definerer, bliver med BSAVE gemt fra den korrekte spriteblok.

De udefinerede sprites lagres også med BSAVE i den binære fil i den rigtige blok, uden at de derved vedkommer computeren. Det er lettere at lagre samtlige sprite's 512 bytes end, uanset om alle spritene bruges, at lagre den enkelte sprite individuelt.

BLOAD

Hvis De senere ønsker at bruge spritene igen, indlæses samtlige 512 bytes med sprites ved brug af BLOAD, som automatisk placerer dem i området 3584 (#0E00) t.o.m. 4095 ($0FFF). Herunder vises kommandoen:

BLOAD "filnavn" [,B0,P3584]

Der skal bruges samme filnavn som det, der blev brugt, da de originale spritedata blev lagret. B0 står for bank nummer 0 og P3584 angiver startadressen på det område, hvortil spritedata indlæses. De to sidste parametere kan udelades.

****

I dette afsnit har De set, hvormeget de nye Commodore 7.0 BASIC kommandoer kan forenkle den normalt meget indviklede måde at fremstille og animere grafiske figurer på. Næste afsnit gennemgår nogle andre nye BASIC 7.0 kommandoer, som gør det samme vedr. musik og lyd.

AFSNIT 7
LYD OG MUSIK I C128 MODE

INTRODUKTION

Commodore 128 har en af de mest avancerede indbyggede lyd synthesizere, som findes på nogen mikrocomputer. Synthesizeren, kaldet Sound Interface Device (SID), er en chip beregnet til udelukkende at fremstille lyd og musik. SID chippen er i stand til at frembringe tre uafhængige stemmer (lydbilleder) på samme tid. Hver af disse stemmer kan spilles i een af fire lydtyper, kaldet bølgeformer. SID chippen har også programmerbar Attack, Decay, Sustain og Release (ADSR) parametere. Disse parametere fastsætter lydens sammensætning. Tilmed har synthesizeren et filter, som kan bruges til at vælge bestemte lyde, udelukke andre eller til at modificere karakteren af lyd eller lyde. I dette afsnit vil De lære, hvorledes disse parametere kan kontrolleres, så man kan fremstille næsten enhver tænkelig lyd.

For at gøre det let for Dem at vælge og bruge SID chippens mange muligheder, har Commodore udviklet nye og kraftfulde BASIC musik instruktioner.

Her er de nye lyd og musik instruktioner, som Commodore 128 indeholder:

SOUND
ENVELOPE
VOL
TEMPO
PLAY
FILTER

Dette afsnit forklarer disse lydinstruktioner, en ad gangen, i løbet af konstruktionen af et musikalsk programeksempel. Når De er færdig med afsnittet, kender De de elementer, der indgår i et musikprogram. De vil være i stand til at bygge videre på eksemplet og skrive programmer, som spiller indviklede musikalske kompositioner. De vil eventuelt kunne programmere Deres egne kompositioner, fremstille Deres egne lydeffekter og spille værker af de store klassiske mestre som Beethoven og nyere kunstnere som The Beatles. De kan endog føje computerfremstillet musik til Deres grafiske programmer og på den måde fremstille Deres egne 'videofilm'.

SOUND INSTRUKTIONEN

SOUND instruktionen er primært beregnet til let og hurtig fremstilling af lydeffekter til Deres programmer. De vil, senere i dette afsnit, lære en mere indviklet måde til at spille komplette musikalske arrangementer på, ved brug af andre lyd instruktioner.

SOUND instruktionens format ser således ud:

SOUND VC,FREQ,DUR [,DIR [,MIN [,SV [,WF [,PW]]]]]

Her er parameternes betydning:

VC - Valg af stemme; VoiCe 1, 2 eller 3
FREQ - Indstilling af frekvensniveau (0-65535)
DUR - Indstilling af lydens varighed (i 60.dele af et sekund)
DIR - Indstilling af den retning, hvori lyden øges eller nedsættes
  0 = Forøgelse af frekensen opadgående
  1 = Formindskelse af frekvensen nedadgående
  2 = Frekvensen svinger op og ned
MIN - Valg af MINimum frekvensen (0-65535), hvis DIR (sweep) angives
SV - Valg af trinværdi for sweep (lydbevægelse) (0-32767)
WF - Valg af bølgeform (0-3)
  0 = trekant
  1 = savtak
  2 = variabel puls
  3 = Hvid støj
PW - Indstilling af pulsbredde, bredden af den variable puls bølgeform.

Vær opmærksom på, at DIR, MIN, SV, WF og PW parameterne er valgfri.

Den første parameter (VC) i SOUND instruktionen vælger, hvilken stemme der vil blive spillet. Den anden parameter (FREQ) fastlægger lydens frekvens og kan ligge i området fra 0 t.o.m. 65535. Tredie oplysning (DUR) angiver varigheden af den tid, hvori lyden spilles. Varigheden måles i 60.dele af et sekund. Hvis De derfor ønsker, at lyden skal vare et sekund, skal DUR sættes til 60, da 60 gange 1/60 er 1. Skal lyden spilles i 2 sekunder, angives DUR til 120. For at spille i 10 sekunder, sættes DUR til 600, o.s.v.

Den fjerde parameter (DIR) vælger i hvilken retning, lydens frekvens skal øges eller formindskes. Dette kaldes også 'sweep', som vi vil bruge herefter. Den femte angivelse (MIN) angiver minimumsfrekvensen, ved hvilken sweep begynder. Sjette angivelse (SV) er sweep's trinværdi. Den svarer til trinværdien i en FOR...NEXT løkke. Hvis DIR, MIN og Sv værdierne angives i SOUND kommandoen, spilles lydens først på det originale niveau, som er specificeret med FREQ parameteren. Derefter 'sweeper' synthesizeren igennem og spiller hvert niveau af hele frekvens-værdiområdet, med udgangspunkt i MIN frekvensen. Sweep øges eller formindskes af trinværdien (SV) i sammenhæng med den retning, der er angivet med DIR parameteren og frekvensen spilles i det nye niveau.

Den syvende parameter (WF) i SOUND kommandoen vælger bølgeform for lyden.
(Bølgeformer forklares i detaljer i afsnittet "Avanceret Lyd og Musik i C128 Mode).

Den sidste angivelse i SOUND kommandoen er bestemmende for pulsens bredde.
(Se Avanceret Lyd for gennemgang af puls bredde bølgeform).

FREMSTILLING AF ET LYDPROGRAM MED SOUND

Det er nu tid til at skrive Deres første lydprogram. Her er et eksempel på SOUND kommandoen:

```
10 VOL 5
20 SOUND 1,4096,60
```

Udfør programmet med RUN. Commodore 128 spiller et kort, højtlydende beep. Før SOUND instruktionen kan spilles, skal

styrken indstilles, så linie 10 indstiller lydchippens VOLumen. Linie 20 spiller stemme 1 ved frekvensen 4096 med en varighed på 1 sekund (60 * 1/60). Prøv at ændre frekvensen med denne instruktion:

30 SOUND 1,8192,60

og læg mærke til, at linie 30 spiller en højere tone end linie 20. Dette viser sammenhængen mellem frekvensangivelsen og lydens aktuelle frekvens. Efterhånden som De forøger frekvensangivelsen, øger Commodore 128 tonens lydbillede. Prøv nu denne instruktion:

40 SOUND 1,0,60

Dette viser, at en FREQ værdi på 0 spiller den laveste frekvens (hvilket er så lavt, at det ikke kan høres). En FREQ angivelse på 65535 spiller den højst mulige frekvens.

Prøv nu at anbringe sound instruktionen i en FOR...NEXT løkke. Dette sætter Dem i stand til at gennemspille hele frekvensområdet i denne løkke.
Føj disse instruktioner til Deres program:

50 FOR I = 1 TO 65535 STEP 100
60 SOUND 1,I,1
70 NEXT

Dette programsegment spiller den variable puls bølgeform i frekvensområdet fra 1 t.o.m. 65535 i trin på 100, fra laveste til højeste frekvens. Angiver De ikke bølgeformen, vil Commodore 128 vælge standardværdien 2, variabel puls bølgeform.

Ret nu bølgeformen med følgende programlinie og prøv programmet igen.

60 SOUND 1,I,1,0,0,0,0,0

Nu spiller programmet stemme 1, bruger trekant bølgeform, i frekvensområdet mellem 1 og 65535 med trin på 100. Denne lyd er en typisk lydeffekt i populære spillemaskiner. Prøv bølgeform 1, savtak bølgeform og se, hvordan det lyder med denne linie:

60 SOUND 1,I,1,0,0,0,1,0

Savtak bølgeformen lyder næsten som trekant bølgeformen,

skønt den summer mindre. Prøv nu til sidst hvid støj bølgeform. Udskift linie 60 med:

60 SOUND 1,l,1,0,0,0,3,0

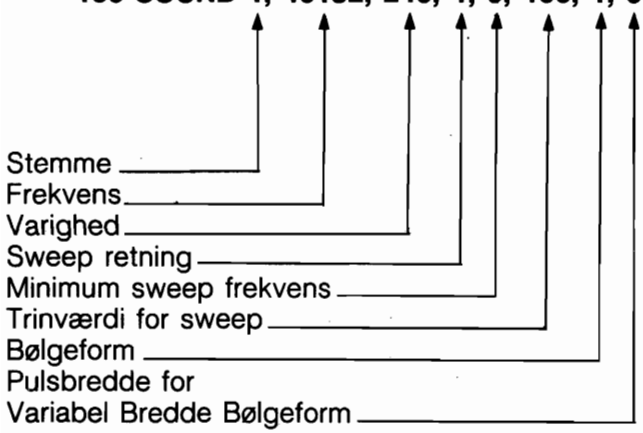
Nu gennemspiller programløkken 'hvid støj generatoren' for hele frekvensområdet. Først høres en svag, rumlende lyd. Efterhånden som frekvensen stiger i løkken, øges lydbilledet og det lyder nærmest, som ved en raketaffyring.

Bemærk, at vi indtil nu ikke har specificeret alle SOUND instruktionens parametere. Se for eksempel på linie 60:

60 SOUND 1,l,1,0,0,0,3,0

De tre nuller efter 1,l,1 henviser til sweep parameterne i SOUND instruktionen. Da ingen af disse parametere er angivet, sweeper lyden ikke. Prøv at føje nedenstående til Deres program:

100 SOUND 1, 49152, 240, 1, 0, 100, 1, 0



Linie 100 begynder sweep frekvensen ved 49152 og formindsker sweep med 100 i nedadgående retning, til minimum sweep frekvensen på 0 nås. Stemme 1, som bruger savtak bølgeform (#1), spiller hver lyd i fire sekunder (240 * 1/60 sek.) Linie 100 lyder i retning af et bombenedkast, som bruges i mange spillemaskiner.

Prøv at forandre nogle af parameterne i linie 100. Ret for eksempel sweep's retning til 2 (svingende) og ret sweep's minimumsfrekvens til 32768; forøg trinværdien til 3000. Den nye SOUND kommando er:

110 SOUND 1,49152,240,2,32768,3000,1

Linie 110 frembringer en sirenelyd, som var politiet lige i hælene på Dem.
En mere behagelig lyd fås med dette:

110 SOUND 1,65535,250,0,32768,3000,2,2600

Dette skulle minde Dem om et populært rumalder TV-show, hvor mandskabet i rumskibet affyrer deres fremtidsvåben mod indtrængende fjender.
Indtil nu har De blot programmeret med een stemme. Med SOUND instruktionen kan De frembringe interessante lydeffekter ved at bruge op til tre stemmer. Eksperimenter og fremstil et program, som bruger alle tre stemmer.
Her følger er programeksempel, som vil hjælpe Dem til at forstå, hvorledes man programmerer Commodore 128's synthesizer chip. Når programmet køres, spørges efter hver parameter, og derefter spilles lyden. Indtast programmet på Deres computer og kør det med RUN.

```
10 PRINT:PRINT:PRINT:PRINT"!SHIFT/CLR!   LYDGENERATOR":PRINT:PRINT:PRINT
20 PRINT"    ANGIV LYDPARAMETERE FOR LYDEN":PRINT:PRINT
30 INPUT "STEMME (1 - 3)";V
40 INPUT "FREKVENS (0 - 65535)";F
50 INPUT "VARIGHED (0 - 32767)";D:PRINT
60 INPUT "ØNSKES FLERE PARAMETERE ANGIVET? J/N";B$:PRINT
70 IF B$ = "N" THEN 130
80 INPUT "SWEEP RETNING 0=OP, 1=NED, 2=SVINGENDE";DIR
90 INPUT "MINIMUM SWEEP FREKVENS (0 - 65535)";M
100 INPUT "SWEEP TRINVÆRDI (0 - 32767)";S
110 INPUT "BØLGEFORM (0=TRE, 1=SAV, 2=VAR PUL, 3=STØJ)";W
120 IF W = 2 THEN INPUT "PULSBREDDE (0 - 4095)";P
130 SOUND V, F, D, DIR, M, S, W, P
140 INPUT "VIL DE HØRE LYDEN IGEN? J/N";A$
150 IF A$ = "J" THEN 130
160 GOTO 10
```

Her er en hurtig gennemgang af programmet.
Linierne 10 og 20 skriver meddelelserne på skærmen.
Linierne 30 til 50 modtager besked om stemme, frekvens og varighed.
Linie 60 spørger, om De vil angive yderligere parametere, som sweep og bølgeform. Hvis De ikke ønsker at specificere disse, trykkes på "N" og programmet går til linie 130 og spiller lyden. Ønsker De at angive de ekstra SOUND parametere tastes "J" og programmet fortsætter med linie 80.
Linierne 80 til 110 specificerer sweep retning, minimum sweep frekvens, sweep trinværdi og bølgeform.
Linie 120 modtager bredden på variabel puls bølgeform, men kun hvis bølgeform 2 (variabel puls) er blevet valgt.
Linie 130 spiller lyden i overensstemmelse med de parametere, som tidligere i programmet blev angivet.
Linie 140 spørger om De vil høre lyden igen. Ønsker De det, skal De taste "J", i modsat fald tastes "N".
Linie 150 kontrollerer om der blev trykket på "J"-tasten. Er det tilfældet, overgives kontrollen til linie 130 og lydens spilles igen. Trykkede De ikke på "J"-tasten, fortsætter programmet med Linie 160, som giver kontrollen til linie 10, hvorved programmet gentages.
Lydgeneratorprogrammet standses ved samtidig at nedtrykke RUN/STOP og RESTORE.

RANDOM (TILFÆLDIGE) LYDE

Det følgende program frembringer tilfældigt udvalgte lyde ved brug af RND funktionen. Hver SOUND parameter beregnes tilfældigt. Indtast programmet på Deres computer. SAVE og RUN det. Programmet illustrerer, hvorledes mange tusinde lyde kan frembringes ved at angive forskellige SOUND parameter kombinationer. Her er programlisten:

```
10 PRINT "VC FREQ DIR MIN SV WF PW "
20 PRINT "-------------------------"
30 V=INT(RND(1)*3)+1:REM STEMME
40 F=INT(RND(1)*65535):REM FREKVENS
50 D=INT(RND(1)*32767):REM VARIGHED
60 DIR=INT(RND(1)*3):REM TRINRETNING
70 M=INT(RND(1*65535):REM MIN FREKVENS
80 S=INT(RND(1)*32767):REM TRINVÆRDI
90 W=INT(RND(1)*4):REM BØLGEFORM
100 P=INT(RND(1)*4095:REM PULS W
110 PRINTV; F;DIR;M;S;W;P:PRINT:PRINT
120 SOUND V, F, D, DIR, M, S, W, P
130 SLEEP 4
140 SOUND V, 0, 0, DIR, 0, 0, W, P
150 GOTO 10
```

Linierne 10 og 20 udskriver parameterkolonne overskrift og understregning. Linierne 30 t.o.m. 100 beregner hver SOUND parameter indenfor det specificerede område. For eksempel beregner linie 30 stemmenummeret således:

30 V = INT(RND(1)*3)+1

Notationen RND(1) angiver det tilfældige tals startværdi. Dette er det basale tal, som beregnes af computeren. Ettallet fortæller computeren, at den skal generere en ny værdi, hvergang kommandoen mødes. Da Commodore 128 råder over tre stemmer, giver notationen *3 computeren besked på at generere et tilfældigt tal fra 0 til 3. Læg imidlertid mærke til, at der ikke findes nogen stemme 0, så angivelsen +1 i linie 30 gør computeren opmærksom på, at det genererede tal skal ligge i intervallet 1 - 3. Proceduren i forbindelse med generering af et tilfældigt tal i det angivne område består i at multiplicere det givne tilfældige tal med parameterens maksimumsværdi (i dette tilfælde 3). Er minimumsværdien større end 0, skal der til det tilfældige tal adderes et tal, som vil angive minimumsværdien af det talområde, i hvilket tal ønskes genereret (i dette tilfælde 1). For eksempel frembringer linie 40 et tilfældigt tal i området 0 - 65535. Da minimumsværdien i dette tilfælde er 0, er det ikke nødvendigt at addere en værdi til det genererede tilfældige tal.

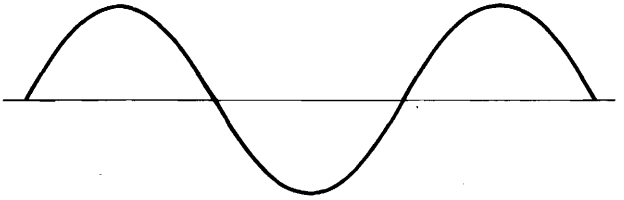
Linie 110 udskriver parameternes værdier. Linie 120 afspiller den lyd, som er specificeret ved anvendelse af de tilfældige tal, som er genereret i linierne 30 t.o.m. 100. Linie 130 forsinker programudførelsen i 4 sekunder, mens lyden spilles. Linie 140 afbryder lyden efter de 4 sekunders forløb. Alle lyde, som frembringes af dette program afspilles lige længe, da de alle afbrydes efter 4 sekunder med linie 140. Til sidst returnerer linie 150 kontrollen til linie 10 og programmet gentages, til der samtidig trykkes på RUN/STOP og RESTORE.

Indtil nu har De eksperimenteret med programeksempler, der udelukkende har anvendt SOUND instruktionen. Skønt De kan bruge SOUND instruktionen til at spille musikstykker, er den bedst egnet til hurtige og nemme lydeffekter som det, der vises i programmet med hundeslagsmål. Commodore 128 har andre instruktioner, som er udviklet specielt for afspilning af sange. Følgende afsnit beskriver de avancerede lyd og musik instruktioner, som sætter Dem i stand til at spille indviklede musikstykker og arrangementer med Deres Commodore 128 synthesizer.

AVANCERET LYD OG MUSIK I C128 MODE

Kortfattet baggrundsviden: LYDENS KARAKTERISTIKA

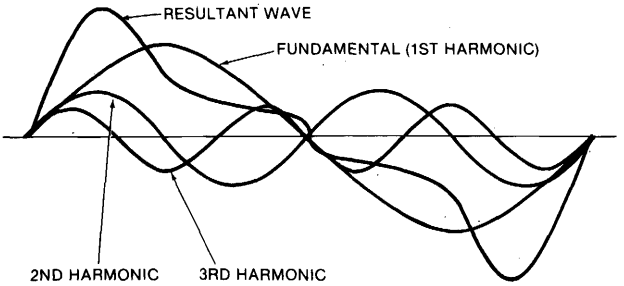
Hver eneste lyd, De hører, er i virkeligheden en lydbølge, som farer gennem luften. Som alle bølger, kan en lydbølge (sinus) repræsenteres grafisk og matematisk (se figur 1).



Figur 1 - Sinuskurve

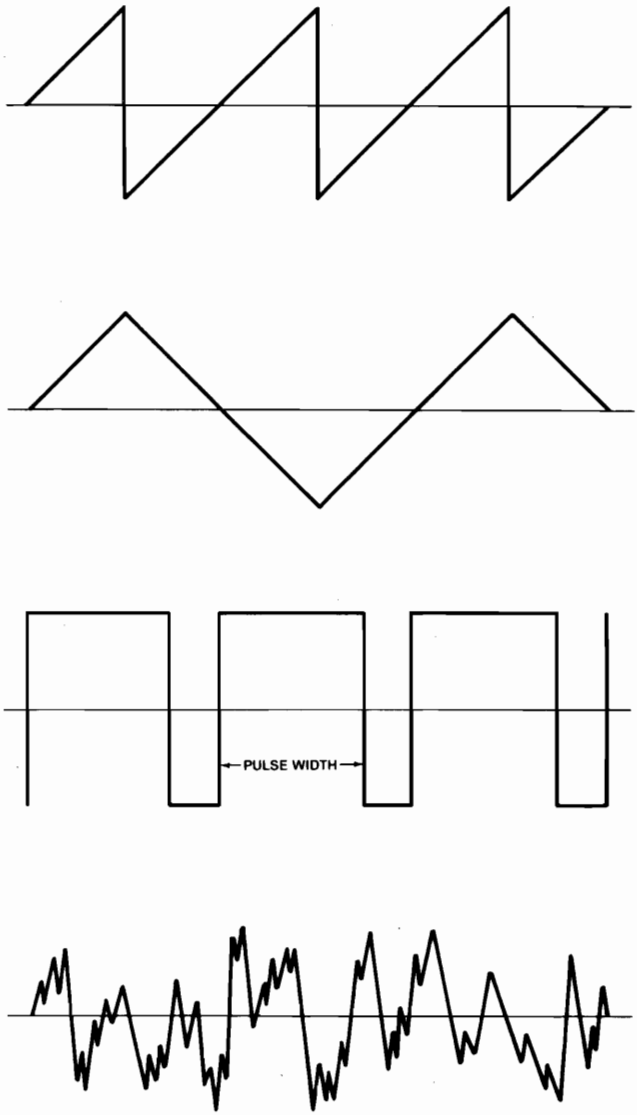
Lydbølgen bevæger (oscillerer) sig indenfor et bestemt område (frekvens), som er afgørende for tonens højde.

Lyden består også af harmonier, som ledsager mange af lydens eller tonens overområder. Det er kombinationen af disse harmoniske lydbølger, som giver noderne deres kvalitet, kaldet 'timbre'. Figur 2 viser sammenhængen mellem basislydens frekvenser og harmonierne.



Figur 2 - Frekvenser og harmonier

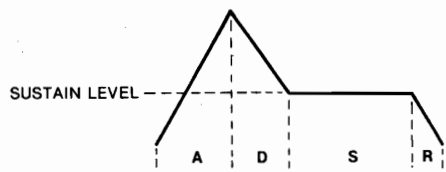
En musikalsk tones timbre (d.v.s. den måde, hvorpå en tone lyder) afgøres af tonens bølgeform. Commodore 128 kan frembringe fire bølgeformstyper:
trekant, savtak, variabel puls og støj. Se figur 3 for grafisk repræsentation af disse fire bølgeformer.



Figur 3 - Lydbølgeforms typer

ENVELOPE instruktionen (envelope = indhyllingskurve).

En lyds lydstyrke svinger i løbet af tonens varighed, fra De hører den første gang til den ikke længere er hørbar. Sådanne volumenkvaliteter kaldes Attack, Decay, Sustain og Release (ADSR). **Attack** er den tid, indenfor hvilken en tone når sin højeste styrke. **Decay** er tiden indenfor hvilken en tone aftager i styrke fra højeste til vedvarende niveau (sustain). **Sustain** er det niveau, hvori en tone spilles vedvarende. **Release** er den tid indenfor hvilken en tone aftager i styrke fra sustain niveauet til 0 volumen. ENVELOPE generatoren styrer lydens ADSR parametere. Se figur 4 for grafisk repræsentation af ADSR. Commodore 128 kan indstille hver ADSR parameter til 16 forskellige tider. Dette giver Dem absolut fleksibilitet overfor ENVELOPE generatoren og styrkemulighederne, når lyden afspilles.



Figur 4 - ADSR faser

En af de kraftigste Commodore 128 sound instruktioner - den, som styrer ADSR og bølgeformen - er ENVELOPE instruktionen. ENVELOPE instruktionen sætter de forskellige kontroller i synthesizer chippen, og gør derved hver lyd unik. ENVELOPE giver Dem styrke til at manipulere med SID synthesizeren. Med ENVELOPE kan De vælge særlige ADSR indstillinger og vælge en bølgeform for Deres egen musik og egne lydeffekter. ENVELOPE instruktionens format er som følger:

ENVELOPE e[,a[,d[,s[,r,[,wf[,Pw]]]]]]

Her er bogstavernes betydning:

e - envelope nummer (0-9)
a - attack tid (0-15)
d - decay tid (0-15)
s - sustain niveau (0-15)
r - release tid (0-15)
wf - bølgeform
0 = trekant
1 = savtak
2 = puls (firkant)
3 = støj
4 = ring modulation
pw - puls bredde (0-4095)

Her følger definition på de parametere, som endnu ikke er forklaret:

ENVELOPE
En musikalsk nodes karakteristika bestemmes af nodens bølgeform og attack, decay, sustain og release indstillingerne. For eksempel har envelope for en guitar node en anderledes ADSR og bølgeform end en fløjtes.

BØLGEFORM
En lydbølges type dannes af kombinationen af en tones ledsagende musikalske harmonier. De ledsagende harmoniske lyde er baseret på tonens overliggende frekvenser. Tonens kvaliteter, som frembringes af hver bølgeform, er tydeligt forskellige fra hinanden, og kan ses grafisk afbilledet i figur 3.

PULSBREDDE
Pulsbredden er tidsintervallet mellem tonernes positive og negative andel og frembringes af puls bølgeformen.

De kan nu realisere ENVELOPE instruktionen fuldt ud. Den styrer de fleste af de musikalske kvaliteter af de noder, som spilles af lyd synthesizeren.
Commodore 128 har 10 forud definerede envelope-indstillinger, som passer til 10 forskellige musikinstrumenter. Ved at bruge disse prædefinerede envelope-indstillinger er det unødvendigt for Dem at specificere ADSR parametere og indstillinger for bølgeform og pulsbredde - dette er allerede gjort for Dem. Alt hvad

De skal gøre er at angive envelope nummeret. Resten af parameterne vælges automatisk af Commodore 128. Her er en liste over de prædefinerede envelope-indstillinger for forskellige musikinstrumenttyper:

Envelope nummer	Instrument	Attack	Decay	Sustain	Release	Bølge	Bredde
0	Piano	0	9	0	0	2	1536
1	Harmonika	12	0	12	0	1	
2	Calliope	0	0	25	0	0	
3	Tromme	0	5	5	0	3	
4	Fløjte	9	4	4	0	0	
5	Guitar	0	9	2	1	1	
6	Spinet	0	9	0	0	2	512
7	Orgel	0	9	9	0	2	2048
8	Trompet	8	9	4	1	2	512
9	Xylofon	0	9	0	0	0	

Figur 5 - standardværdi-parametere for ENVELOPE instruktionen

Nu da De har nogen baggrund vedr. ENVELOPE instruktionen, kan de prøve et andet eksempel ved at indtaste denne instruktion på Deres C128:

```
10 ENVELOPE 0,5,9,2,2,2,1700
```

ENVELOPE instruktionen omdefinerer standard envelope piano (0) til følgende: Attack = 5, Decay = 9, Sustain = 2, Release = 2, Bølgeform forbliver 2 og pulsbredden på den variable puls bølgeform er nu 1700. Piano envelope vil ikke bruge disse karakteristika før det er valgt med en PLAY instruktion, hvilket De vil lære senere i dette afsnit.

Næste trin i musikprogrammering består i at indstille chippens volumen som følger:

```
20 VOL 8
```

VOL instruktionen indstiller lydchippens styrke mellem 0 og 15, hvor 15 er maksimum og 0 er nulstillet.

TEMPO Instruktionen

Næste trin i Commodore 128 musikprogrammering er at kontrollere tempoet, eller Deres melodis hastighed. TEMPO instruktionen udfører dette for Dem.
Her er formatet:

```
TEMPO n
```

hvor n er et ciffer mellem 0 og 255 (255 er det hurtigste). Hvis De ikke angiver tempoet i Deres program, indstiller Commodore 128 automatisk tempoet til 8. Føj denne instruktion til Deres programeksempel:

30 TEMPO 10

PLAY instruktionen

Det er nu på tide at lære, hvorledes De afspiller noderne i Deres melodi.
De er allerede klar over, hvorledes PRINT instruktionen virker. De spiller noderne i Deres melodi på samme måde, som De PRINTer en tekststreng på skærmen, borset fra, at De bruger PLAY i stedet for PRINT. PRINT udskriver tekst, PLAY udsender musikalske toner.

Her er PLAY instruktionens generelle format:

PLAY " streng indeholdende synthesizer kontrolkarakterer og musikalske noder"

Det totale antal karakterer (incl. musikalske noder og synthesizer kontrolkarakterer), der kan lægges i en PLAY kommando, er 255. Da dette imidlertid overstiger det tilladte karakterantal (160) for en enkelt BASIC 7.0 programlinie, er De nødt til at koordinere (d.v.s. sammensætte) mindst to strenge for at nå denne længde. De kan undgå brugen af sammensatte linier ved at sikre Dem, at Deres PLAY kommandoer aldrig overstiger 160 tegn, een programlinie. Dette modsvarer fire skærmlinier i 40 kolonners format eller to skærmlinier i 80 kolonners format. De vil derved lave PLAY kommandolinier, som er lettere at forstå og anvende.

For at spille musikalske noder, skal bogstavet for den node, De ønsker at spille, omgives af anførelsestegn. Her er et eksempel på, hvorledes De kan spille den musikalske skala:

40 PLAY "C D E F G A B"

Dette afspiller de anførte noder i piano envelope, som er envelope 0. Efter hver gang, De udfører dette programeksempel med RUN, skal De nedtrykke RUN/STOP og RESTORE for at nulstille synthesizer chippen.

De har mulighed for at angive nodens varighed ved, i anførelsestegn, at foranstille et af følgende bogstaver:

W - helnode
H - halvnode
Q - fjerdedelsnode
I - ottendedelsnode
S - sekstendedelnode

Angives varighed ikke, vælges standardværdien W for helnode.

De kan opnå pause i afspilningen med følgende PLAY streng:

R - Rest (hvil)

De kan instruere computeren om at vente, til alle stemmer, som spiller lige nu, når slutningen af en takt ved at medtage følgende i anførelsestegn:

M - vent for slutningen på takten

Commodore 128 har endvidere synthesizer kontrolkarakterer, som De kan medtage indenfor anførelsestegn i en PLAY streng. Dette giver Dem fuldstændig kontrol over alle noder, og tillader Dem at ændre styringen af synthisizeren med en nodestreng. Efterfølg kontrolkarakteren med et tal, som ligger indenfor denne karakters gyldige område. Kontrolkaraktererne og disses områdeværdier fremgår af figur 6. Det "n", som følger kontrolkarakteren, er det tal, De vælger fra det angivne område.

Kontrolkarakter	Beskrivelse	Område	Standardværdi
Vn	Stemme	1-3	1
On	Oktav	0-6	4
Tn	Envelope	0-9	0
Un	Styrke	0-15	9
Xn	Filter	0=fra 1=til	0

Figur 6 - Lydsynthesizerens kontrolkarakterer

Skønt SID chippen kan godtage disse kontrolkarakterer i vilkårlig rækkefølge, placer så, for at få det bedste resultat, kontrolkaraktererne i Deres streng i den rækkefølge, som de er vist i i figur 6.
De behøver ikke nødvendigvis at angive nogen af kontrolkarak-

tererne, men De bør maksimere synthesizerens styrke. Commodore 128 indstiller automatisk synthesizerens kontroller til standardværdierne, som er vist i figur 6. Hvis De ikke angiver specielle kontrolkarakterer, kan SID chippen kun spille een envelope, een stemme og een oktav uden nogen form for filtrering. Eksperimenter lidt med kontrolkaraktererne for at øve Dem på at opnå størst mulig kontrol over de noder, der er i Deres PLAY streng.

Hvis De specificerer en ENVELOPE instruktion og i stedet for standardværdierne vælger Deres egne, skal antallet af envelope kontrolkarakterer i Deres PLAY streng stemme overens med envelope nummeret i Deres ENVELOPE instruktion, for at kunne bruge de parametere, De har indlagt. Hvis De blot ønsker at PLAYe med standardværdierne, behøver De overhovedet ikke at specificere ENVELOPE instruktionen. I så fald skal De simpelthen vælge et envelope nummer i PLAY instruktionen med (T) kontrolkarakteren.

Her er et eksempel, hvor PLAY instruktionen bruger SID chip kontrolkarakterer indenfor en streng. Føj denne linie til Deres program og læg mærke til forskellen mellem denne instruktion og PLAY instruktionen i linie 40.

50 PLAY "V2 05 T7 U5 X0 C D E F G A B"

Denne instruktion spiller samme noder som i linie 40, men der er valgt stemme 2, noderne spilles en oktav højere (5) end linie 40, styrkeindstillingen er nedsat til 5 og FILTER er specificeret som slukket. Lad filteret være slået fra. Når De i næste afsnit lærer om filtrering, kan De vende tilbage og tænde for filteret for at se, hvorledes det påvirker afspilningen af noderne. Læg mærke til, at linie 50 vælger et nyt instrument, orgel envelope, med T7 kontrolkarakteren. Nu spiller Deres program to forskellige instrumenter i to af de uafhængige stemmer. Tilføj denne instruktion for at spille den tredie stemme:

60 PLAY "V3 06 U7 T6 X0 C D E F G A B"

Her skal forklares, hvordan linie 60 styrer synthesizeren. V3 vælger stemme 3, O6 placerer stemme tre en oktav højere (6) end stemme to. T6 vælger spinet envelope, U7 sætter volumen til 7 og X0 bevarer filteret fraslået for alle tre stemmer. Nu spiller Deres program alle tre stemmer, hver en oktav højere end den anden, på tre forskellige instrumenter, piano, orgel og spinet.

Indtil nu har Deres PLAY instruktioner kun spillet helnoder. Tilføj noder af forskellig varighed ved at indsætte varigheds kontrolkarakterer i Deres PLAY streng som følger:

70 PLAY "V2 O6 T0 U7 X0 H C D Q E F I G A S B"

Linie 70 spiller stemme 2 i oktav 6 med styrken 7, med den omdefinerede piano envelope (0) og fraslået filter. Denne instruktion spiller noderne C og D som halvnoder, E og F som fjerdedelsnoder, G og A som ottendelsnoder og B som en sekstendedelsnode. Læg mærke til forskellen mellem piano envelope i linie 40 og den omdefinerede piano envelope i linie 70. Linie 40 lyder mere som et piano end linie 70.

De kan spille noder i dur, moll eller forlænget ved at foranstille følgende karakterer i anførelsestegn:

 # - kryds for...
$ - b for...
. - forlænget

En forlænget node spiller halvanden gang længere end en almindelig node.

Prøv nu at tilføje kryds-, b- og forlængede noder med denne instruktion:

80 PLAY "V1 O4 T4 U8 X0 .H C D Q
 # E F I $ G A .S # B"

Linie 80 spiller stemme 1 i oktav 4 med styrken 8 med fløjte envelope og filteret fraslået. Den spiller endvidere C og D som forlængede halvnoder, E og F som fjerdedelsnoder med kryds for, G og A som ottendedelsnoder med kryds for og B som en forlænget sekstendedels node med kryds. De kan indsætte pauser (R) et hvilket som helst sted i Deres PLAY streng. Mellemrummene i de nye PLAY instruktions eksempler er ikke nødvendige, men er kun brugt for oversigtens skyld.

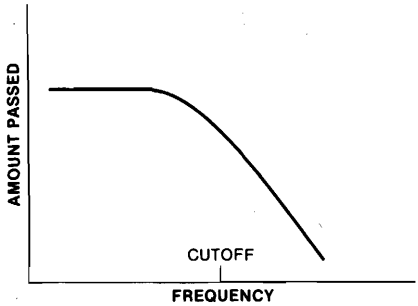
Indtil nu har Deres eksempler haft filteret slået fra i lydsynthesizeren og har endnu ikke vist deres sande kraft. Da De nu har hørt om de fleste lyd og musikinstruktioner og om SID kontrolkaraktererne, bør De gå videre med næste afsnit, for at lære hvorledes Deres musiks kvalitet kan forbedres betydeligt med FILTER instruktionen.

SID FILTERET

Når først De har valgt ENVELOPE, ADSR, VOLumen og TEMPO, bør De bruge FILTER for at gøre Deres synthesizer lyde perfekte. FILTER instruktionen vil, i Deres programmer, skulle stå foran PLAY instruktionen. De bør først være fortrolig med fremstilling af lyd og først senere bekymre Dem om filtrering. Da SID chippen kun har eet filter, benyttes det i forbindelse med alle tre stemmer. Deres computermelodier kan spille uden filtrering, men for at få fuld fornøjelse af deres musik synthesizer, skal De bruge FILTER instruktionen for at forbedre lydens skarphed og kvalitet.

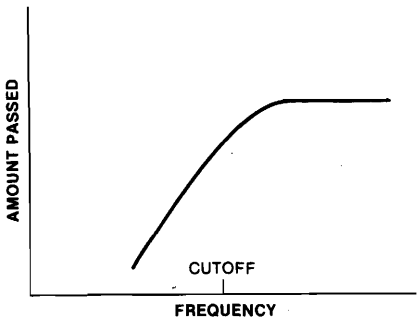
I dette afsnits første del, LYDENS KARAKTERISTIKA, defineredes en lyd som en bølge, der går gennem luften med en bestemt hastighed. Hastigheden, hvormed en lydbølge oscillerer kaldes bølgens frekvens. Husk, at en lydbølge er sammensat af en generel frekvens med ledsagende harmonier, som er et multiplum af den generelle frekvens. Se figur 2. De ledsagende harmonier giver lydens dens timbre, de lydkvaliteter, som bestemmes af bølgeformen. SID chippens indbyggede filter sætter Dem i stand til at fremhæve eller fjerne en bølgeforms harmonier og derved ændre dens timbre.

SID chip filtrene har tre egenskaber: low-pass, band-pass og high-pass filtrering. Disse filtreringsmetoder kan sættes sammen, hvilket betyder, at De kan bruge mere end et filter på en gang. Dette gennemgås i næste afsnit. Low-pass bortfiltrerer frekvenser, som ligger over en bestemt frekvens, som De specificerer og som kaldes cut-off frekvensen. Cut-off frekvens er adskillelseslinien, som markerer bundgrænsen af hvilket frekvensniveau, der vil blive spillet og hvilket, der ikke vil blive det. I low-pass filtrering spiller SID chippen alle frekvenser, som ligger under cutoff frekvensen og bortfiltrerer alle frekvenser, der ligger over denne.
Som navnet antyder, tillades lave frekvenser at passere gennem filteret, mens høje ikke må. Low-pass filteret frembringer fulde, solide lyde. Se figur 7-7. (cut-off frekvensen defineres som den frekvens hvor lydstyrken er faldet 3dB).



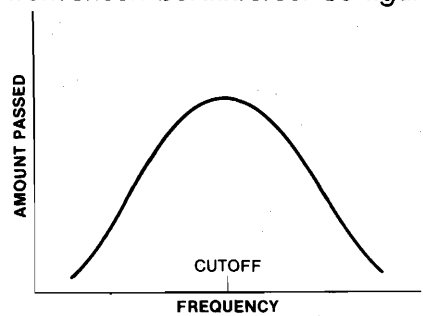
Figur 7 - Low-pass filter

Modsat, tillader high-pass filteret, at alle frekvenser over cut-off frekvensen passerer gennem chippen. Alle under frekvensen udelukkes. Se figur 8. High-pass filteret frembringer tynde, hule lyde.



Figur 8 - High-pass filter

Band-pass filteret tillader et område af frekvenser, som ligger noget over og under cut-off frekvensen, at passere gennem SID chippen. Alle andre frekvenser over og under det bånd, som omgiver cut-off frekvensen bortfiltreres. Se figur 9.



Figur 9 - band-pass filter

FILTER instruktion

Filter instruktionen angiver cut-off frekvensen, den filtertype, som skal bruges og resonans. Resonansen er lydbølge frekvens toppunkt, hvor den nærmer sig cut-off frekvensen. Resonansen er bestemmende for en lyds klarhed og tydelighed; jo højere resonans, des skarpere lyd.

Dette er FILTER instruktionens format:

FILTER cf, lp, bp, hp, res

Her er parameternes betydning:

cf - cut-off frekvens (0-2047)
lp - low-pass filter 0 = frakoblet 1 = tilkoblet
bp - band-pass filter 0 = frakoblet 1 = tilkoblet
hp - high-pass filter 0 = frakoblet 1 = tilkoblet
res - resonans (0-15)

Cut-off frekvensen kan specificeres til at være enhver værdi mellem 0 og 2047. Slå low-pass filteret til ved at angive et 1-tal som anden parameter i FILTER instruktionen. Band-pass filteret aktiveres ved at angive et 1tal som tredie parameter og slå high-pass filteret til med et 1-tal i fjerde parameterposition. Sluk for et hvilket som helst af de tre filtre ved at sætte et 0 i den position for det filter, De ønsker at frakoble. De kan til- eller frakoble eet, to eller alle tre filtre samtidig.

Nu, hvor De har en vis baggrundsviden om FILTER instruktionen, kan De tilføje Deres lydprogram denne linie. Men brug ikke RUN endnu.

45 FILTER 1200, 1, 0, 0, 10

Linie 45 indstiller cut-off frekvensen til 1200, tilkobler low-pass filteret, frakobler high-pass og band-pass filtrene og angiver 10 som resonansniveau. Gå nu tilbage for at tilkoble filteret i Deres PLAY instruktioner ved at forandre alle X0 filterkontroller til X1. Nulstil lydchippen ved at trykke RUN/STOP og RESTORE tasterne og kør Deres lydprogram igen. Læg mærke til forskellene mellem den måde, tonerne lyder på nu og den måde de lød uden filter. Linie 45 ændres til:

45 FILTER 1200, 0, 1, 0, 10

Den nye linie 45 frakobler low-pass filteret og tilkobler band-pass filteret. Tryk på RUN/STOP og RETURN og kør Deres program igen. Læg mærke til forskellen mellem low-pass og band-pass filtrene. Nu ændres linie 45 til:

45 FILTER 1200, 0, 0, 1, 10

Nulstil lydchippen og kør igen programeksemplet. Læg mærke til forskellen mellem high-pass og low-pass filtrene. Foretag eksperimenter med forskellige cut-off frekvenser, resonansniveauer og filtre for at perfektionere musik og lyd i Deres egne programmer.

SAMMENKÆDNING AF DERES MUSIKPROGRAM

Deres første musikprogram er nu færdigt. De er nu i stand til at programmere Deres yndlingssange. Lad os nu kæde alle komponenterne sammen. Her er programlisten. Vær ikke nervøs, det er samme program, som De opbyggede i dette afsnit, bortset fra at der er tilføjet print instruktioner, så De kan se, hvilke programlinier, der bliver spillet.

```
10 ENVELOPE 0,5,9,2,2,2,1700
15 VOL 8
20 TEMPO 10
25 PRINT "LINIE 30"
30 PLAY "C D E F G A B M"
35 FILTER 1200,0,0,1,10
40 PRINT "LINIE 45 - FILTER FRA"
45 PLAY "V2 O5 T7 U5 X0 C D E F G A B M"
50 PRINT "SAMME SOM LINIE 45 - FILTER TIL"
55 PLAY "V2 O5 T7 U5 X1 C D E F G A B M"
60 PRINT "LINIE 65 - FILTER FRA"
65 PLAY "V3 O6 U7 T6 X0 C D E F G A B M"
70 PRINT "SAMME SOM LINIE 65 - FILTER TIL"
75 PLAY "V3 O6 U7 T6 X1 C D E F G A B M"
80 PRINT "LINIE 85 - FILTER FRA"
85 PLAY "V2 O6 TO U7 X0 H CD Q EF I GA S B M"
90 PRINT "SAMME SOM LINIE 85 - FILTER TIL"
95 PLAY "V2 O6 TO U7 X1 H CD Q EF I GA S B M"
100 PRINT "LINIE 105 - FILTER FRA"
105 PLAY "V1 O4 T4 U8 X0 H .C D Q £ EF I $ GA S .B M"
110 PRINT "SAMME SOM LINIE 105 - FILTER TIL"
115 PLAY "V1 O4 T4 U8 X1 H .C D Q £ EF I $ GA S .B M"
```

Linie 10, ENVELOPE instruktionen, specificerer envelope for piano (0), som sætter attack til 0, decay til 9, sustain til 0 og release til 0. Den vælger endvidere variabel puls bølgeform med pulsbredde 1700. Linie 15 indstiller VOLumne til 8. Linie 20 vælger TEMPO 10.

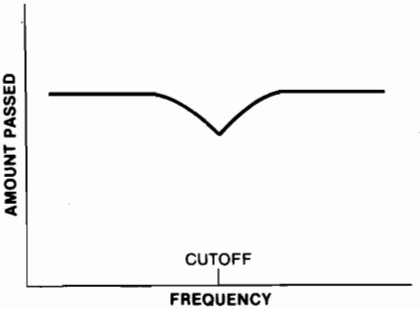
Linie 35 filtrerer noderne som spilles i linierne 30 t.o.m. 115 og sætter FILTER cut-off frekvensen til 1200. Samtidig frakobler linie 35 low-pass og band-pass filtrene med de to nuller, som følger efter cut-off frekvensen (1200). High-pass filteret tilkobles med 1-tallet efter de to nuller. Resonansen sættes til 10 med sidste parameter i FILTER instruktionen.

Linie 30 spiller noderne C, D, E, F, G, A, B i denne rækkefølge. Linie 45 spiller samme noder som i linie 30, men specificerer SID kontrolkaraktererne U5 som styrkeniveau 5, V1 som stemme 1 og O5 som oktav 5. Husk, at SID kontrolkarakterne tillader Dem at ændre synthesizer-kontroller indenfor en streng og derved overtage størsteparten af kontrollen over synthesizeren. Linie 65 specificerer kontrolkarakterne U7 for styrkeniveau 7, V3 for stemme 3, O6 for oktav 6 og X0 for at frakoble filteret. Linie 65 spiller samme noder som linierne 30 og 45, men med anden styrke, stemme og oktav.

Linie 85 har samme styrke, stemme og oktav som linie 65, men specificerer halvnoder for noderne C og D, fjerdedelsnoder for noderne E og F, ottendedelsnoder for noderne G og A og en sekstendedelsnode for B noden. Linie 105 sætter styrken til 7, stemme 1, oktav 4 og frakobler filteret. Den angiver også C noden som en forlænget halvnode, E som en fjerdedelsnode med kryds, G og A som ottendedelsnoder med b og B som en forlænget node med kryds.

AVANCERET FILTRERING

Hvert af de foregående FILTER eksempler brugte kun et filter ad gangen. De kan kombinere SID chippens tre filtre med hinanden for at opnå forskellige filtreringseffekter. De kan, for eksempel, tilkoble low-pass og high-pass filtrene samtidig for at danne et 'notch-reject' filter. Et notch-reject filter lader frekvenserne under og over cut-off passere gennem SID chippen, mens frekvenserne lige omkring cut-off filtreres fra. Se figur 10 for en grafisk fremstilling af et 'notch-reject' filter.



Figur 10 - notch-reject filter

De kan også koble enten low-pass eller high-pass filteret sammen med bandpass filteret, for at opnå interessante effekter. Ved at mixe band-pass filteret med low-pass filteret, kan De vælge båndet med frekvenser under, og lavere end cut-off frekvensen. Resten bortfiltreres.

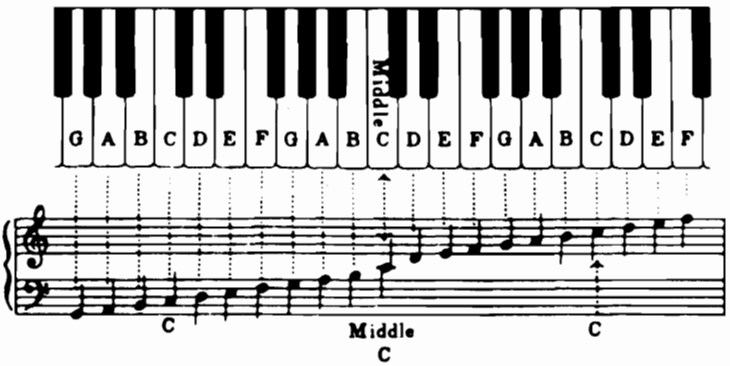
Ved at mixe band-pass og high-pass filtrene, kan De vælge det frekvensbånd, der ligger over cut-off frekvensen og højere. Alle frekvenser under cut-off bortfiltreres.

Eksperimenter med forskellige filterkombinationer og konstater, hvor mange forskellige accenter, De kan placere på Deres musikalske noder og lydeffekter. Filtrene er fremstillet til at perfektionere de lyde, som er skabt af andre komponenter end SID chippen. Når De har fremstillet de musikalske noder eller lydeffekter med SID chippen, gå så tilbage og tilføj Deres program FILTER funktioner, for at gøre dem så gode og rene som muligt.

De har nu al den information, De behøver for at kunne skrive Deres egne musikprogrammer i Commodore 128 BASIC. Eksperimenter med de forskellige bølgetyper, ADSR indstillinger, TEMPOer og FILTER instruktioner. Få fat i et nodehæfte og indtast noderne fra en musikskala i den rigtige sekvens i en PLAY streng. Marker strengens noder med SID kontrolkarakterer. De kan kombinere Deres Commodore 128 musik synthesizer med C128 mode grafik og på den måde fremstille Deres egne 'videofilm', komplette med lydspor.

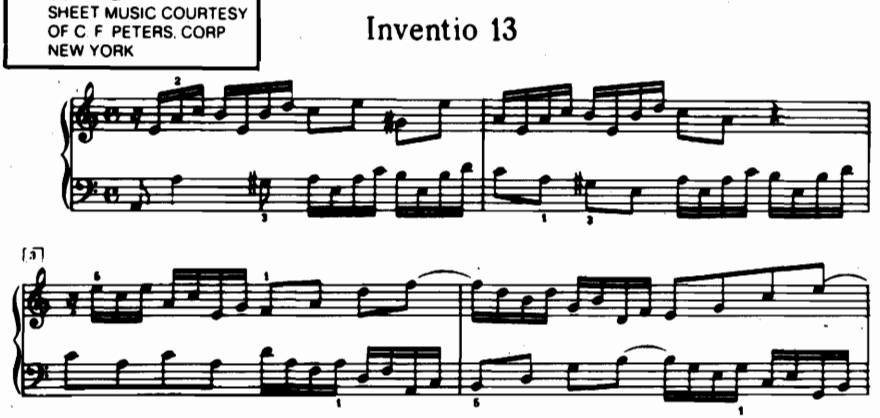
HVORLEDES MAN INDKODER EN SANG FRA ET NODEBLAD

Dette afsnit giver et eksempel på nodebladsmusik og illustrerer, hvorledes man oversætter noder til en form, som Commodore 128 kan forstå. Denne øvelse vil naturligvis være væsentlig hurtigere og lettere, hvis De ved, hvordan man læser musik. De behøver imidlertid ikke at være musiker for at være i stand til at spille melodien på Deres Commodore 128. For dem, som ikke kan læse musik, viser figur 11, hvorledes en typisk noderække er opbygget og hvorledes noderne på nodestregen er relaterede til tangenterne på et piano.



Figur 11 - pianotangenter og nodelinie

Figur 12 er et udsnit af en komposition med titlen INVENTIO 13, skrevet af Johann Sebastian Bach. Skønt denne komposition er skrevet for nogle hundrede år siden, kan den spilles og nydes med de mest moderne computer synthesizere, som f. eks. SID chippen i Commodore 128. Her er åbningsstykket af INVENTIO 13. (De første fire takter).



Figur 12 - del af Bach's INVENTIO 13

Den bedste måde at begynde indkodning af en komposition på til Deres Commodore 128 er ved at nedbryde noderne til en foreløbig kode. Nedskriv diskantlinier på et stykke papir. Nedskriv derefter baslinierne. Sæt varighedskoder foran nodeværdien. Sæt for eksempel et ottetal foran en ottendedelsnode, seksten foran en sekstendedelsnode o.s.v. Adskil derefter noderne, så diskantlinierne i een takt proportionalt stemmer overens med baslinierne i samme takt.

Har kompositionen en tredie stemme, skal De adskille den, så varigheden er proportional med de to andre nodelinier. Når alle liniers noder er adskilt med ens varighed, bør en særlig udvalgt stemme spille de enkelte linier.
For eksempel kunne stemme 1 spille øverste nodelinie, stemme 2 anden linie og stemme 3 den nederste stemme, hvis den findes.

Lad os sige, at øverste nodelinie indledes med en streng indeholdende fire ottendedelsnoder. Lad os samtidig sige, at nederste nodelinie indleder med en streng indeholdende otte sekstendedelsnoder. Da en ottendedelsnode tidsmæssigt er proportional med to sekstendedelsnoder, adskilles noderne som vist i figur 13.

STEMME 1 =	**8A**	**8B**	**8C**	**8D**
STEMME 2 =	**16D 16E**	**16F 16G**	**16A 16B**	**16C 16D**

Figur 13 - Synkronisering af to stemmers noder

Da synkroniseringen og timingen i en musikalsk komposition er kritisk, må De være sikker på, at noderne i øverste nodelinie for stemme 1, for eksempel, tidsmæssigt stemmer overens med stemme 2's noder i nederste nodelini e. Den første node i figur 13's øverste linie er en A ottendedelsnode. De første to noder for stemme 2 er D og E sekstendedelsnoder. De må i dette tilfælde først angive stemme 1 ottendedelsnoden i PLAY strengen, og umiddelbart efter sekstendedelsnoderne for stemme 2. For at fortsætte med eksemplet er anden node i figur 13 en B ottendedelsnode for stemme 1 (øverste nodelinie). B ottendedelsnoden svarer tidsmæssigt til de to sekstendedelsnoder, F og G, som findes i nederste linie for stemme 2. For at koordinere timingen, indtastes B ottendedelsnoden i stemme 1's streng efterfulgt af de to sekstendedelsnoder, F og G, for stemme 2.

Følg reglen med altid at begynde med den node, som har den længste varighed. Hvis, for eksempel, en nodelinie indledes med en serie på to sekstendedelsnoder for stemme 2 i nederste linie, og den øverste linie begynder med en ottendedelsnode for stemme 1, indtastes ottendedelsnoden først i strengen, idet den skal spille i tidsrummet, mens de to sekstendedelsnoder hentes af Commodore 128. De må give computeren tid til at spille den længste node først, og så spille noderne af kortere varighed, ellers vil kompositionen ikke blive synkroniseret.

Her er det program, som spiller de første fire takter af 'Inventio 13'.

```
10  REM INVENTIO 13 af J. S. BACH (fire første takter)
20  TEMPO 6
30  PLAY"V104T7U8X0":REM STEMME 1=ORGEL
40  PLAY"V204T7U8X0":REM STEMME 2=PIANO
50  REM FØRSTE TAKT
60  A$="V201IAV103IEV202QAV103SA04C03BEV202I#GV103SB04DV104ICV202SAEM"
70  B$="V104IEV202SA03CV103I#GV202SBEV104IEV202SB03D"
80  REM ANDEN TAKT
90  C$="V203ICV103SAEV202IAV103SA04CV202I#GV103SBEV202IEV103SB04D"
100 D$="V104ICV202SAEV103IAV202SA03CV104QRV202SBEB03D"
110 REM TREDIE TAKT
120 E$="V203ICV104SREV202IAV104SCEV203ICV103SA04CV202IAV102SEG"
130 F$="V103IFV203SD02AV103IAV202SFAV104IDV202SDFV104IFV201SA02C"
140 REM FJERDE TAKT
150 G$="V201IBV104SFDV202IDV103SB04DV202IGV103SGBV202IBV103SDF"
160 H$="V103IEV202SGEV103IGV202SEGV104ICV202SCEV104IEV201SGB"
170 PLAY A$:PLAY B$:PLAY C$:PLAY D$:PLAY E$:PLAY F$:PLAY G$:PLAY H$
180 END
```

Indtast det på Deres C128, gem det med SAVE for fremtidig anvendelse, og spil det så med RUN. Har De fået lyst til at fuldende værket for at kunne høre det i sin fulde udstrækning, kan De købe nodehæftet J. S. Bach "Zweistemmige Inventionen" hos Wilhelm Hansens Musik-Forlag i Århus eller København. Nodehæftet har "Edition nummer 3970".

De kan bruge teknikkerne beskrevet i dette afsnit til indkodning fra nodebladene med den musik, De holder mest af, og spille den på Deres Commodore 128.

****

De er nu blevet introduceret til de fleste af de kraftfulde nye kommandoer i BASIC 7.0 sproget, som De kan bruge i C128 mode. I følgende afsnit vil De lære at bruge såvel 40- som 80 kolonners skærmbilleder på Commodore 128

AFSNIT 8
BRUG AF 80 KOLONNER

INTRODUKTION

I C128 og CP/M mode kan De vælge mellem skærmbilleder med 40 eller 80 kolonner. De kan endog bruge begge indenfor eet program.

Hver skærm bruges til noget specielt. 40 kolonners skærmen har samme format, som bruges af Commodore 64. Med 40 kolonners skærmen kan De anvende alle grafiske muligheder i Commodore 128. De kan tegne cirkler, grafik, spritekarakterer, kasser og andre figurer i højopløsnings- eller flerfarvegrafik modes. De kan også bruge sprites.

Hvis De bruger 80 kolonner, får De dobbelt så mange karakterer pr. linie.
I 80 kolonners mode kan De bruge de standard grafikkarakterer og farver, som kan dannes med tastaturet.

De kan også, for at udnytte begge skærmformater, skrive programmer, som bruger to monitorer, hvor hver monitorskærm viser forskellige af programmets aspekter. For eksempel kunne tekstuddata vises på 80 kolonners skærmen, mens grafisk uddata vistes på 40 kolonners monitoren.

40/80 TASTEN

De kan bruge 40/80 tasten til at indstille skærmbredden til enten 40 eller 80 kolonner. Nedtrykning af denne tast har kun effekt, hvis een af følgende forholdsregler er taget:

1. Der tændes for strømmen
2. RESET tasten er indtrykket
3. RUN/STOP og RESTORE tasterne er nedtrykket samtidigt

40/80 tasten virker på samme måde som SHIFT/LOCK tasten; den fastlåses, når den trykkes ned og udløses først, når der trykker på den igen. Hvis tasten er i øverste stilling mens en af ovennævnte hændelser indtræffer, sættes skærmformatet til 40 kolonner. Hvis tasten nedtrykkes og dermed fastlåses, før der er tændt for strømmen, og en af ovennævnte ting sker, sættes skærmen til 80 kolonner. Når computeren kører i et af skærmformaterne (40 eller 80), kan der ikke skiftes til andet format ved at nedtrykke 40/80 tasten. I så fald må De nedtrykke og udløse ESC tasten og derefter trykke på X tasten.

VIDEO STIK OG MONITORER TILSLUTNING AF EN MONITOR

Vær omhyggelig med at forbinde Deres monitor korrekt til stikkene bag på Deres computer. Der findes to åbninger; en mærket VIDEO og en anden mærket RGBI.

VIDEO er tilslutningsstikket for en almindelig 40 kolonners video monitor, mens RGBI bruges til 80 kolonners monitorer. Dobbeltfungerende monitorer som Commodore 1901, som kan vise enten 40 kolonners standard eller RGBI 80 kolonners skærmbilleder, tilsluttes begge stik.

MONITORTYPER

Standardmonitorer (composite) er fremstillet til at vise 40 kolonners uddata på skærmen. Eksempler på sådanne monitorer er Commodore 1701 og 1702 monitorerne. Disse monitorer kan bruges med alle 40 kolonners programmer og i alle tre modes. De kan imidlertid ikke bruges i forbindelse med 80 kolonners arbejde.

RGBI monitorer

RGBI monitorer er fremstillet til specielt at vise 80 kolonners uddata. Selv om RGBI står for Rød Grøn Blå Intensitet, kan RGBI monitorer være både monochrome (enkeltfarvede) og farvemonitorer. De fleste populære monochrom monitorer har skærmfarven grøn eller amber. En RGBI monitor, som er sluttet til RGBI stikket kan behandle 80 kolonners uddata i både C128 og CP/M mode.

Dobbeltfunktions monitorer

Dobbeltfunktions monitorer, som Commodore 1901, kan fremstille enten et standard (40 kolonners) eller et RGBI (80 kolonners) billede. Denne monitortype skal forbindes til begge videostik. Med en kontakt på monitoren kan man vælge mellem skærmformaterne. 40/80 tasten på Deres computer fastlægger, ved opstart, typen på de viste skærmdata. Undersøg om 40/80 tastens indstilling svarer overens med 40/80 kolonners glidekontakten på monitorens frontpanel. OBS! De kan stadig skifte frem og tilbage mellem 40 og 80 kolonners uddata ved at nedtrykke

og genudløse ESC tasten og derefter trykke på X tasten, uanset hvilket position 40/80 tasten er i.

ANVENDELSE AF PRÆFREMSTILLET 80-KOLONNERS SOFTWARE

De fleste CP/M programmer benytter 80 kolonners skærmbilleder, og det samme gør mange af de andre administrative systemer, De kan bruge i C128 mode. Da normalbredden af en udskrevet papirside er 80 tegn, kan et 80 kolonners tekstbehandlingsanlæg vise information på skærmen, nøjagtigt som samme information vil fremtræde på papiret. Kalkulationssystemer (spreadsheet) specificerer ofte et 80 kolonners format for at give plads nok til de nødvendige kolonner og oplysningskategorier. Mange databasesystemer og telekommunikationsprogrammer kræver ligeledes, eller kan bruge en 80 kolonners skærm.

FREMSTILLING AF 80 KOLONNERS PROGRAMMER

Samtidig med, at man bruger præprogrammeret software, kan 80 kolonners skærmbredde være nyttig, når De fremstiller egne programmer. De har formentlig lagt mærke til, hvad der sker, når De på en 40 kolonners skærm indtaster en linie som er mere end 40 karakterer lang. Linien 'wrapper', d.v.s fortsætter med teksten på næste linie. Dette kan gøre det vanskeligt at læse linien, og programmeringsfejl opstår lettere. En 80 kolonners skærm hjælper til at udelukke disse problemer. Generelt set giver en 80 kolonners skærm klarere skærm og mulighed for lettere redigering.

SAMTIDIG ANVENDELSE AF 40 OG 80 KOLONNER

Hovedfordelen ved anvendelse af 40 kolonners standard video uddata er, at man kan vise højopløsnings grafik, mens 80 kolonner giver uddata for tekstbehandling og andre administrative systemer. Hvis De har to monitorer, kan De skrive programmer, som 'deles' mellem 80 kolonners tekst og 40 kolonners grafik. En speciel kommando, GRAPHIC 1,1, kan bruges i et program for at overføre udførelsen af grafikkommandoer til 40 kolonners skærmen. Har De en dobbeltfunktions monitor, som kan vise både 40 og 80 kolonners skærmbilleder, kan De placere GRAP-

HIC 1,1 instruktioner i Deres program, så al grafik bliver uddata i 40 kolonners skærmformat. De vil, imidlertid, for at se den grafiske uddata, være nødt til at skifte kontakten på monitoren til 40 kolonner. Skriver De et sådant program, vil det være en god ide at indsætte skærmmeddelelser til brugeren om at omstille monitorens kontakt.

De ønsker for eksempel at skrive et program, som beder brugeren om indtastning af data, og som svar herpå danner en grafisk søjle. Meddelelsen "SKIFT TIL 40 KOLONNERS FORMAT FOR GRAFIK" kunne så fortælle brugeren, at han for at se resultatet skal ændre kontaktindstillingen.
Som før nævnt, kan De, efter at strømmen er slået til, skifte mellem 40 og 80 kolonners formater med ESCape/X sekvensen.

Følgende eksempel viser, hvorledes dobbeltfunktions skærme kan bruges i et program:

```
10  GRAPHIC 5,1 :REM DENNE INSTRUKTION SKIFTER TIL 80 KOLONNERS MODE
20  PRINT "!SHIFT/CLR!  START I 40 KOLONNERS UDDATAFORMAT":PRINT
30  PRINT "SÆT GLIDEKONTAKTEN PÅ COMMODORE 1902 I MIDTERPOSITIONEN"
40  PRINT:PRINT"TRYK RETURN, NÅR DE ER KLAR"
50  GRAPHIC 0,1
60   PRINT:PRINT"TRYK RETURN, NÅR DE ER KLAR":GETKEY A$:IF A$ <> CHR$13
THEN 60
70  COLOR 1,5: COLOR 4,1: COLOR 0,1
80  GRAPHIC 2,1:CHAR 1,8,18,"HØJOPL/TEKST DELT SKÆRM":REM VALG DELT SKÆRM
90  FORI=70 TO 220 STEP 20:CIRCLE 1,I,50,30,30:NEXT
100 PRINT "  SKIFT TIL 80 KOLONNERS FORMAT"
110 PRINT "  SÆT GLIDEKONTAKTEN PÅ MONITORENS FORSIDE HELT TIL HØJRE"
120 PRINT "  TRYK RETURN, NÅR DE ER KLAR": GETKEY A$:IF A$ <> CHR$(13)
THEN 120
130 GRAPHIC 5,1:REM DENNE INSTRUKTION SKIFTER TIL 80 KOLONNERS MODE
140 FOR J=1 TO 10
150 PRINT "DE ER NU I 80 KOLONNERS TEKSTFORMAT"
160 NEXT:PRINT
170 PRINT"SKIFT NU TILBAGE TIL 40 KOLONNERS FORMAT"
180 PRINT"SÆT GLIDEKONTAKTEN PÅ MONITOREN I MIDTERSTILLINGEN"
190 PRINT"TRYK RETURN, NÅR DE ER KLAR" : GETKEY A$ : IF A$ <> CHR (13)
THEN 190
200 GRAPHIC 0,1:REM DENNE INSTRUKTION SKIFTER TIL 40 KOLONNERS FORMAT"
210 FOR J=1 TO 70
220 PRINT "DE ER NU I 40 KOLONNERS SKÆRMFORMAT"
230 NEXT
```

Hvert skærmformat indeholder visse fordele; de to formater kan endog kombineres i et program for at komplettere hinanden.

Bruger De 40 kolonners skærm, opnår De alle fordele ved avanceret BASIC grafik. 80 kolonners format giver Dem mere plads til Deres programmer. Til og med lader det Dem bruge et bredt udsnit af software, som er lavet til at køre i 80 kolonners skærmformat.

****

Afsnittene i dette kapitel har introduceret Dem til mange faciliteter og muligheder med Commodore 128 i C128 mode. Følgende kapitel fortæller, hvorledes De bruger Commodore 128 i C64 mode.

KAPITEL

3

BRUG AF C64 MODE

AFSNIT 9
BRUG AF TASTATURET I C64 MODE

ANVENDELSE AF BASIC 2.0

Hele det BASIC 2.0 sprog, som findes i Commodore 64, er indlagt i Commodore 128's BASIC 7.0 sprog. De kan bruge BASIC 2.0 kommandoer i såvel C128 som C64 mode. For fuld beskrivelse af disse kommandoer henvises til afsnittene 3 og 4 i kapitel 2.

TASTATURETS KARAKTERSÆT

På illustrationen i afsnit 3, er de skyggelagte taster dem, der kan bruges i C64 mode. Tastaturet har i C64 mode de to samme karaktersæt som i C128 mode:

- Upper-case/grafisk tegnsæt
- Upper-case/lower-case tegnsæt

Når De går i C64 mode, har tastaturet upper-case/grafik, således at alt, hvad De indtaster bliver i store bogstaver. I C64 mode kan kun eet karaktersæt bruges ad gangen. Der kan skiftes frem og tilbage mellem tegnsættene ved samtidigt at nedtrykke SHIFT tasten og Commodore (C=) tasten.

BRUG AF SKRIVEMASKINE TASTERNE

Som i C128 mode kan De i C64 mode bruge både store og små bogstaver på tastaturet. De kan også indtaste tallene i hovedtastaturets øverste række.
De kan derudover fremstille de grafiske tegn som findes på tasternes forsider.

BRUG AF KOMMANDO TASTERNE

De fleste kommandotaster (d.v.s. de taster, som afgiver meddelelser til computeren, som RETURN, SHIFT, CTRL etc.) virker i C64 mode på samme måde som i C128 mode.

Den eneste forskel, som er i C64 mode, er, at De kun kan bevæge markøren ved hjælp af de to CRSR taster i tastaturets nederste højre hjørne.
(I C128 mode kan De også bruge de fire piltaster på tastaturet øverste højre del.)

MARKØRBEVÆGELSE I C64 MODE

I C64 mode bruges de to CRSR taster på tastaturet samt SHIFT tasten til at flytte markøren, som beskrevet i afsnit 3.

PROGRAMMERING AF FUNKTIONSTASTER I C64 MODE

De fire taster på tastaturets højre side, lige ovenfor det numeriske tastatur, kaldes funktionstasterne. De er på oversiden mærket F1, F3, F5 og F7 og på forsiden F2, F4, F6 og F8. Disse taster kan programmeres - d.v.s. at de kan instrueres om at udføre en bestemt opgave eller funktion. Derfor kaldes disse taster ofte programmerbare funktionstaster.

For at udføre de funktioner, som er knyttet til markeringerne på tasternes forside - det er F2, F4, F6, F8 - skal De holde SHIFT tasten nedtrykket. Derfor kaldes disse taster af og til SHIFTede programmerbare funktionstaster.

Funktionstasterne er, i C64 mode, ikke tildelt karakterer, som udskrives. De har imidlertid tildelte CHR$ koder. I virkeligheder har hver af dem to CHR$ koder - een som virker, når der blot trykkes på tasten, og een som virker, når De nedtrykker tasten samtidig med SHIFT. For at aktivere tasterne med lige numre holdes SHIFT nede, mens der trykkes på funktionstasten. For for eksempel at bruge F2 holdes SHIFT nede, mens der trykkes på F1.

CHR$ koder for F1-F8 tasterne ligger i området fra 133 til 140. Koderne er imidlertid ikke tildelt tasterne i numerisk rækkefølge. Tasterne og deres modsvarende værdier er som følger:

F1 - CHR$(133)
F2 - CHR$(137)
F3 - CHR$(134)
F4 - CHR$(138)
F5 - CHR$(135)
F6 - CHR$(139)
F7 - CHR$(136)
F8 - CHR$(140)

De kan bruge funktionstasterne i Deres program på flere måder. For at gøre dette, er De nødt til at bruge GET instruktionen (se afsnit 4 for beskrivelse af GET instruktionen). Som et eksempel instruerer efterfølgende program F1 tasten om at udskrive en meddelelse på skærmen:

```
10 ? "TRYK PÅ F1 FOR AT FORTSÆTTE"
20 GET A$ : IF A$ = "" THEN 20
30 IF A$ <> CHR$(133) THEN 20
40 ? "DE HAR TRYKKET PÅ F1"
```

Linierne 20 og 30 udfører det meste af arbejdet i programmet. Linie 20 får computeren til at vente, til en tast er blevet nedtrykket, før den udfører mere af programmet. Læg mærke til, at hvis kommandoen umiddelbart efter THEN er en GOTO, er kun linienummeret nødvendigt. Læg endvidere mærke til, at en GOTO kommando kan GOTO til sin egen linie. Linie 30 får computeren til at returnere og vente, hvis andre taster end F1 er blevet nedtrykket.

AFSNIT 10
LAGRING OG GENINDLÆSNING AF DERES PROGRAMMER I C64 MODE

Når De har redigeret et program, ønsker De formentlig at lagre det permanent, så De senere vil være i stand til at genkalde og bruge det igen. For at gøre dette, skal De enten råde over en Commodore diskettestation eller Commodore Datassette.

DISKETTEFORMATTERING I C64 MODE

Før det er muligt at lagre programmer på en ny (blank) diskette, skal den formatteres for at kunne modtage data. Dette kaldes disketteformattering.
Vær sikker på, at De har tændt for diskettestationen, før De indsætter en diskette.

En blank diskette formatteres i C64 mode ved at indtaste denne kommando:

OPEN 15,8,15:PRINT# 15,"N0:NAVN,ID" RETURN

I stedet for NAVN indtaster De her et diskettenavn efter eget valg; til navngivning af disketten kan benyttes op til 16 karakterer. I stedet for ID tastes en totegns kode efter eget valg, som f. eks. W2 eller 10.

Under formatteringsprocessen forsvinder markøren. Når markøren atter blinker, indtastes følgende kommando:

CLOSE 15 RETURN

OBS! Uanset om en diskette er formatteret i C64 eller C128 mode, kan den bruges i begge modes.

SAVE KOMMANDOEN

De kan bruge SAVE kommandoen til at gemme Deres program på diskette eller kassettebånd.

LAGRING PÅ DISKETTE

Hvis De har en Commodore enkeltdrevs diskettestation, kan Deres program gemmes med følgende kommando:

SAVE "PROGRAMNAVN",8 RETURN

8-tallet oplyser computeren om, at der bruges diskettestation til lagring af programmet.

For PROGRAMNAVN gælder samme regler, uanset om der bruges diskette eller bånd. PROGRAMNAVN kan være alt, hvad De ønsker. De kan bruge bogstaver, tal eller/og symboler - op til 16 karakterer ialt. Læg mærke til, at De skal omgive programmets navn med anførelsestegn. Markøren på skærmen forsvinder, mens lagringen foretages, men vender tilbage, når processen er færdig.

LAGRING PÅ KASSETTEBÅND

Hvis De bruger en Datasette til at gemme Deres programmer på, lægges et blankt bånd i båndstationen, og hvis nødvendigt tilbagespoles båndet.
Derefter tastes:

SAVE "PROGRAMNAVN" RETURN

LOAD og RUN KOMMANDOERNE DISKETTE

For at indlæse et program fra diskette skal indtastes:

LOAD "PROGRAMNAVN",8 RETURN

8-tallet indikerer overfor computeren, at der arbejdes med diskettedrev.

For at udføre programmet tastes RUN RETURN .

KASSETTEBÅND

Et program indlæses ved at indtaste:

LOAD "PROGRAMNAVN" RETURN

Kender De ikke programmets navn, kan De blot indtaste:

LOAD RETURN

og det næste program på båndet vil blive indlæst.

De kan bruge Datasettens tæller til at finde programmernes startposition.
Når De så ønsker at finde et bestemt program, spoler De blot båndet fremad til det rigtige tal og indtaster:

LOAD RETURN

I dette tilfælde behøver De ikke at angive programmets navn; programmet vil automatisk indlæses, fordi det er det næste på båndet.

> OBS! Under indlæsning med LOAD bliver programmet på båndet eller disken ikke slettet; det bliver simpelthen kopieret til computerens hukommelse. Imidlertid bliver et program, som evt. befinder sig i computerens hukommelse, slettet under indlæsning af et andet.

ANDRE DISKETTERELATEREDE KOMMANDOER
VERIFIKATION AF ET PROGRAM

For at verificere at et program er blevet korrekt lagret eller indlæst, tastes:

VERIFY "PROGRAMNAVN",8 RETURN

Hvis programmet i computeren er identisk med programmet på disketten, vil der på skærmen blive svaret med "OK".

VERIFY kommandoen virker også i forbindelse med båndprogrammer. Indtast:

VERIFY "PROGRAMNAVN" RETURN

Læg mærke til, at der ikke må indtastes komma og 8, da 8 vil indikere, at der arbejdes med diskettedrev.

VISNING AF DIRECTORY (indholdsfortegnelse)

Ønsker De på skærmen at se en oversigt over programmer på en diskette, skal De indtaste:

LOAD "$",8 RETURN

Under denne procedure forsvinder markøren. Når den atter viser sig tastes:

LIST RETURN

På skærmen vil nu blive vist en oversigt over programmerne på disketten.

Læg mærke til, at et program i hukommelsen vil blive slettet, hvis De indlæser Directory.

INITIALISERING AF ET DISKETTEDREV

Hvis lampen på diskettestationen blinker, betyder det, at der er opstået en diskettefejl. De kan genetablere den tilstand, diskettedrevet var i, før fejlen opstod, ved at benyttede en procedure som kaldes INITIALISERING. Et diskettedrev initialiseres ved at taste:

OPEN 1,8,15, "I":CLOSE 1 RETURN

Blinker lampen stadig, må De fjerne disketten, slukke for diskettestationen og tænde den igen.

For yderligere information om SAVE og LOADE bedes De gennemlæse Deres diskettestations- eller Datasette vejledning. Se også under SAVE og LOAD kommando beskrivelserne i kapitel 5, BASIC 7.0 ordlisten.

KAPITEL

4

BRUG AF CP/M MODE

AFSNIT 11
INTRODUKTION TIL CP/M 3.0

HVAD ER CP/M 3.0

CP/M er et produkt fra Digital Research Inc. Den CP/M version som bruges på Commodore 128 er CP/M Plus Version 3.0. I dette kapitel refereres der generelt til CP/M som CP/M 3.0 eller blot CP/M. Kapitlet summerer brug af CP/M på Commodore 128.

CP/M er et populært operativsystem beregnet for mikrocomputere. Som operativsystem styrer og fordeler CP/M Deres computers resourcer, incl. hukommelse og diskettelager, konsol (skærm og tastatur), printer og kommunikationsenheder. CP/M håndterer endvidere den information, der er lagret i diskettefiler. CP/M 3.0 kan indlæse filer fra en diskette til computerens hukommelse, eller udlæse til en ydre enhed som f.eks. en printer.

De kan fremstille egne programmer under CP/M, eller De kan vælge blandt det store udbud af færdige CP/M applikationsprogrammer.

HVAD DE BEHØVER FOR AT KØRE CP/M 3.0

Nødvendigt hardware for at kunne bruge CP/M 3.0 er en computer, som indeholder en Z80 mikroprocessor, en konsol bestående af et tastatur og en skærm, og i det mindste eet diskettedrev. Commodore 128 har en indbygget Z80 mikroprocessor, som sætter Dem i stand til at kunne køre CP/M 3.0; konsollen består af hele Commodore 128 tastaturet og en 80 kolonners monitor; og diskettedrevet er den nye, hurtige Commodore 1570/1571 diskettestation. Samtidig findes indlagt i computerens emballage en CP/M diskette, hvor den ene side indeholder CP/M 3.0 systemet og et HELP utility program, og den anden side indeholder et antal af andre utility programmer.

OBS! CP/M kan bruges med en 40 kolonners monitor, men så kan kun 40 kolonner vises ad gangen. For at se alle 80 kolonner på skærmen, må skærmbilledet rulles (scrolles) horisontalt ved at trykke på CONTROL tasten og den passende markørtast (højre eller venstre).

HVAD DER FINDES PÅ DERES CP/M 3.0 DISKETTE

```
show b:

B: RW, Space:          3361<

A>dir !full

Scanning Directory...

Sorting  Directory...

Directory For Drive A:  User  0

    Name      Bytes  Recs Attributes     Name      Bytes  Recs   Attributes
-------------------------------------------------------------------------------
CCP      COM     4k    25 Dir  RW     CPM+     SYS    23k   182   Dir  RW
DIR      COM    15k   114 Dir  RW     FORMAT   COM     5k    35   Dir  RW
HELP     COM     7k    56 Dir  RW     HELP     HLP    83k   664   Dir  RW
KEYFIG   COM    10k    75 Dir  RW     KEYFIG   HLP     9k    72   Dir  RW
PIP      COM     9k    68 Dir  RW

Total Bytes     =   165k   Total Records =    1291   Files Found =   9
Total 1k Blocks =   165    Used/Max Dir Entries For Drive A:   15/  64

A>dir b:  !full

Scanning Directory...

Sorting  Directory...

Directory For Drive B:  User  0

    Name      Bytes  Recs Attributes     Name      Bytes  Recs   Attributes
-------------------------------------------------------------------------------
DATE     COM     8k    25 Dir  RW     DATEC    ASM     2k     5   Dir  RW
DATEC    RSX     2k     3 Dir  RW     DEVICE   COM    16k    58   Dir  RW
DIR      COM    30k   114 Dir  RW     DIRLBL   RSX     4k    12   Dir  RW
DUMP     COM     2k     8 Dir  RW     ED       COM    20k    73   Dir  RW
ERASE    COM     8k    29 Dir  RW     GENCOM   COM    30k   116   Dir  RW
GET      COM    14k    51 Dir  RW     INITDIR  COM    64k   250   Dir  RW
PATCH    COM     6k    19 Dir  RW     PIP      COM    18k    68   Dir  RW
PUT      COM    14k    55 Dir  RW     RENAME   COM     6k    23   Dir  RW
SAVE     COM     4k    14 Dir  RW     SET      COM    22k    81   Dir  RW
SETDEF   COM     8k    32 Dir  RW     SHOW     COM    18k    66   Dir  RW
SUBMIT   COM    12k    42 Dir  RW     TYPE     COM     6k    24   Dir  RW

Total Bytes     =   314k   Total Records =    1168   Files Found =  22
Total 1k Blocks =   314    Used/Max Dir Entries For Drive B:   23/  64

A>
```

CP/M+.SYS

Dette er CP/M Plus systemets hovedfil. Den indeholder alle de systemkomponenter, som permanent vil findes i hukommelsen: Basic Input/Output Systemet (BIOS), som indlæses til øverste niveau i hukommelsen, Basic Disk Operativsystemet, som indlæses lige under BIOS i hukommelsen, og System Parameterne, som indlæses i hukommelsens nederste niveau.

CCP.COM

Når CP/M "bootes", indlæses Console Command Processor (CCP) til hukommelsen lige under BDOS. Den resterende hukommelse, under CCP og over side 0, kaldes Transient Program Area (TPA), og det er hertil, applikationer bliver indlæst. I C128's CP/M udgør dette areal 59k. CCP er det program, som behandler al inddata (som normalt indtastes fra tastaturet) der gives som svar på systemmeldingen (A⟩). Det indeholder de 6 indbyggede kommandoer (vist i tabel 14.1) og understøtter endvidere de 14 konsol redigeringskommandoer (vist i tabel 13.1).

Hvis der som svar på systemmeldingen indtastes et ord, som ikke svarer til en af de indbyggede kommandoer, vil det af CCP blive opfattet som en ydre kommando, og CCP vil prøve at finde og udføre en fil, som har det angivne navn med tilføjelsen .COM. Hvis en sådan fil ikke findes på den monterede diskette, viser CCP ordet efterfulgt af et spørgsmålstegn og bringer derefter igen systemmeldingen.

Hvis mere end eet ord indtastes som svar på systemmeldingen, bliver alle ord efter det første behandlet som parametere, som skal anvendes sammen med den ydre kommando.

Et sprog eller et applikationsprogram indlæses og udføres ved at angive det, som var det en kommando. Alle CP/M programmer indeholder en .COM fil.

.COM filer

Alle andre .COM filer indeholder ydre kommandoer (vist i tabel 14.2).

HELP.COM viser meddelelser, som findes i HELP.HLP (tilføjelsen viser, at der er tale om en datafil, ikke en programfil), om C128 CP/M systemet og dets kommandoer. Hvis De ikke har kendskab til CP/M og ikke råder over manualer eller bøger om emnet, vil De finde det meget nyttigt at udskrive HELP vejledninger. Hvis De samtidig nedtrykker CONTROL og P vil skærmbilledet også blive udskrevet på printeren; trykkes der atter på CONTROL P vil denne facilitet afbrydes.

Hvis De indtaster HELP, får De en liste over de dækkede områder, og taster De HELP C128_CP/M fås specifik information om denne version af CP/M.
(Tegnet mellem C128 og CP/M fås ved at nedtrykke tasten med venstrepilen, øverst til højre på tastaturet.) Ønsker De udskrivning uden pause efter hvert skærmbillede, indtastes HELP C128 CP/M [NOPAGE.

Ønsker De at lære mere om CP/M, kan disse 2 bøger anbefales:

"Inside CP/M Plus: A Guide For Users" af David E. Corter
Holt, Rinehart & Winston ISBN:0-03-070671
"CP/M - The Software Bus: a programmer's companion" af A. Clarke, J.M. Eaton & D. Powys-lybbe
Sigard Technichal Press ISBN:0-905104-18-8

ANDRE FILER

.ASM betyder, at der er tale om en Assembler source fil.
.RSX betyder, at der er tale om en Resident System eXtension, hvilket er en fil, der automatisk indlæses af en kommandofil, hvis det er nødvendigt.

OPSTART MED CP/M 3.0

Følgende afsnit fortæller, hvorledes De skal starte eller 'boote' CP/M 3.0, hvorledes De skal indtaste på eller redigere kommandolinien og hvorledes De laver sikkerhedskopier af Deres CP/M 3.0 disketter.

LOADING ELLER BOOTING AF CP/M 3.0

Med Loading eller 'booting' af CP/M 3.0 menes indlæsning af en kopi af operativsystemet fra Deres CP/M systemdiskette til computerens hukommelse.

De kan 'boote' CP/M 3.0 på flere måder. Er Deres computer slukket, kan De indlæse CP/M ved først at tænde for Deres diskettedrev og isætte CP/M systemdisketten og derefter tænde for computeren. CP/M 3.0 vil automatisk indlæses. Er De allerede i C128 mode, kan CP/M 3.0 indlæses ved, at De sætter CP/M systemdisketten i diskettestationen og derefter indtaster BASIC kommandoen BOOT, hvorefter CP/M 3.0 så vil blive indlæst. I C128 mode kan De også indlæse CP/M ved at isætte systemdisketten og trykke på RESET knappen.

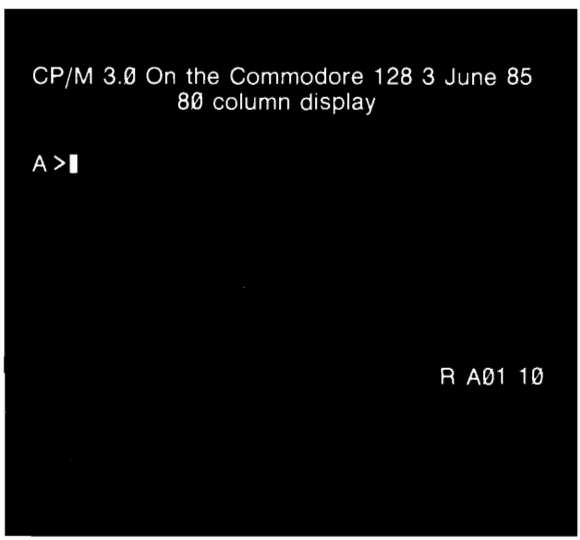
Er De i C64 mode og ønsker at komme i CP/M mode, skal De slukke for computeren. Sæt derefter CP/M systemdisketten i diskettestationen og tænd for computeren.

ADVARSEL! Vær altid sikker på, at disketten er skubbet helt på plads i 1571 drevet, før De drejer låsetappen.

I CP/M 3.0 på Commodore 128 har brugeren 59K TPA (Transient Program Area), der virker som bruger-RAM.

ÅBNING AF CP/M SKÆRMBILLEDE

Når CP/M 3.0 er indlæst til huhommelsen, vises en meddelelse som denne på skærmen:



En vigtig del af åbningsbilledet er følgende totegns meddelelse:

A⟩

Dette er CP/M 3.0 systemets melding (prompt). Meldingen fortæller Dem, at CP/M er klar til at modtage en kommando fra Deres tastatur. Meldingen viser også, at drev A er standardværdien. Dette betyder, at såfremt De ikke giver CP/M besked om andet, vil CP/M lede efter program- og datafiler på diskettedrev A.

OBS! I CP/M opfattes et enkelt diskettedrev som drev A. Dette svarer til enhed nr. 8, drev 0 i C128 og C64 mode. Sædvanligvis er det højeste antal drev, som kan bruges med CP/M 3.0, fire. Disse ekstra drev kaldes drev B, C etc.

KOMMANDOLINIEN

CP/M 3.0 udfører opgaver i henhold til de kommandoer, De indtaster på tastaturet. Disse kommandoer vises på skærmen på den såkaldte kommandolinie.
En CP/M kommandolinie er sammensat af kommando nøgleord og en evt. efterfølgende kommando tilføjelse (tail). Kommandonøgleordene identificerer den kommando, som skal udføres. Kommando-tilføjelsen kan indeholde ekstra oplysninger til kommandoen, som f.eks. filnavne eller parametere. Det følgende er et eksempel på en kommandolinie:

A⟩DIR MINFIL

I dette eksempel er DIR kommando-nøgleordet og MINFIL kommando-tilføjelsen. Kommandolinien afsendes til CP/M 3.0 for behandling ved at trykke på RETURN tasten.

Mens De indtaster tegn på Deres tastatur, vises de på skærmen. Efterhånden som De skriver, bevæger markøren sig mod højre. Laver De en indtastningsfejl, kan de flytte markøren mod venstre og rette fejlen ved enten at trykke på INST/DEL tasten eller CTRL-H. CTRL er en forkortelse for CONTROL tasten. For at angive en kontrolkarakter holdes CTRL tasten nede og der trykkes på den passende bogstavtast. (En fortegnelse over kontrolkarakterer og deres anvendelse findes i afsnit 13).

Kommando-nøgleord og -tilføjelser kan indtastes i en hvilken som helst kombination af store og små bogstaver. CP/M 3.0 opfatter alle tegn i en kommandolinie som store bogstaver.

Generelt skal De indtaste en kommandolinie direkte efter systemmeldingen.
CP/M 3.0 tillader dog mellemrum mellem meldingen og kommando-nøgleordet.

KOMMANDOTYPER

CP/M 3.0 skelner mellem to forskellige kommandotyper: indbyggede kommandoer og ydre utility kommandoer. Indbyggede kommandoer udfører kommandoer, som findes i hukommelsen som en del af CP/M operativsystemet. Indbyggede kommandoer kan udføres omgående. Utility kommandoer er lagret på disketten som programfiler. De skal indlæses fra disketten for at udføre deres opgaver. De kan finde ydre utility programfiler, når et Directory vises på skærmen, fordi deres filnavne er efterfulgt af et punktum samt COM (.COM).
Afsnit 14 indeholder en fortegnelse over indbyggede og ydre utility kommandoer.

Ved ydre utility kommandoer kontrollerer CP/M 3.0 kun kommando-nøgleordet.
Mange utilities kræver særlige tilføjelser. Medtager De en kommando-tilføjelse, videresender CP/M 3.0 den uden yderligere kontrol. En kommando-tilføjelse kan højst indeholde 128 karakterer.

HVORDAN CP/M LÆSER KOMMANDOLINIER

Lad os benytte DIR kommandoen til at demonstrere, hvordan CP/M læser kommandolinier. DIR, som er en forkortelse for Directory, giver CP/M besked på at vise en fortegnelse over diskettefiler på skærmen. Indtast DIR nøgleordet efter systemmeldingen og tryk på RETURN :

A⟩DIR RETURN

CP/M svarer på denne kommando ved at vise navnene på de filer, som er lagret på den diskette, der sidder i drev A. Hvis det f.eks. drejer sig om CP/M systemdisketten, fremkommer en

liste som denne med filnavne på Deres skærm.

A:PIP COM:ED COM:CCP COM:HELP COM:HELP HLP
A:DIR COM:CPM SYS

CP/M 3.0 genkender kun korrekt stavede kommando-nøgleord. Hvis De laver en stavefejl og trykker på RETURN, før De har opdaget Deres stavefejl, gentager CP/M 3.0 kommandolinien, efterfulgt af et spørgsmålstegn. Staver De f.eks. kommandoen forkert som i følgende eksempel:

A⟩DJR RETURN

vil CP/M 3.0 svare med:

DJR?

Dette fortæller Dem, at CP/M ikke kan finde en kommando, som staves DJR. For at rette sådanne indtastningsfejl, kan De bruge INST/DEL tasten for at slette forkerte bogstaver. En anden måde at rette på er, at holde CTRL tasten nedtrykket og trykke på H for at flytte markøren mod venstre. CP/M omfatter et antal andre kontrolkarakterer, som kan hjælpe Dem med at redigere kommandolinier. Afsnit 13 fortæller om brugen af kontrolkarakterer ved rettelse af kommandolinier og anden information, De indtaster på Deres konsol.

DIR accepterer et filnavn som kommando-tilføjelse. De kan bruge DIR med et filnavn for at se, om en bestemt fil findes på disketten. For, for eksempel, at kontrollere eksistensen af programfilen MINFIL, indtastes:

A⟩DIR MINFIL RETURN

CP/M 3.0 udfører denne opgave ved enten at vise det specificerede filnavn eller meddelelsen:

No File

Vær sikker på at De laver mindst eet mellemrum mellem kommandoens nøgleord og tilføjelsen. Gør De ikke det, vil CP/M 3.0 svare følgende:

A⟩DIRMINFIL RETURN
DIRMINFIL?

HVORDAN DE KOPIERER DERES CP/M 3.0 DISKETTE

Før De gør noget som helst andet, bør De lave en kopi af Deres CP/M systemdiskette. Dette kan gøres ved anvendelse af enten eet eller to diskettedrev. Hvis De benytter to drev, kan disse være 1541ere, 1571ere, eller et af hver. De benyttede disketter kan være nye eller brugte. De kan enten formattere nye disketter eller omformattere brugte. Kopiering foretages ved anvendelse af de FORMAT og PIP utility programmer, der findes på Deres CP/M systemdiskette.

FORMATTERING AF EN DISKETTE

Formatter disketten med FORMAT programmet, enten som en C128 enkeltsidet (hvis De bruger en 1541) eller en C128 dobbeltsidet (hvis De har en 1571).
Den enkeltsidede version benyttes til formattering af disketter, som skal være kompatible med de CP/M 2.2 programpakker, som engang blev solgt til Commodore 64.

Indtast kommandoen FORMAT, vælg den rigtige diskettetype med 'markør-nedtasten', tryk på RETURN og følg instruktionerne, som bliver givet på skærmen. Tryk på Y (ja) eller N (nej) som svar på spørgsmålet "Do you want to format another disk?" (ønsker De at formattere en diskette mere?).

KOPIERING AF FILER

Brug PIP (Peripheral Interchange Program) - som findes på side 2 på den originale diskette - til filkopiering. Indtast PIP, hvorved den sædvanlige systemmelding (A)) vil blive erstattet af PIP meddelelsestegnet (*).

Hvis De benytter et enkelt diskettedrev, skal De bruge drev A som kildedrevet og drev E som destinationsdrev. Drev E kaldes et virtuelt drev d.v.s. at det ikke eksisterer i form af et stykke hardware. Sæt den diskette, hvorfra der skal kopieres i diskettedrevet og indtast E:=A:*.*. Systemet vil selv meddele, hvornår De skal skifte mellem disketterne.

Hvis De bruger to diskettedrev, skal De sætte kildedisketten i drev A (som er enhed nummer 8) og den netop formatterede

diskette i drev B (enhed nummer 9 - hvilket nummer gives ved at sætte kontakten bag på Commodore 1571 i nederste stilling, mens der er slukket for strømmen) og indtaste B:=A:*.*, hvilket vil betyde, at alle filer bliver kopieret.

Deres originale CP/M diskette er en såkaldt "flippy" - hvilket vil sige, at den er optaget, som var den to enkelte disketter - så den må tages ud af diskettestationen og vendes for at kunne læses på side 2. Dette er nødvendigt på en 1541 diskettestation, som indeholder et enkeltsidet diskettedrev. Hvis De har en 1541 diskettestation, bør De derfor kopiere de to sider på originaldisketten til to separate disketter. Har De imidlertid en 1571, vil det være mere hensigtsmæssigt at kopiere begge sider til en standard dobbeltsidet diskette; når første side er kopieret, vendes originaldisketten og den anden side kopieres til samme destinationsdiskette ved atter at give kopieringsinstruktionen som svar på PIP meldingen.

Engang imellem vil De måske ønske at kunne fremstille disketter, hvorpå kun systemfilerne findes. Hertil kan de bruge PIP til at foretage kopiering af filerne CPM+.SYS og CCP.COM til en nyformatteret diskette.

OBS! Det er kun nødvendigt at have disse to filer på disketter, som De vil bruge for indlæsning af CP/M - på andre disketter betyder de kun spild af plads.

Har De et enkelt diskettedrev, skal De taste E:=A:CPM+.SYS for at kopiere den første fil og E:=A:CCP.COM for at kopiere den anden fil.

Når De er færdig med at bruge PIP, skal De trykke på RETURN for at komme fra PIP systemet til systemmeldingen.
Fuld beskrivelse af PIP kan fås ved at taste HELP PIP, HELP PIP OPTIONS og HELP PIP EXAMPLES.

SPROG OG APPLIKATIONS SOFTWARE

CP/M er blot et operativsystem, det vil sige et middel og ikke et mål i sig selv. Det kan ikke i sig selv udføre noget. Hvis De ønsker at skrive Deres egne programmer, har De brug for et sprog, enten et assembler sprog eller et højniveau sprog,

hvori de kan skrives. Ønsker De at bruge spil eller at udføre forretningsmæssig databehandling, vil De behøve applikationsprogrammer.

HVAD SKAL MAN KØBE?

Da CP/M er blevet implementeret på stort set alle computere, som benytter en Intel 8080 eller Zilog Z80 centralenhed, findes der et meget stort udbud af software, som kan udføres med CP/M systemer. Det mest omfattende katalog over færdigfremstillet software er "CP/M Software Finder", som er udgivet for Digital Research af Que Corporation (ISBN 0-88-022-021-X), og som kan anskaffes gennem gode softwareforhandlere. Da det ikke er overkommeligt for CP/M Users Group at vedligeholde sit bibliotek med Commodore formatdisketter, er der, af Independent Products Users Group (ICPUG), fremstillet en samling af offentligt tilgængeligt CP/M software. Formular for medlemsskab kan rekvireres ved indsendelse af frankeret svarkuvert til Membership Secretary, ICPUG, 30 Brancaster Road, Newbury Park, Ilford, Essex, IG2 7EP, England.

For optagelse på disketter benytter CP/M normalt Modified Frequency Modulation (MFM). Commodore DOS anvender normalt Group Code Recording (GCR).
Commodore 1571 kan læse begge, men 1541 kan kun læse GCR. De CP/M programmer der er i handelen leveres udelukkende som MFM disketter. Selv inden for MFM findes mange forskellige formater; 1571 kan læse disketter i følgende formater:

Epson QX10
(512 byte sektorer, dobbeltsidet, 10 sektorer pr. spor)
IBM-8 SS (CP/M-86)
(512 byte sektorer, enkeltsidet, 8 sektorer pr. spor)
IBM-8 DS (CP/M-86)
(512 byte sektorer, dobbeltsidet, 8 sektorer pr. spor)
IBM-9 SS (CP/M-86)
(512 byte sektorer, enkeltsidet, 9 sektorer pr. spor)
IBM-9 DS (CP/M-86)
(512 byte sektorer, dobbeltsidet, 9 sektorer pr. spor)
KayPro II
(512 byte sektorer, enkeltsidet, 10 sektorer pr. spor)

KayPro IV
(512 byte sektorer, dobbeltsidet, 10 sektorer pr. spor)
Osborne DD SS
(1024 byte sektorer, enkeltsidet, 5 sektorer pr. spor)
Osborne DD DS
(1024 byte sektorer, dobbeltsidet, 5 sektorer pr. spor)

De må derfor, når De køber CP/M software, passe på kun at købe det på disketter i et af de nævnte formater. Husk endvidere, at Deres Commodore 128 kan køre software som er skrevet til udførelse under enten CP/M 2.2 eller CP/M Plus (Plus er det nyere navn for, hvad der tidligere hed version 3). CP/M-86 er imidlertid den CP/M version, som er beregnet for anvendelse på 16 bit processorer; CP/M-86 kan ikke afvikles på Deres C128 8 bit Z80 processor, selvom 1571 drevet er i stand til at læse CP/M-86 datafiler.

Hvis De har en 1541 diskettestation, vil det være nødvendigt at få den software, De køber, overført fra MFM format til GCR format. Nogle software forhandlere vil muligvis påtage sig denne opgave, men De må regne med at skulle betale for ydelsen. Visse steder råder den lokale ICPUG, hvor en sådan findes, over faciliteter til dette formål.

HVORDAN INSTALLERES DET PÅ DERES C128?

Fordi CP/M operativsystemet anvendes af så mange forskellige computere, må mange CP/M programmer konfigureres specielt til den maskintype, hvorpå de skal bruges. Fremgangsmåden ved installering af et program på Deres C128 omfatter indstilling af parametere i softwaren. Hvis det er nødvendigt, vil programmets brugervejledning indeholde de nødvendige oplysninger herom. De fleste programmer indeholder en liste over, hvilke terminaler programmet kan anvende. Er det ikke tilfældet, vil De skulle gøre det efter det enkelte systems krav.

Herunder vises de oplysninger, der skal angives ved kørsel af WINSTALL.COM (det installationsprogram, som er en del af Wordstar pakken). Her vil man også kunne se de oplysninger, som kræves ved installation af andre programmer, selv om alle pakker ikke stiller de samme spørgsmål.

Terminalnavn	Commodore 128
Skærmformat	
Skærmhøjde	24
Skærmbredde	80
Markørpositionering	
Funktionskode sekvens	1Bh 3Dh
Karakterer til afsendelse mellem linienummer og kolonnenummer.	none
Karakterer til afsendelse efter linienummer og kolonnenummer.	none
Sendes kolonnenummer før linienummer?	NO
Hvilken karakter sendes til terminalen for at angive linie 1?	20h
Hvilken karakter sendes til terminalen for at angive kolonne 1?	20h
Hvilke kodetyper sendes for angivelse af linie- og kolonnenumre?	Single byte BINARY value
Terminal opstart	
Funktionskode sekvens	1Bh 59h 1Bh 1Bh 1Bh 60h
Terminal nedlukning	
Funktionskode sekvens	none
Highlight-on	
Funktionskode sekvens	1Bh 1Bh 1Bh 52h
Highlight-off	
Funktionskode sekvens	1Bh 1Bh 1Bh 51h
Slet til linieslut	1Bh 54h
Fjern linie	1Bh 52h
Indsæt linie	1Bh 45h
Bruger Deres terminal sidste karakter på skærmen som scroll kommando?	YES

De fleste Commodore printere skal installeres som Standard Printere UDEN kommunikations protokol og uden angivelse af enhedsnummer for Printer Driver.

OBS! "h" efter de viste tal indikerer, at der er tale om hexidecimale tal (som bruger basen 16 i stedet for decimalbasen 10).

AFSNIT 12
FILER, DISKETTER OG DREV I CP/M 3.0

HVAD ER EN FIL?

En af CP/M's vigtigste opgaver er at få tilgang til og vedligeholde filer på Deres disketter. Filer er i CP/M fundamentalt det samme som i C128 og C64 mode - d.v.s., de er samlinger af oplysninger. CP/M håndterer imidlertid filer noget forskelligt fra, hvordan C128 og C64 mode gør. Dette afsnit fortæller om de to forskellige filtyper, som bruges i CP/M; om hvorledes man danner, navngiver og får adgang til en fil; og beskriver, hvorledes filer lagres på Deres CP/M disketter.

Som nævnt ovenfor, er en CP/M 3.0 fil en samling af oplysninger. Hver fil skal have et særligt navn, hvorved CP/M kan identificere filen. På disketten lagres også et Directory, som indeholder en fortegnelse over de filnavne, som findes på disketten, samt den enkelte fils placering på disketten.

Der findes to slags CP/M filer: PROGRAM (kommando) filer og DATA filer. En programfil indeholder en serie instruktioner, som computeren følger trin for trin for at opnå et eller andet ønsket resultat. En datafil er normalt en samling af sammenhørende oplysninger (en liste med navne og adresser, en lagerliste, bogholderiposteringer, dokumenttekster e.l.).

DANNELSE AF EN FIL

En CP/M fil kan dannes på flere måder. Een måde er at bruge en tekst editor. CP/M tekst editoren ED bruges for at oprette og navngive en fil. De kan også oprette en fil ved at kopiere en eksisterende fil til en ny lokation; under denne proces kan filen omdøbes. I CP/M kan PIP kommandoen bruges til at kopiere og omdøbe filer. Endelig danner visse programmer (som f.eks. MAC, et CP/M maskinsprogsprogram) uddatafiler efterhånden som de behandler inddatafiler.

ED og PIP kommandoerne gennemgås summarisk i afsnit 14, sammen med andre almindeligt anvendte CP/M kommandoer. Detaljer om disse og alle andre CP/M 3.0 kommandoer kan findes i CP/M Plus User's Guide, som De kan købe ved Deres Commodore forhandler.

NAVNGIVNING AF EN FIL

FILANGIVELSE

Enhver fil genkendes af CP/M ud fra en specifik filangivelse. En filangivelse kan bestå af fire dele: en drevangivelse, et filnavn, en filtype og en adgangskode. Den eneste krævede del er filnavnet.

DREVANGIVELSE

Drevangivelsen består af et enkelt bogstav (A-P) efterfulgt af et kolon.
Hvert drev i Deres system har et bestem bogstav. Når De medtager en drevangivelse som en del af filangivelsen, giver De CP/M besked på at lede efter filen på det specificerede drev. Hvis De for eksempel indtaster:

B:MINFIL RETURN

vil CP/M lede på drev B efter filen MINFIL. Udelader De drevangivelsen, vil CP/M 3.0 lede efter filen på det drev, som har standardværdien (A).

FILNAVN

En filnavn kan være fra en til otte karakterer langt, som f.eks.:

MINFIL

En filangivelse kan bestå af kun et filnavn. Når De bestemmer filnavne, prøv da at lade navnet indeholde oplysning om, hvad filen indeholder. Har De f.eks. en liste med kundenavne for Deres forretning, kunne De kalde filen for:

KUNDER

så navnet giver Dem en ide om, hvad filen indeholder.

FILTYPE

For at hjælpe Dem med at identificere filer, som tilhører samme kategori, tillader CP/M Dem at tilføje en evt. en-til-tre karakters oplysning, kaldet en filtype, til filnavnet. Når De tilføjer filnavnet en filtype, skal filnavn og filtype adskilles med et punktum. Prøv at benytte bogstaver, som siger noget om filens art. De kunne

f.eks. tilføje følgende filtype til den fil, som indeholder Deres kundeliste:

KUNDER.NAV

Når CP/M viser filspecifikationer, sætter den 'blanke' positioner til korte filnavne, så De hurtigt kan sammenligne filtyper. De programfiler, som CP/M indlæser til hukommelsen fra diskette har filtypen COM.

ADGANGSKODE

I CP/M 3.0 på Commodore 128 kan De indsætte en adgangskode som en del af filangivelsen. Adgangskodens længde kan variere fra een til otte karakterer. Hvis De medtager en adgangskode, skal den adskilles fra filtyper (eller filnavnet, hvis filtype ikke bruges) med et semikolon, som her:

KUNDER.NAV;KONTO

En adgangskode er en mulighed. Er en fil imidlertid blevet beskyttet med en adgangskode, SKAL koden inddateres som en del af filangivelsen for at kunne komme i kontakt med filen.

EKSEMPEL PÅ EN FILANGIVELSE

En filangivelse indeholdende alle fire mulige elementer består af drevangivelse, et primært filnavn, en filtype og en adgangskode, alle adskilt af skilletegn som i følgende eksempel:

A:DOKUMENT.LOV;SUSAN RETURN

BRUGERNUMMER

CP/M 3.0 identificerer endvidere alle filer ved at føje et brugernummer, som ligger i intervallet fra 0 til 15, til hver fil,. CP/M 3.0 føjer brugernummeret til en fil, når den dannes. Brugernumre tillader Dem at inddele Deres filer i 16 filgrupper.

Brugernummeret står altid foran drevangivelsen, undtagen for bruger 0, som er standardværdien, der ikke vises i meldingen. Her er nogle eksempler på brugernumre og deres betydning:

4A⟩ - Brugernummer 4, drev A
A⟩ - Brugernummer 0, drev A
2B⟩ - Brugernummer 2, drev B

De kan bruge den indbyggede USER kommando til at ændre det nuværende brugernummer således:

A⟩ USER 3 RETURN
3A⟩

De kan ændre både brugernummeret og drevnummeret ved samtidig at indtaste det nye brugernummer og den nye drevangivelse:

A⟩3B: RETURN
3B⟩

De fleste kommandoer har kun adgang til de filer, som har det aktuelle brugernummer. Hvis imidlertid en fil tilhører bruger 0 og er markeret med en systemfil-attribut, kan den hentes af ethvert brugernummer.

BRUG AF SØGEKARAKTERER FOR SØGNING AF MERE END EEN FIL

Visse indbyggede og ydre CP/M 3.0 kommandoer kan udvælge og behandle flere filer, når specielle søgekarakterer medtages i filnavnet eller filtypen.
En søgekarakter er en karakter, som kan benyttes i stedet for andre karakterer. CP/M 3.0 bruger asterisken (*) og spørgsmålstegnet (?) som søgekarakterer. Bruger De for eksempel et ? som tredie tegn i et filnavn, giver De CP/M besked på at lade spørgsmålstegnet erstatte en hvilken som helst karakter, som opdages i denne position. På samme måde instruerer en * CP/M om at udfylde filnavnet med ? som angivet. En filangivelse indeholdende søgekarakterer kaldes en flertydig filangivelse. Den kan referere til mere end een fil, fordi den giver CP/M 3.0 et mønster til sammenligning. Taster De for eksempel:

?????TAX.LIB

vil CP/M 3.0 udvælge alle filer, hvis navn ender på TAX og hvis filtype er .LIB.

RESERVEREDE KARAKTERER

Karaktererne i tabel 12-1 har en speciel betydning i CP/M 3.0, så anvend ikke disse karakterer i filangivelser, undtagen i nævnte tilfælde.

Tabel 12-1 - CP/M 3.0 Reserverede Karakterer

⟨$,!⟩[]	skilletegn for filangivelser
tab space	
vognretur	
:	drev skilletegn i filangivelse
.	filtype skilletegn i filangivelse
;	adgangskode skilletegn i filangivelse
;	kommentar skilletegn i begyndelse af en kommandolinie
* ?	søgekarakterer for flertydig filangivelse
⟨⟩&!] + -	mulige liste skilletegn
[]	mulige liste skilletegn for globalt og lokalt brug
()	skilletegn for flere kriterier i skarpe paranteser for muligheder med flere modifikationer
/$	mulige skilletegn i en kommandolinie

RESERVEREDE FILTYPER

CP/M 3.0 har allerede etableret flere filgrupper. Tabel 12-2 viser nogle af disses filtyper med en kort forklaring af hver.

Figur 12-2 - CP/M 3.0 Reserverede Filtyper

Filtype	Betydning
ASM	Assembler source fil
BAS	BASIC source program
COM	Z80 eller lignende maskinsprogs program
HEX	Uddatafil fra MAC (bruges af HEXCOM)
HLP	HELP meddelelses fil
$$$	Midlertidig fil
PRN	Printfil fra MAC eller RMAC
REL	Uddatafil fra RMAC (bruges af LINK)
SUB	Liste over kommandoer som skal udføres af SUBMIT
SYM	Symbolfil fra MAC, RMAC eller LINK
SYS	Systemfil

AFSNIT 13
BRUG AF KONSOL OG PRINTER I CP/M 3.0

Dette afsnit gennemgår, hvorledes CP/M 3.0 kommunikerer med Deres konsol og printer. Det fortæller, hvordan man starter og stopper konsol- og printeruddata, og om redigeringskommandoer, De kan indtaste på Deres konsol.

STYRING AF KONSOL UDDATA

Somme tider viser CP/M 3.0 information på skærmen så hurtigt, at det kan være svært at læse det. For at få systemet til at vente, mens De læser, hvad der står på skærmen, kan De holde CTRL-tasten nedtrykket og trykke på S. En CTRL-S indtastnings sekvens får skærmbilledet til at blive stående. Når De er klar til at fortsætte, trykkes CTRL-Q. Nedtrykning af NO SCROLL tasten vil også få systemet til at holde en pause og anbringe et pausevindue på statuslinien i linie 25 på skærmen. For at fortsætte trykkes igen på NO SCROLL. Hvis De under en pause trykker på andre taster end CTRLQ eller NO SCROLL, ringer CP/M 3.0 med konsolklokken.

Nogle CP/M 3.0 utilities (som DIR og TYPE) understøtter automatisk sideskift på konsollen. Dette betyder, at hvis uddata fra programmet er længere end skærmen kan vise på een gang, standses visningen automatisk, når skærmen er fyldt. Hvis dette sker, beder CP/M 3.0 Dem om at trykke på RETURN for at fortsætte. Denne facilitet kan slås til og fra med SETDEF kommandoen.

STYRING AF PRINTER UDDATA

De kan også bruge en kontrolkommando til at 'echo' (vise) konsollens uddata på printeren. For at starte printer 'echo' tastes CTRL-P. Der gives en beep-lyd for at fortælle Dem, at 'echo' er slået til. For at standse trykkes atter på CTRL-P. Mens printer-echo er i funktion, bliver ethvert tegn, som vises på skærmen, også udskrevet på printeren.

De kan bruge printer 'echo' sammen med en DIR kommando for at få en udskrift af indholdsfortegnelsen på en diskette. De kan også bruge CTRL-P sammen med CTRL-S og CTRL-Q for at få en udskrift af en del af en fil.Brug TYPE kommandoen til at starte visning af filen på konsollen. Når skærmbilledet når til det, De ønsker at få udskrevet, trykkes på CTRL-S for at

standse visningen, CTRL-P for at tilkoble printeren, og derefter CTRL-Q for at fortsætte fremvisning på skærmen og begynde udskrivning. De kan bruge en anden CTRL-S, CTRL-P, CTRL-Q sekvens for at afbryde printer 'echo'.

KONSOLLINIE REDIGERING

Som tidligere nævnt, kan De rette simple indtastningsfejl ved at bruge INST/DEL tasten eller CTRL-H. CP/M 3.0 indeholder også yderligere linieredigerings funktioner, som kan udføres med kontrolkarakterer. De kan bruge kontrolkaraktererne til at redigere kommandolinier eller inddatalinier til de fleste programmer.

BRUG AF KONTROLKARAKTERER FOR LINIEREDIGERING

Ved anvendelse af de linieredigerings kontrolkarakterer, der er vist i tabel 13-1, kan De flytte markøren mod venstre og højre for at indsætte eller slette karakterer i midten af en kommandolinie. På denne måde behøver De ikke at omtaste alt, hvad der står til højre for Deres rettelse.

I følgende eksempel staver brugeren PIP forkert, og CP/M 3.0 returnerer en fejlmeddelelse. Brugeren genkalder den fejlramte kommandolinie ved at taste CTRL-W og rette fejlen (understregning repræsenterer markøren):

A⟩POP A:=B:*.*	PIP fejlstavet
POP?	
A⟩POP A:=B:*.*	CTRL-W genkalder linien
A⟩POP A:=B:*.*	CTRL-B flytter markøren til liniens start
A⟩POP A:=B:*.*	CTRL-F flytter markøren mod højre
A⟩PP A:=B:*.*	CTRL:G sletter det forkerte tegn
A⟩PIP A:=B:*.*	Bogstavet I retter kommandonavnet

Efter at kommandolinien er rettet, kan brugeren nedtrykke RETURN, selv om markøren befinder sig midt på linien. Tryk på RETURN (eller en tilsvarende kontrolkarakter) udfører ikke blot kommandoen, men lagrer også kommandoen i en buffer (mellemlager), så den kan genkaldes for redigering eller genanvendelse ved at trykke på CTRL-W.

Hvis De indsætter en karakter midt på en linie, vil de karakterer, som findes til højre for markøren, blive flyttet mod højre. Overstiger liniebredden skærmens bredde, vil karaktererne forsvinde ud af skærmens højre side. Disse karakterer er ikke gået tabt. De kommer igen, hvis de sletter karakterer i linien eller hvis De trykker på CTRL-E, mens markøren befinder sig midt på linien. CTRL-E flytter alle karakterer til højre for markøren til næste linie på skærmen.

Tabel 13-1 viser en komplet liste over linieredigerings kontrolkarakterer for CP/M 3.0 systemet på Commodore 128.

Tabel 13-1 - CP/M 3.0 Linieredigerings kontrol karakterer

Karakter	Betydning
CTRL-A	Flytter markøren en karakter mod venstre
CTRL-B	Flytter markøren til kommandoliniens begyndelse uden at ændre liniens indhold. Befinder markøren sig i liniens start, flytter CTRL-B den til liniens slutning.
CTRL-E	Gennemtvinger en fysisk vognretur, men sender ikke kommandolinien til CP/M 3.0. Flytter markøren til næste linies begyndelse uden at slette nuværende inddata.
CTRL-F	Flytter markøren en karakter mod højre
CTRL-G	Sletter den karakter, der er under markøren. Markøren flyttes ikke. Karakterer på markørens højre side flytter en plads mod venstre.
CTRL-H	Sletter karakteren til venstre for markøren og flytter markøren en karakterposition mod venstre. Karakterer på markørens højre side flytter en plads mod venstre.
CTRL-I	Flytter markøren til næste tabulatorstop. Tabulatorstop sættes automatisk ved hver ottende kolonne. Har samme effekt som tryk på TAB tasten.
CTRL-J	Sender kommandolinie til CP/M 3.0 og returnerer markøren til begyndelsen af en ny linie. Har samme effekt som tryk på RETURN eller CTRL-M.
CTRL-K	Sletter liniens indhold fra markøren og frem.

Tabel 13-1 - CP/M 3.0 Linieredigerings kontrol karakterer

Karakter	Betydning
CTRL-M	Sender kommandolinie til CP/M 3.0 og returnerer markøren til begyndelsen af en ny linie. Har samme effekt som tryk på RETURN eller CTRL-J.
CTRL-R	Genskriver kommandolinien. Placerer en #-karakter i den aktuelle markørposition, flytter markøren til næste linie og genskriver enhver del af en kommando, De har skrevet til nu.
CTRL-U	Kasserer alle karakterer i kommandolinien, placerer en #-karakter i den aktuelle markørposition og flytter markøren til næste linie. De kan imidlertid bruge en CTRL-W til at genkalde enhver af de karakterer, som stod til venstre for markøren, da De tastede CTRL-U.
CTRL-W	Genkalder og viser en tidligere indtastet kommando-linie såvel på operationssystem niveau som under udførelse af programmer, hvis CTRL-W er den først indtastede karakter efter systemmeldingen ⟩. CTRL-J, CTRL-M, CTRL-U og RETURN definerer den kommandolinie, De kan genkalde. CTRL-W flytter markøren til slutningen af komman-dolinien. Hvis De trykker på RETURN, udfører CP/M 3.0 den genkaldte kommando.
CTRL-X	Kasserer alle karakterer til venstre for markøren og flytter denne til den aktuelle linies begyndelse. CTRL-X lagrer enhver karakter, som står til højre for markøren.

AFSNIT 14
SUMMERING AF DE VIGTIGSTE CP/M 3.0 KOMMANDOER

Som anført i afsnit 11 består en CP/M 3.0 kommandolinie af et kommandonøgleord, en mulig tilføjelse samt et anslag på RETURN. Dette afsnit beskriver de to kommandotyper som kan identificeres af kommando-nøgleordet, og summerer individuelle kommandoer og deres funktioner. Afsnittet giver endvidere eksempler på nogle almindeligt anvendte kommandoer. Samtidig forklarer afsnittet konceptet på logiske og fysiske enheder under CP/M 3.0. Derefter fortælles, hvorledes CP/M 3.0 søger en programfil på en diskette, hvorledes flere sammenhængende kommandoer behandles og hvordan man sætter et diskettesystem tilbage til opstartsværdien. Til sidst forklares i afsnittet, hvordan HELP kommandoen bruges for at få oplysninger om forskellige CP/M emner, omfattende kommandoformater og brug af disse på tastaturet.

DE TO CP/M 3.0 KOMMANDOTYPER

I CP/M 3.0 findes to kommandotyper:

- **Indbyggede kommandoer** - som er programmer i hukommelsen
- **Ydre utility kommandoer** - som er programfiler på en diskette.

CP/M 3.0 har seks indbyggede kommandoer og mere end 20 ydre utility kommandoer. De kan tilføje Deres system flere utilities ved at købe forskellige CP/M 3.0 kompatible applikationsprogrammer. Er De en øvet programmør, kan De også skrive Deres egne utilities, som kan arbejde sammen med CP/M 3.0.

INDBYGGEDE KOMMANDOER

Indbyggede kommandoer er dele af CP/M 3.0, som altid er tilgængelige for brug, uanset hvilken diskette, der sidder i hvilket drev. Indbyggede kommandoer lægges i computerens hukommelse, når CP/M 3.0 indlæses, og udføres derfor hurtigere end ydre utility kommandoer. Tabel 14-1 indeholder de indbyggede kommandoer i Commodore 128 CP/M 3.0.

Nogle indbyggede kommandoer har mulige parametere, som kræver bistand fra en tilsvarende ydre utility. Den tilsvarende ydre utility-kommando har samme navn som den indbyggede og har filtypen COM.

Tabel 14-1 - Indbyggede kommandoer

Kommando	Funktion
DIR	Viser filnavne på alle filer i indholdsfortegnelsen (directory) undtaget dem, som er markeret SYS.
DIRSYS	Viser filnavne på filer markeret med SYS i directory
ERASE	Sletter et filnavn i directory og frigør den lagerplads, som filen beslaglagde.
RENAME	Omdøber en diskettefil
TYPE	Viser indholdet af en ASCII tekstfil på skærmen
USER	Skifter til et andet brugernummer

YDRE UTILITY KOMMANDOER

De ydre utility kommandoer i CP/M 3.0 er anført i tabel 14-2. Hvis De indtaster et kommando-nøgleord, som identificerer en ydre utility, indlæser CP/M 3.0 programfilen fra disketten og giver denne fil ethvert filnavn samt de data eller parametere, De indtaster i kommando-tilføjelsen.

DIR, RENAME og TYPE er indbyggede kommandoer med mulige ydre udvidelser.

Tabel 14-2 - Ydre Utility Kommandoer

Navn	Funktion
DATE	Indstiller eller viser dato og tidspunkt
DEVICE	Tildeler en eller flere fysiske enheder 'logiske' CP/M enheder, ændrer enhedsprotokol og baud rates (overførselshastigheder) eller indstiller konsolskærmens størrelse.
DIR	Viser directory med filer og deres karakteristika.
DUMP	Viser en fil i ASCII eller hexadecimalt format.
ED	Danner og ændrer ASCII filer
ERASE	Bruges til sletning med søgekarakterer
GENCOM	Opretter en speciel COM fil med tilhørende RSX fil.

GET	Får midlertidigt inddata fra en diskettefil i stedet for fra tastaturet.
HELP	Viser information om brug af CP/M 3.0 kommandoer.
INITDIR	Initialiserer et diskette directory for tid og dato registrering.
KEYFIG	Tillader ændring af definition på tastaturets taster.
PATCH	Viser eller opretter instruktioner for CP/M systemet.
PIP	Kopierer og kombinerer filer
PUT	Dirigerer midlertidigt printer- og konsoluddata til en diskettefil.
RENAME	Ændrer et filnavn, eller en filgruppe, ved hjælp af søgekarakterer.
SAVE	Kopierer hukommelsens indhold til en fil.
SET	Indstiller filmuligheder omfattende diskettenavn, filattributter, type på tid og dato registrering samt adgangskode beskyttelse.
SETDEF	Indstiller systemmuligheder incl. drev søgekæde.
SHOW	Viser statistik over diskette og diskettedrev
SUBMIT	Udfører automatisk flere sammensatte kommandoer
TYPE	Viser tekstfilers indhold (eller filgruppers, hvis der bruges søgekarakterer) på skærmen (og om ønsket printer).

OMDIRIGERING AF INDDATA OG UDDATA

PUT kommandoen i CP/M 3.0 sætter Dem i stand til at dirigere konsol- eller printeruddata til en diskettefil. De kan bruge en GET kommando for at få CP/M 3.0 eller et utility program til at tage konsolinddata fra en diskettefil. Følgende eksempler illustrerer nogle af de muligheder, GET og PUT tilbyder.

Med en PUT kommando kan De, lige så vel som til konsollen, dirigere konsoluddata til en diskettefil. Med PUT kan De danne en diskettefil indeholdende en indholdsfortegnelse over alle filer på disketten, som vist i figur 14-1.

Figur 14-1 - PUT Kommando Eksempel

```
A)PUT CONSOLE OUTPUT TO FILE
DIR.PRN PUTTING CONSOLE OUTPUT TO FILE:DIR.PRN

A)DIR
A:FILENAME TEX:FRONT  TEX:FRONT   BAK:ONE     BAK:THREE  TEX
A:FOUR     TEX:ONE    TEX:LINEDIT TEX:EXAMP1  TXT:TWO    BAK
A:TWO      TEX:THREE  BAK:EXAMP2  TXT
A)TYPE DIR.PRN
A:FILENAME TEX:FRONT  TEX:FRONT   BAK:ONE     BAK:THREE  TEX
A:FOUR     TEX:ONE    TEX:LINEDIT TEX:EXAMP1  TXT:TWO    BAK
A:TWO      TEX:THREE  BAK:EXAMP2  TXT
```

En GET kommando kan instruere CP/M 3.0 eller et program om at tage konsolinddata fra en diskettefil i stedet for fra tastaturet. Hvis filen skal læses af CP/M 3.0, skal den indeholde standard CP/M 3.0 kommandolinier.
Skal filen læses af et utility program, skal den indeholde inddata, som passer til dette program. En fil kan indeholde både CP/M 3.0 kommandolinier og program inddata, hvis den også indeholder en kommando, som starter et program.

TILDELING AF LOGISKE ENHEDER

Mindste konfiguration af Commodore 128 CP/M 3.0 hardware omfatter en konsol bestående af et tastatur og en skærm, samt en 1571 diskettestation. De ønsker muligvis at udvide denne konfiguration med andre enheder, som for eksempel en printer eller et modem. Som hjælp til at holde styr på disse forskellige fysiske inddata og uddata enheder, indeholder tabel 14-3 navnene på CP/M 3.0 logiske enheder. Den viser også sammenhængen mellem de fysiske og logiske enheder i Commodore 128 CP/M 3.0 systemet.

Figur 14-3 - CP/M 3.0 Logiske Enheder

Logisk enhedsnavn	Enhedstype	Fysisk enhed
CONIN:	Konsolinddata	Tastatur
CONOUT:	Konsoluddata	80 kolonners skærm
AUXIN:	Ydre inddata	Ingen
AUXOUT:	Ydre uddata	Ingen
LST:	Uddataliste	PTR1 eller PTR2

Disse tildelinger kan ændres med en DEVICE kommando. De kan, f.eks., tildele AUXIN og AUXOUT til et modem, så Deres computer kan kommunikere med andre computerbrugere over telefonnettet.

SØGNING AF PROGRAMFILER

Hvis et kommando-nøgleord skrives, søger CP/M 3.0 efter denne programfil på et drev med standardværdien eller en angivet værdi. Der søges under korrekt brugernummer, og derefter under bruger 0 for samme fil markeret med SYS attributten. Når som helst under søgeprocessen standser CP/M 3.0 søgningen, hvis programfilen findes. CP/M 3.0 indlæser derefter programmet til hukommelsen og udfører det. Når programmet afsluttes, viser CP/M 3.0 systemmeldingen og venter på Deres næste kommando. Finder CP/M 3.0 imidlertid ikke kommandofilen, gentages kommandolinien med et spørgsmåltegn efter, og venter på næste kommando.

UDFØRELSE AF FLERE KOMMANDOER

I de hidtil nu viste eksempler har CP/M 3.0 kun udført een kommando ad gangen. CP/M 3.0 kan også udføre en sekvens af kommandoer. Ved systemmeldingen kan De indtaste en kommandosekvens, eller De kan anbringe en nødvendig kommandosekvens i en diskettefil, idet filtypen SUB benyttes.
Hvis De har lagret en sekvens i en diskettefil, kan De når som helst udføre sekvensen med SUBMIT kommandoen.

AFBRYDELSE AF PROGRAMMER

Programudførelse eller nulstilling af diskettesystemets opstartsværdi kan udføres med kommandoen CTRL-C. En CTRL-C kommando afgives ved at holde CTRL-tasten nedtrykket mens der trykkes på C.
De fleste af de applikationsprogrammer som kører under CP/M samt de fleste ydre CP/M utilities kan afbrydes med CTRL-C. Hvis De imidlertid prøver at afbryde et program, mens det sender data til skærmen, kan det, før De taster CTRL-C, være nødvendigt at give kommandoen CTRL-S for at standse visningen.

HJÆLP

CP/M 3.0 indeholder en ydre utility kommando, HELP, som viser en summering af formatet og anvendelsen af de mest almindelige CP/M kommandoer. For at få denne hjælp, tastes simpelthen:

```
A>HELP
```

I stedet for at indtaste ordet HELP RETURN kan De blot nedtrykke HELP tasten.
Derefter vises fortegnelsen over tilgængelige emner, som her:

Topics available:

COMMANDS	CNTRLCHARS	COPYSYS	DATE	DEVICE	DIR
DUMP	ED	ERASE	FILESPEC	GENCOM	GET
HELP	HEXCOM	INITDIR	LIB	LINK	MAC
PATCH	PIP(COPY)	PUT	RENAME	RMAC	SAVE
SET	SETDEF	SHOW	SID	SUBMIT	TYPE
USER	XREF				

Hvis De indtaster:

HELP⟩PIP

vil CP/M give følgende oplysninger:

PIP(COPY)
Syntax:
DESTINATION SOURCE
PIP d: Gn filespec [Gn] =filespec [o] ,... d: [o]

Forklaring:

Filkopieringsprogrammet PIP kopierer filer, kombinerer filer og overfører filer mellem disketter, printere eller andre enheder, der er koblet til Deres computer. Første 'filespec' er bestemmelsesstedet.
Anden 'filespec' er kilden. Brug to eller flere kilde 'filespec' adskilte af kommaer, hvis De ønsker at kombinere to eller flere filer i een fil. 'o' er enhver kombination af mulige tilføjelser.
'Gn' tilføjelsen ved destinations'filespec' fortæller PIP, at filen skal kopieres til dette brugernummer.

PIP uden nogen kommandotilføjelse viser en * og afventer Deres serier af kommandoer, som indlæses og behandles med een linie ad gangen.
Kilde- eller destinationssted kan være enhver logisk CP/M 3.0 enhed.

HELP faciliteten giver lignende information om alle CP/M 3.0 indbyggede og ydre utility kommandoer. Hvis De ønsker information om et bestemt emne, kan De indtaste 'HELP emne' efter systemmeldingen, hvor 'emne' er en kommandotilføjelse, som beskriver det emne, De er interesseret i. F.eks.:

A⟩HELP PIP
A⟩HELP DIRSYS

De kan henvende Dem til HELP når De ønsker information om en bestemt kommando. Eller De kan blot blade gennem HELP for at øge Deres kendskab til CP/M 3.0.

AFSNIT 15
COMMODORE UDVIDELSER TIL CP/M 3.0

Commodore har tilføjet en del udvidelser til CP/M 3.0. Disse udvidelser understøttes af CP/M 3.0. Dette afsnit gennemgår disse forbedringer.

TASTATUR UDVIDELSER

Enhver tast på tastaturet kan defineres til at afgive en kode eller funktion, **undtagen** følgende taster:

Venstre SHIFT tast
Højre SHIFT tast
Commodore tasten (C=)
CONTROL tasten
RESTORE tasten
40-80 tasten
CAPS LOCK tasten

Ved definition af en tast accepterer tastaturet de efterfølgende specielle funktioner. For at indikere disse funktioner skal CONTROL tasten og den højre SHIFT tast holdes nedtrykkede, mens den ønskede funktionstast samtidig nedtrykkes.

TAST	FUNKTION
MARKØR TIL VENSTRE	**Definerer tast**
MARKØR TIL HØJRE	**Definerer streng (peger på funktionstast)**
ALT	**Skifter tast filter**

DEFINITION AF EN TAST

Brugeren kan definere at en tast skal give en bestemt kode. Hver tast har fire mulige definitioner: Normal, Alfa Shift, Shift og Control. Alfa Shift slås til/fra ved at trykke på Commodore tasten. Efter at være kommet i denne mode, vil en lille kasse vise sig nederst på skærmen. Den første tast som nedtrykkes er den tast, der skal defineres. Den hexadecimale værdi tasten har for øjeblikket bliver vist; brugeren kan derefter indtaste den nye HEX værdi, tasten skal have, eller afbryde ved at trykke på en ikke-HEX tast. Følgende er en definition af de koder, som kan tildeles en tast. (I ALT mode returneres koderne til applikationen; se ALT Mode).

KODE	FUNKTION
00h	**Intet (det samme som ikke at trykke på en tast)**
01h til 7Fh	**Normale ASCII koder**
80h til 9Fh	**Tildeling af streng**
A0h til AFh	**80 kolonners karakter farve**
B0h til BFh	**80 kolonners baggrunds farve**
C0h til CFh	**40 kolonners karakter kode**
D0h til DFh	**40 kolonners baggrunds farve**
E0h til EFh	**40 kolonners kant farve**
F0h	**Slå diskstatus til/fra**
F1h	**System pause**
F2h	**(udefineret)**
F3h	**40 kolonners skærmvindue, venstre**
F4h	**40 kolonners skærmvindue, højre**
F5 til FFh	**(udefineret)**

DEFINITION AF EN STRENG

Denne funktion tillader brugeren at tildele en tast mere end en enkelt kode. Enhver tast som nedtrykkes i denne mode bliver indsat i strengen.
Brugeren kan se resultatet af indtastningen i en lang kasse nederst på skærmen.

OBS! Visse taster viser måske ikke, hvad de er. For at give brugeren styring over processen med indtastning af data, er de følgende fem funktionstaster til rådighed. For at bruge disse funktioner nedtrykkes CONTROL og højre SHIFT tast samt den ønskede funktionstast.

TAST	FUNKTION
RETURN	**Fuldstændig strengdefinition**
+ (på hovedtastatur)	**Indsæt mellemrum i en streng**
- (på hovedtastatur)	**Slet markørkarakter**
Venstre pil	**Markør til venstre**
Højre pil	**Markør til højre**

ANVENDELSE AF ALT MODE

ALT mode er en omstillingsfunktion (d.v.s. den kan skifte mellem TIL og FRA). Standardværdien er FRA. Denne funktion sætter brugeren i stand til at sende 8-bit koder til en applikation.

UDVIDELSER AF SKÆRMEN

Standardskærmen i CP/M 3.0 emulerer en ADM31 terminal. Følgende skærmfunktioner emulerer ADM 3A operation, som er en underfunktion af ADM31.

CTRL G	Ringer med klokken
CTRL H	Markør mod venstre
CTRL J	Markør ned
CTRL K	Markør op
CTRL L	Markør mod venstre
CTRL M	Markør til aktuel linies start (CR)
CTRL Z	Markør til HOME position og sletning af skærm
ESC=RC	Markørpositionering, hvor R betyder Række (med værdi fra mellemrum til 8) og C er Kolonne (næste værdier fra mellemrum til 0), refererende til statuslinien.

Tilføjede funktioner i ADM31 mode includerer:

ESC T	Slet til linieslut
ESC t	-do-
ESC Y	Slet til skærmslut
ESC y	-do-
ESC:	Markør til HOME position og sletning af skærm, incl. statuslinien.
ESC*	-do-
ESC Q	Indsæt karakter
ESC W	Slet karakter

ESC E	Indsæt linie
ESC R	Slet linie

ESC ESC ESC farvenummer, sætter en skærmfarve fra en tabel med 16 farveindgange. (Samme farver som vist i kapitel 2, afsnit 6, figur 2).

Farvenumrene vil blive sat som følger:
20h til 2Fh karakterfarve
30h til 3Fh baggrundsfarve
40h til 4Fh kantfarve (kun 40 kolonner)

Den visuelle effekt af følgende funktioner er kun synlig med 80 kolonners skærmformat.

ESC)	halv intensitet
ESC (	Fuld intensitet
ESC G4	Reverse video TIL
ESC G3	Understregning TIL (ikke normal ADM31 sekvens)
ESC G2	Blink TIL
ESC G1	Vælg alternativt karaktersæt (ikke normal ADM31 sekv.)
ESC G0	Alle ESC G attributter FRA

****

Afsnittene i dette kapitel indeholder en summering af strukturen og de omfattende muligheder med CP/M 3.0.

KAPITEL

5

BASIC 7.0 LEKSIKON

AFSNIT 16
INTRODUKTION

LEKSIKONNETS OPBYGNING

Dette kapitel indeholder BASIC 7.0 sprogelementer. Det giver en komplet opstilling af reglerne (syntaksen) for Commodore 128 BASIC 7.0, og giver samtidig en nøjagtig beskrivelse af de enkelte.

BASIC 7.0 indeholder alle elementer fra BASIC 2.0. De kommandoer, instruktioner, funktioner og operatorer, som er markeret med en foranstillet *, kan bruges i begge versioner.

De forskellige operationstyper i BASIC opstilles i sektioner baseret på følgende kriterier:

1. KOMMANDOER: Kommandoer som anvendes ved opbygning, lagring og sletning af programmer; og de BASIC prograaminstruktioner, der bruges i et programs nummererede linier.
2. FUNKTIONER: Streng-, numeriske og print- funktioner.
3. VARIABLER OG OPERATORER: De forskellige typer variabler, lovlige variabelnavne og aritmetiske og logiske operatorer.
4. RESERVEREDE ORD OG SYMBOLER: de ord og symboler som er reserveret til brug for BASIC 7.0 sproget, og som ikke kan bruges til andre formål.

FORMAT AF KOMMANDOER OG INSTRUKTIONER

Kommando og instruktions definitionerne er i dette leksikon arrangeret i følgende format:

Kommando navn -	AUTO
Kort forklaring	- Starter/slutter automatisk linienummerering
Kommandoformat	AUTO [linienr.]
Gennemgang af format og brug	Denne kommando starter automatisk linienummering. Dette letter arbejdet med indtastning af programmer, da linienumre automatisk skrives for brugeren. Efterhånden som den enkelte programlinie afsluttes med tryk på RETURN, skrives næste linienummer på skærmen og markøren placeres to pladser til højre for linienummeret. Linienummerargumentet refererer til den ønskede forøgelse mellem linienumrene. AUTO uden noget argument afslutter automatisk nummerering, det samme gør RUN. Denne instruktion kan kun bruges i Direkte Mode (ikke i et program).
EKSEMPLER:	AUTO 10 - Nummererer aut. programlinierne med en forøgelse på 10.
	AUTO 50 - Nummererer med en forøgelse på 50
	AUTO - Afbryder automatisk linienummerering

Linien, som definerer formatet, består af følgende elementer:

DLOAD "programnavn" [D0,U8]

nøgleord **argument** **yderligere argumenter (mulige)**

Delene af en kommando eller instruktion skal indtastes nøjagtigt som vist ovenfor (med store bogstaver). Ord, som brugeren indlægger, som f.eks. programnavnet er skrevet med små bogstaver.

Når der er anførelsestegn ("") omkring noget (normalt et programnavn eller et filnavn), skal brugeren medtage dem på det korrekte sted, i henhold til ovenfor viste eksempel.

NØGLEORD, også kaldet RESERVED WORDS, vises i versalier. DISSE NØGLEORD SKAL INDTASTES NØJAGTIGT, SOM DE ER VIST. Imidlertid har mange nøgleord også forkortelser, som kan

bruges. Disse er vist i Appendiks K. Staves de ikke nøjagtigt, som de skal, bliver resultatet en fejlmeddelelse. BASIC og DOS fejlmeddelelser defineres i Appendiks A og B.
Nøgleord er ord, som er en del af det Basic sprog, som computeren kan forstå. Nøgleord er den centrale del af en kommando eller en instruktion. De fortæller computeren, hvad den skal udføre. Disse ord kan ikke benyttes som variabel navne. En komplet liste over reserverede ord og symboler kan ses i afsnit 20.

ARGUMENTER (også kaldet parametere) vises med små bogstaver. Argumenter er dele af en kommando eller en instruktion; de forsyner nøgleordet med specifik information vedr. kommandoen eller instruktionen. For eksempel fortæller et nøgleord, at computeren skal indlæse et program, mens argumentet oplyser computeren om, hvilket program, det drejer sig om og et andet argument specificerer på hvilket drevnummer disketten med dette program befinder sig. Argumenter omfatter filnavne, variabler, linienumre, etc.

SKARPE PARANTESER ([]) viser mulige argumenter. Afhængig af ønsker vælger brugeren et, flere eller ingen af de anførte argumenter. Paranteserne er vejledende og skal ikke indtastes.

VINKEL PARANTESER ⟨ ⟩ indikerer, at brugeren SKAL vælge eet af de anførte argumenter. Paranteserne er vejledende og skal ikke indtastes.

LODRET STREG (|) adskiller elementer i en række argumenter, når valget er begrænset til de anførte argumenter, og ikke andre argumenter kan bruges.
Såfrem der optræder en lodret streg, indrammet af SKARPE PARANTESER, i en listning, er valget begrænset til de elementer, der vises i listen, men det kan stadig vælges ikke at bruge nogen af disse.

ELLIPSER ... en sekvens bestående af tre prikker betyder, at en valgmulighed eller et argument kan repeteres mere end een gang.

ANFØRELSESTEGN " " indrammer karakterstrenge, filnavne og andre udtryk.
Når argumenter er indrammet af anførelsestegn i et format, skal

disse anførelsestegn medtages i en kommandofil eller instruktion. Anførelsestegn er ikke betingelser, som beskriver formater; de er krævede dele af en kommando eller instruktion.

PARANTESER ( ) Hvis argumenter er indrammet af paranteser i et format, skal de medtages i en kommando eller instruktion. Paranteser er ikke betingelser, som beskriver formater; de er krævede dele af en kommando eller instruktion.

VARIABLER – en variabel er ethvert gyldigt BASIC variabelnavn som X, A$ eller T%.

UDTRYK – ved udtryk forstås ethvert gyldigt BASIC udtryk som A+B+2 eller .5*(X+3).

KOMMAER, KOLONER og SEMIKOLONER -- skal medtages i kommandoer eller instruktioner, da de er krævede dele af sådanne.

GRAPHIC OG SOUND KOMMANDO FORMAT –

Mulige parametere i GRAPHIC kommandoer er således repræsenterede:

[,parameter]

Såfremt parametere udelades, skal kommaet medtages, fordi parameterne er positionsafhængige. Man må imidlertid ikke medtage kommaer efter den sidst angivne parameter.

EKSEMPEL:

ENVELOPE n [,atk][,dec][,sus][,rel][,wf][,pw]

Hvis blot rel parameteren ønskes ændret, bruges:

ENVELOPE n,,,,rel

De tre første kommaer angiver positionerne for atk, dec, sus og det fjerde er kommaet for rel. Kommaer for wf og pw må ikke indtastes.

Hvis der i en grafikkommando er angivet en koordinat (med x,y), er det muligt at erstatte denne med en vector (x;y). I følgende tilfælde er:

x = distancen (skalasat)
y = vinklen i grader (0 = op, 90 = ret vinkel etc.)

Det vil sige, at:

```
LOCATE 160,100
DRAW TO 40;45
```

vil tegne en linie i en 45 graders vinkel med længden 40.

DISKETTEKOMMANDO FORMAT –

Mulige parametere i diskettekommandoer vises således:

[,parameter]

Hvis parameteren er den første efter selve kommandoen, er kommaet unødvendigt. Hvis andre parametere, som kræver kommaer, udelades, skal kommaerne også udelades.

EKSEMPEL:

DIRECTORY [Ddrev][⟨ON,⟩Uenhedsnummer][,søgekarakter]

vil frembringe:

DIRECTORY D0 ON U8,"AB*"

Hvis kun søgekarakteren ønskes angivet, er komma ikke nødvendigt, d.v.s.:

DIRECTORY "AB*"

Hvis variabler bruges i diskettekommandoer skal de omgives af paranteser.

For eksempel:

DIRECTORY D(DV),(A$)

AFSNIT 17
BASIC KOMMANDOER OG INSTRUKTIONER

APPEND - fil udvidelse

APPEND # logisk filnummer,"filnavn",[Ddrevnummer] [⟨ON|, ⟩Uenhed]

Åbner filen med det angivne filnavn for skrivning og positionerer filpointeren ved filens slutning. Senere PRINT# logisk filnummer instruktioner vil forårsage, at data blive tilføjet fra filens slutposition. Standardværdier for hhv. drev og enhed er 0 og 8.

Bruges variabler eller udtryk som filnavnne, skal de være omgivet af paranteser.

EKSEMPLER:

Append # 8, "MINFIL"

Åbner logisk fil nummer 8 , "MINFIL"
for tilføjelse af efterfølgende PRINT# instruktioner.

Append # 7,(A$),D0,U9

Åbner logisk fil navngivet af variablen A$ i drev 0 på enhed 9, og klargør til tilføjelse.

AUTO

- åbner/lukker for automatisk linienummerering

AUTO [linie#]

Iværksætter automatisk linienummerering. Dette letter arbejdet med indtastning af programmer, idet linienumrene sættes for brugeren. Når hver programlinie afsluttes med tryk på RETURN, skrives næste linienummer på skærmen, og markøren placeres to pladser til højre for nummeret. Linienummerargumentet er af betydning for spring mellem linienumrene. Bruges AUTO uden ARGUMENT afbryder den automatiske nummering, på samme måde som RUN. Denne instruktion kan kun bruges i direkte mode.

EKSEMPLER:

AUTO 10 automatisk linienummerering med en forøgelse på 10
AUTO 50 automatisk linienummerering med en forøgelse på 50

AUTO afbryder automatisk linienummerering

BACKUP

- drev-til-drev diskette backup

BACKUP source Ddrevnummer TO destination Ddrevnummer [⟨ON|,⟩Uenhed nummer]

Denne kommando kopierer alle filer fra een diskette til en anden på en dobbeltdrev diskettestation. Kopiering til en ny diskette kan foretages uden først at anvende HEADER kommandoen til at formattere den nye diskette. Årsagen hertil er, at BACKUP kommandoen kopierer al den information, der findes på disketten, inclusive formatet. Derfor ødelægger BACKUP kommandoen enhver information som i forvejen måtte findes på destinationsdisketten. Når De foretager BACKUP til en tidligere anvendt diskette, skal De derfor sikre Dem, at den ikke indeholder programmer, De ikke vil miste. Som en sikkerhedsforanstaltning spørger computeren "ARE YOU SURE? (Er De sikker?) før operationen påbegyndes. Tryk på "Y" for at udføre backup'en, eller en anden tast for at afbryde. Brug altid BACKUP, hvis den originale diskette er mistet eller ødelagt. Se også COPY kommandoen. Standardværdi er enhed 8.

OBS! Denne kommando kan kun bruges på en dobbeltdrev diskettestation.

EKSEMPLER:

BACKUP D0 TO D1

Kopierer alle filer fra disketten i drev 0 til disketten i drev 1 på diskenhed nummer 8.

BACKUP D0 TO D1 ON U9

Kopierer alle filer fra drev 0 til drev 1 på diskenhed nummer 9.

BANK

Vælger een af de seksten RAM-banker, nummereret 0 - 15.

BANK banknummer

Denne instruktion specificerer banknummeret og den modsvarende hukommelses konfiguration for Commodore 128 hukommelsen. Standardværdien er bank nummer 15. Her er en tabel indeholdende mulige BANK konfigurationer i Commodore 128's hukommelse:

BANK	Konfiguration
0	Kun RAM(0)
1	Kun RAM(1)
2	Kun RAM(2) RAM 2 og 3 er kun til brug på en computer,
3	Kun RAM(3) hvis hukommelse er udvidet til 256K.
4	Intern ROM, RAM(0), I/O
5	Intern ROM, RAM(1), I/O
6	Intern ROM, RAM(2), I/O som 2, 3
7	Intern ROM, RAM(3), I/O som 2, 3
8	Ekstern ROM, RAM(0), I/O
9	Ekstern ROM, RAM(1), I/O
10	Ekstern ROM, RAM(2), I/O som 2, 3
11	Ekstern ROM, RAM(3), I/O som 2, 3
12	Kernal og intern ROM(LOW), RAM(0), I/O
13	Kernal og ekstern ROM(LOW), RAM(0), I/O
14	Kernal og BASIC ROM, RAM(0), Karakter ROM
15	Kernal og BASIC ROM, RAM(0), I/O

For at få adgang til en speciel bank , tastes BANK n (n=0-15). Fra monitoren foranstilles det firecifrede hexadecimale tal på det ønskede adresseområde med et hexadecimalt ciffer (0-F).

Denne procedure er detaljeret beskrevet i Commodore 128 Programmer's Reference Guide.

BEGIN/BEND

En betinget instruktion som IF...THEN:ELSE, struktureret så De kan medtage adskillige programlinier mellem start (BEGIN) og slut (BEND) Her er formatet:

IF betingelse THEN BEGIN: instruktion
instruktion
instruktion BEND:ELSE BEGIN
instruktion
instruktion BEND

For eksempel:

```
10 IF X=1 THEN BEGIN: PRINT "X=1 er sand"
20 PRINT "Nu er denne del af instruktionen udført"
30 PRINT "Når X er lig med 1"
40 BEND: PRINT "Slut på BEGIN/BEND strukturen":GOTO
   60
50 PRINT "X er ikke lig med 1":PRINT "Der ses bort fra
   instruktionerne mellem BEGIN/BEND"
60 PRINT "Resten af programmet"
```

Hvis vilkårsinstruktionen (IF...THEN) i linie 10 er opfyldt, udføres instruktionerne mellem nøgleordene BEGIN og BEND, indbefattet alle instruktioner, som står på samme linie som BEND. Er vilkåret ikke opfyldt (falsk), ses der bort fra alle instruktioner mellem BEGIN og BEND, incl. instruktionerne på samme programlinie som BEND, og programmet fortsætter med den programlinie, der følger umiddelbart efter den linie, som indeholder BEND. BEGIN/BEND behandler stort set linierne fra 10 t.o.m. 40 som een lang linie.

Samme regler gælder, hvis ELSE:BEGIN klausulen er medtaget. Er vilkåret opfyldt, udføres alle instruktioner mellem ELSE:BEGIN og BEND, incl. alle instruktioner, som står på samme linie som BEND. Er vilkåret ikke opfyldt, fortsætter programudførelsen fra den linie, der følger umiddelbart efter linien med BEND.

BLOAD

Indlæser en binær fil med start i den anførte hukommelseslokation

BLOAD "filnavn"[,Ddrevnummer][⟨ON|,⟩ Uenhedsnummer] [,Bbanknummer][,Pstartadresse]

hvor

- **filnavn er navnet på Deres fil**
- **banknummer lader Dem vælge en af 16 banker**
- **startadresse er den hukommelseslokation, hvorfra indlæsning begyndes.**

En binær fil er en fil, enten et program eller data, som er blevet lagret enten fra maskinsprogs monitoren med SAVE eller ved hjælp af BSAVE kommandoen. BLOAD kommandoen indlæser den binære fil til den lokation, der er angivet med startadressen.

EKSEMPLER:

BLOAD "SPRITES",B0,P3584
Indlæser den binære fil "SPRITES" startende i lokation 3584 (i bank 0)

BLOAD "DATA1",D0,U8,B1,P4096
Indlæser den binære fil "DATA1" til lokation 4096 (bank 1) fra drev 0, enhed 8.

BOOT

Indlæser og udfører et program, som er lagret som en binær fil.

BOOT filnavn [,Ddrev nummer][⟨ON|,⟩Uenhedsnummer]

Denne kommando indlæser en udførbar binær fil og begynder udførelsen ved den prædefinerede startadresse. Standardværdien for enhedsnummer er 8 (drev 0).

EKSEMPEL:

BOOT

BOOTer et udførbart program (f.eks. CP/M Plus)
Dette er et specielt tilfælde, som kræver oprettelse af en specifik sektor på disketten.

BOOT "GRAFIK1",D0,U9

BOOTer programmet "GRAFIK1" fra drev 0 på enhed 9, og udfører det. Udførelsen begynder i startadressen i den linie, hvor indlæsning påbegyndes.

BOX

Tegner en kasse på en specificeret plads på skærmen

BOX [farveområde],X1,Y1,[,X2,Y2][,vinkel][,farv]

hvor:

farveområde . 0 = baggrundsfarve
1 = forgrundsfarve
2 = flerfarve 1
3 = flerfarve 2

X1, Y1 ..... Koordinat på øverste venstre hjørne (skaleret)

X2, Y2 ..... Højre nederste hjørne (skaleret): standardværdien er PC lokationen.

vinkel ..... Rotation i urets retning. Standard er 0 grader

farv ..... Farvelægger figur
0 = farvelæg ikke
1 = farvelæg
(standardværdi = 0)

Denne instruktion tillader brugeren at tegne et rektangel af enhver størrelse på skærmen. Rotation baseres på rektanglets centrum. Pixel markøren (PC) er placeret i X2, Y2 efter udførelse af BOX instruktionen. Er man i standard højopløsnings mode, skal farveområde nummeret være 0 (nul) eller 1, eller 2 eller 3, hvis man er i flerfarve højopløsnings mode. Se også GRAPHIC kommandoen for valg af den passende grafiske mode til brug sammen med BOX farveområde nummeret.

X og Y værdierne kan placere pixelmarkøren på absolutte koordinater som f.eks. (100,100) eller på koordinater, som er relative i forhold til pixelmarkørens tidligere position (+/-x og +/-y) som (+20,-10). En akses (x eller y) koordinat kan være relativ og den anden absolut. Her ses de mulige kombinationer på måder, hvorpå x og y koordinaterne kan specificeres:

x,y	absolut x, absolut y
+/-x,y	relativt x, absolut y
x,+/-y	absolut x, relativt y
+/-x,+/-y	relativt x, relativt y

Se endvidere LOCATE kommandoen for information om pixel-markøren.

EKSEMPLER:

BOX 1,10,10,60,60	Tegner omridset af en kasse
BOX,10,10,60,60,45,1	Tegner en farvelagt, drejet kasse (en diamant)
BOX,30,90,,45,1	Tegner en udfyldt, drejet polygon

Enhver parameter kan udelades, men der skal indsættes et komma på dens plads, som i de to sidste eksempler.

NB: X2, Y2 gælder som een parameter, hvorfor det kun er nødvendigt med eet ekstra komma.

OBS! Hvis gradangivelsen er større end 360, vil der ske 'wrapping'.
d.v.s. at 360 = 0 (450 = 90).

BSAVE

Lagrer en binær fil fra enhver specificeret hukommelsesplads
BSAVE "filnavn"[,Ddrevnummer][⟨ON|,⟩ Uenheds-nummer][,Bbanknummer] ,Pstartadresse TO Pslutadresse

hvor:

- filnavn er det navn, De giver filen
- drevnummer er enten 0 eller 1 på et dobbeldrev (0 er standardværdien for enkeltdrev)
- enhedsnummer er diskettestationens nummer (standardværdi er 8)
- banknummer er nummeret på den bank, De specificerer (0-15)
- startadresse er den adresse, hvorfra programmet lagres
- slutadresse er den sidste adresse i hukommelsen, hvorfra der lagres. Skal være een byte større end den sidste byte, der ønskes lagret.

Dette er det samme som SAVE kommandoen udfører i maskinsprogs monitoren.

EKSEMPLER:

BSAVE "SPRITE DATA",B0,P3584 TO P4096
Lagrer den binære fil "SPRITE DATA" med begyndelse i adresse 3584 og slutning i 4095. (bank 0)

BSAVE "PROGRAM.SCR",D0,U9,B0,P3182 TO P8000
Lagrer den binære fil "PROGRAM.SCR" i adresseområdet 3182 til 7999 (bank 0) på drev 0, enhed 9.

CATALOG

Viser en diskettes indholdsfortegnelse på skærmen.

CATALOG [Ddrevnummer][⟨ON|,⟩Uenhedsnummer][,søgestreng]

CATALOG kommandoen viser directory på det specificerede drev på samme måde som DIRECTORY kommandoen. Se denne kommando.

EKSEMPEL:
CATALOG viser directory for drev 0

CHAR

Viser karakterer i den specificerede skærmposition.

CHAR [farveområde],x,y[,streng][,RVS]

Formålet med dette er primært at vise karakterer på en højopløsnings skærm, men kan også bruges på en tekstskærm. Her er parameternes betydning:

Farveområde	0 = baggrund 1 = forgrund
x	Karakterkolonne (0-79) (skifter til næste linie i 40 kolonners mode)
y	karakterrække (0-24)
streng	streng til udskrivning
RVS	Reverse felt flag (0 = fra, 1 = til)

Tekst (alfanumeriske strenge) kan blive vist på enhver skærm i en given position ved brug af CHAR instruktionen. Karakterdata læses fra Commodore 128's karakter ROM område. Brugeren tilføjer x og y koordinaterne på startpositionen og den tekst-

streng, der skal vises. Farveområde og reverse (negativ) kan medtages.

Strengen fortsætter strengen på næste linie, hvis der forsøges på skrive de sidste tegn udenfor skærmens højre kant. I TEXT mode virker den streng, som skrives af CHAR kommandoen, på nøjagtig samme måde som en PRINT streng, incl. farve og markørkontrol. Disse kontrolfunktioner i strengen virker ikke, når CHAR kommandoen bruges til at vise tekst i højopløsnings mode. Upper og lower case kontroller CHR$(14) eller CHR$(142) virker også i højopløsnings mode.

Flerfarve-karakterer behandles anderledes end standard-karakterer, og nedenstående tabel viser, hvorledes de mulige kombinationer oprettes:

Farvesource	Reverse 0(fra)	Flag 1(til)
0	ekst 1 Baggr. 2	1 3
1	1 0	0 1
2	2 0	0 2
3	3 0	0 3

```
10 COLOR 2,3: REM flerfarve 1 = rød
20 COLOR 3,7: REM flerfarve 2 = blå
30 GRAPHIC 3,1
35 CHAR 0,10,10,"TEKST",0
40 CHAR 0,10,11,"TEKST",1
```

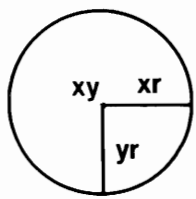
CIRCLE

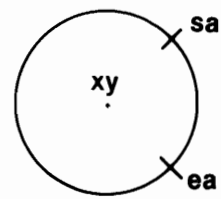
Tegner cirkler, ellipser, buer etc. i specificerede skærmpositioner

CIRCLE [farveområde],X,Y[,Xr][,Yr][,sa][,ea][,vinkel][,inc]

hvor:

farveområde	0 = baggrundsfarve 1 = forgrundsfarve 2 = flerfarve 1 3 = flerfarve 2
X, Y	Cirklens centrums koordinat
Xr	X radius (skaleret)
Yr	Y radius (standardværdi er Xr)
sa	Buevinklens start (standard 0 grader)
ea	Buevinklens slut (standard 360 grader)
vinkel	Rotation i urets retning (standard 0 grader)
inc	Grader mellem segmenter (standard 2 grader)





Med CIRCLE instruktionen kan brugeren tegne cirkler, ellipser, buer, trekanter, oktagoner eller andre polygoner. Pixelmarkøren (PC) efterlades i cirkelbuen ved buens slutvinkel. Enhver rotation sker relativt i forhold til centrum. Sættes y radius lig med x radius opnås ikke en perfekt cirkel, da x og y koordinaterne er forskelligt skalerede (x = 0 -320 og y = 0 -200). Cirkelbuer tegnes i urets retning fra startvinklen til slutvinklen. Forøgelsen styrer figurens skarphed; brug af små værdier resulterer i mere næsten cirkulære figurer. Angives "inc" højere end 2, fås skarpkantede figurer.

X og Y værdierne kan placere pixelmarkøren på absolutte koordinater som f.eks. (100,100) eller på koordinater, som er relative i forhold til den tidligere position (+/-x og +/-y) af pixelmarkøren som f.eks. (+20, -10). Den ene akses koordinat (x eller y) kan være relativ og den anden absolut. Her er de mulige kobinationer på måder, hvorpå x og y koordinaterne kan angives:

x,y	absolut x, absolut y
+/-x,y	relativt x, absolut y
x,+/-y	absolut x, relativt y
+/-x,+/-y	relativt x, relativt y

Se endvidere LOCATE kommandoen for information om pixel-markøren.

EKSEMPLER:

CIRCLE1, 160,100,65,10	**Tegner en ellipse**
CIRCLE1, 160,100,65	**Tegner en cirkel**
CIRCLE1, 60,40,20,18,,,,45	**Tegner en oktagon**
CIRCLE1, 260,40,20,30,,,,90	**Tegner en diamant**
CIRCLE1, 60,140,20,18,,,,120	**Tegner en trekant**

De kan udelade en parameter, men skal huske at sætte et komma på det rigtige sted. Udelukkede parametere antager standardværdierne.

*CLOSE

Lukker logisk fil

CLOSE filnummer

Denne instruktion lukker alle filer brugt med DOPEN eller OPEN instruktionerne. Talangivelsen efter ordet CLOSE er nummeret på den fil, der ønskes lukket.

EKSEMPEL:
CLOSE 2 Logisk fil nummer 2 lukkes

*CLR

Sletter programvariabler

CLR

Denne instruktion sletter enhver variabel i hukommelsen, men efterlader programmet intakt. Instruktionen udføres automatisk i forbindelse med en RUN eller en NEW kommando.

*CMD

Omdirigerer skærm uddata.

CMD logisk filnummer [,uddataliste]

Denne kommando sender de uddata, som normalt sendes til skærmen, som f.eks. PRINT instruktioner, LIST, men ikke POKES til skærmen, til en anden enhed, som en diskette datafil eller en printer. Enheden eller filen skal først åbnes med OPEN.

CMD kommandoen skal efterfølges af et tal eller en numerisk variabel, som refererer til filen. Uddatalisten kan være enhver alfanumerisk streng eller variabel. Kommandoen er nyttig ved udskrivning af overskrifter på programlister.

EKSEMPEL:

OPEN 1,4 Åbner enhed nummer 4, printeren
CMD 1 Al normal uddata går nu til printeren
LIST Listningen går til printeren, ikke til skærmen, selv ordet READY.

PRINT # 1 Stiller uddata tilbage til skærmen

CLOSE 1 Lukker filen

COLLECT

Frigiver utilgængelig plads på disketten

COLLECT [Ddrev nummer] [,ON Uenheds nummer]

Brug denne kommando til at frigøre plads, som er allokeret til ukorrekt lukkede filer (splat files) og til at slette referencer til disse filer fra indholdsfortegnelsen. Splat filer er filer, som viser sig i directory markeret med en *. Standardværdi til enhed 8.

EKSEMPEL

COLLECT D0

Frigiver al den plads, som er blevet allokeret til ukorrekt lukkede filer.

COLLISION

Fastlægger behandling ved sprite kollisions afbrydelse

COLLISION type [,instruktion]

type ..... Afbrydelsestype, som følger:
1 = Sprite-til-Sprite kollision
2 = Sprite-til-skærmdata kollision
3 = Lyspen (kun 40 kolonner)
instruktion Linienummer på en BASIC subrutine
Hvis den specificerede situation opstår, vil BASIC afbryde udfø-

relsen af den aktuelle instruktion og udføre en GOSUB til det anførte linienummer. Når subrutinen afsluttes (den skal slutte med RETURN), vil BASIC genoptage behandlingen, hvor den blev afbrudt. Afbrydelses aktion vil fortsætte, til en COLLISION af samme art, uden linienummer, specificeres. Mere end een type afbrydelse kan fungere samtidigt, men kun een kan behandles ad gangen (d.v.s. at der ikke kan være nogen rekursion eller 'nesting' af afbrydelser). Grunden til en afbrydelse kan muligvis fortsætte med at lave afbrydelser i nogen tid, med mindre situationen ændres eller afbrydelsen sættes ud af kraft.

Hvis en sprite er ude af skærmen og dermed usynlig, kan den ikke forårsage en afbrydelse. For at kunne afgøre, hvilke sprites der er kollideret siden sidste kontrol, bruges BUMP funktionen.

EKSEMPEL:

COLLISION 1, 5000
Opdager en sprite-til-sprite kollision og programkontrol sendes til en subrutine i linie 5000.

COLLISION 1
Standser afbrydelsesaktionen, som var iværksat i ovenstående eksempel.

COLLISION 2, 1000
Opdager sprite-til-data kollision og programkontrol dirigeres til subrutine i linie 1000.

NB: Sprite kan stadig kollidere, selvom de er udenfor skærmbilledet, dog ikke hvis de er slukkede.

COLOR

Definerer farver for hvert skærmområde

COLOR områdenummer, farvenummer

Denne instruktion tildeler farve til et af de syv farveområder:

NUMMER	OMRÅDE
0	40 kolonners (VIC) baggrund
1	40 kolonners (VIC) forgrund
2	flerfarve 1
3	flerfarve 2
4	40 kolonners (VIC) kant

5	karakterfarve (40 eller 80 kolonners skærm)
6	80 kolonners baggrundsfarve

Farver kan bruges i intervallet 1 -16.

KODE	FARVE	KODE	FARVE
1	Sort	9	Orange
2	Hvid	10	Brun
3	Rød	11	Lyserød
4	Cyan	12	Mørkegrå
5	Purpur	13	Mellemgrå
6	Grøn	14	Lysegrøn
7	Blå	15	Lyseblå
8	Gul	16	Lysegrå

FARVEKODER I 40 KOLONNERS FORMAT

1	Sort	9	Mørk purpur
2	Hvid	10	Mørkegul
3	Mørkerød	11	Lyserød
4	Lys cyan	12	Mørk cyan
5	Lys purpur	13	Mellemgrå
6	Mørkegrøn	14	Lysegrøn
7	Mørkeblå	15	Lyseblå
8	Lysegul	16	Lysegrå

FARVEKODER I 80 KOLONNERS FORMAT

EKSEMPEL:

COLOR 0,1:
Ændrer 40 kolonners baggrundsfarve til sort.

COLOR 5,8:
Ændrer karakterfarven til gul

CONCAT

Sammenkæder to datafiler.

CONCAT "fil2"[,Ddrevnummer]TO"fil1"[,Ddrevnummer][⟨O-N|,⟩Uenhed]

CONCAT kommandoen føjer fil2 til fil1's slutning og får fil 1's navn. Standardværdierne er for enheden 8, og for drevet 0.

EKSEMPEL:

CONCAT "filB" TO "filA"
FilB føjes til FilA og de kombinerede filer får navnet filA.

CONCAT (A$) TO (B$),D1,U9
Filnavnet fra B$ bliver en ny fil med samme navn som den fil, der er navngivet af A$ tilføjet enden af B$ - dette udføres på enhed 9, drev 1 (en dobbelt diskettestation)

Når som helst en variabel bruges som filnavn, som i sidste eksempel, skal filnavn-variablen omgives af paranteser.

*CONT

Fortsætter programudførelse

CONT

Denne kommando anvendes for at genstarte udførelse af et program, som er blevet standset enten ved nedtrykning af STOP tasten, eller på grund af en STOP instruktion eller en END instruktion i programmet. Programmet vil fortsætte, hvorfra det standsede. CONT vil ikke virke, hvis linier er ændret eller tilføjet til programmet (eller hvis blot markøren er flyttet til en programlinie og RETURN er nedtrykket uden at ændring er sket), hvis programmet er standset på grund af en fejl, eller hvis brugeren har lavet en eller anden fejl før genstart er forsøgt. Fejlmeddelsen vil i dette tilfælde være CAN'T CONTINUE ERROR.

COPY

Kopierer filer fra et drev til det andet på en dobbelt diskette station, eller på et enkeltdrev.

COPY "kilde filnavn"[,Ddrevnummer]TO"destinationsfil" [,Ddrevnummer][⟨ON|,⟩Uenhed]

Kopierer en fil fra disketten i eet drev (kilde filen) til en 'destinationsfil' på disketten i det andet drev, derfor kun på en dobbelt diskettestation; kommandoen kan også skabe en kopi af en fil på samme drev (med et nyt filnavn). Ved kopiering fra et drev til et andet kan filnavnene være ens.

COPY kommandoen kan også kopiere alle filer fra et drev til det andet på en dobbelt diskettestation. I dette tilfælde specificeres drevnumrene, mens kilde og destination filnavne udelades. Standardværdier for COPY er enhed nr. 8, drev 0.
OBS! Kopiering mellem to enkelte eller dobbelte enheder kan ikke lade sig udføre. Se BACKUP.

EKSEMPLER:

COPY "test",D0 TO "test prog",D1

Kopierer "test" fra drev 0 til drev 1 og omdøber det til "test prog" på drev 1.

COPY "DIVERSE",D0 TO "DIVERSE",D1 Kopierer "DIVERSE" fra drev 0 til drev 1.

COPY D0 TO D1

Kopierer alle filer fra drev 0 til drev 1.

COPY "ARB.PROG" TO "BACKUP"

Kopierer "ARB.PROG" som et program ved navn "BACKUP" på den samme diskette.

*DATA

Definerer data til brug i et program.

DATA liste over konstanter

Denne instruktion efterfølges af en liste med dataindivider, som skal være inddata til computerens hukommelse via READ instruktioner.
Individerne kan være numeriske eller strenge og skal adskilles med kommaer. Strengdata behøver ikke at være i anførelsestegn, medmindre de indeholder nogle af følgende tegn: mellemrum, kolon eller komma.
Er der intet mellem to kommaer, opfatter READ værdien som et 0 eller en tom streng. Se endvidere RESTORE instruktionen, som sætter Commodore 128 i stand til at genlæse data.

EKSEMPEL:

```
DATA 100,200,FRED,"GODDAG, MOR",,3,14,ABC123
DCLEAR
```

DCLEAR

Klargør alle åbne kanaler på diskettedrevet

DCLEAR [Ddrevnummer][〈ON|,〉Uenhed]

Denne instruktion lukker og klargør alle åbne kanaler på det anførte drevnummer. Standard er enhed 8. Kommandoen svarer til:

OPEN 10,8,15,"I0" CLOSE10.

EKSEMPLER:

DCLEAR D0
Klargør alle åbne kanaler på drev 0, enhed nummer 8.

DCLEAR D1,U9
Klargør alle åbne kanaler på drev 2, enhed nummer 9.

DCLEAR D0 ON U8
Klargør alle åbne kanaler på drev 0, enhed nummer 8.

NB: Alle filer vil blive afbrudt. Data kan muligvis ikke genskabes fra filer, som var under skrivning. SE CLOSE/DCLOSE.

DCLOSE

Lukker diskettefil

DCLOSE [#logisk filnummer][⟨ON|,⟩Uenhed]

Instruktionen lukker en enkelt fil eller alle aktuelt åbne filer på en disketteenhed. Er intet logisk filnummer angivet, lukkes alle de filer, som er åbne. Standard enhedsnummer er 8. Læg mærke til følgende eksempler:

EKSEMPLER:

DCLOSE
Lukker alle aktuelt åbne filer på enhed 8.

DCLOSE #2
Lukker den fil, som svarer til det logiske filnummer 2 på enhed 8.

DCLOSE ON U9
Lukker alle åbne filer på enhed 9.

*DEF FN

Returnerer værdien af en brugerdefineret funktion

DEF FN navn (variabel) = udtryk

Denne instruktion tillader definition af en kompleks kalkulation som funktion. I tilfælde af, at en lang formel skal bruges flere gange i et program, kan dette nøgleord spare værdifuld programplads. Det navn, som gives til funktionen, begynder med bogstaverne FN, efterfulgt af et gyldigt numerisk variabelnavn (heltal eller tabel).
Definer først funktionen med instruktionen DEF, efterfulgt med det navn, den skal have. Efter navnet følger et sæt paranteser () med et numerisk variabelnavn, i dette tilfælde (X). Denne variabel har kun en værdi, hvis den optræder til højre for lighedstegnet i et udtryk.
En funktion og en variabel kan have samme navn, uden at forstyrre hinanden. Andre variabler og/eller funktioner kan indgå i et udtryk og bliver benyttet med den værdi de har i den linie, hvor funktionen kaldes. Derefter et lighedstegn fulgt af den formel, der skal defineres. Funktionen kan udføres med ethvert tal som erstatning for X, når formatet i eksemplets linie 20 bruges:

EKSEMPEL:

```
10 DEF FNA(X)=12*(34.75-X/.3)+X
20 PRINT FNA(7)
30 J = 8
40 PRINT FNA(J)
```

Tallet 7 indsættes automatisk hvert sted, hvor X findes i den formel, der er anført i DEF instruktionen. I ovenanførte eksempel returneres svaret 144 for FNA(7) og 105 for FNA(J).

DELETE

Sletter linier i et BASIC program i det angivne område

DELETE ⟨[start|start-]|[start-slut]|[-slut]⟩

Sletter linier med BASIC tekst. Kan kun bruges i direkte mode.

EKSEMPLER:

DELETE 75
Sletter linie 75

DELETE 10-50
Sletter linier fra 10 til og med 50

DELETE -50
Sletter alle linier i programmet fra start til om med linie 50.

DELETE 75-
Sletter alle programlinier fra linoe 75 til programmets slutlinie.

*DIM

Fastlægger antallet af elementer i et område

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

Før områder med variable kan bruges, skal programmet først udføre en DIM instruktion for at etablere områdets DIMensioner (med mindre der kun er 11 eller færre elementer i hver dimension af området). DIM instruktionen efterfølges af områdets navn, hvilket kan være et hvilket som helst lovligt variabelnavn. Derefter, omgivet af paranteser, indsættes antallet (eller en numerisk variabel) af elementer i hver dimension. Et område med mere end een dimension kaldes en matrix. Ethvert antal dimensioner må benyttes, men husk, at hele listen med variabler optager plads i hukommelsen, og det er let, hvis der bruges for mange, at opbruge hele hukommelsen (out of memory).
Herunder anføres, hvorledes man beregner den hukommelse, der bruges til et område:

5 bytes til hvert områdenavn
2 bytes til hver dimension
2 bytes/elementer til heltals variabler
5 bytes/elementer til normale numeriske variabler
3 bytes/elementer til strengvariabler
1 byte for hver karakter i hvert streng element

Heltalsområder optager to femtedele af 'flydende-punkt' områder, (d.v.s. DIM A%(100) kræver 209 bytes; DIM A(100) kræver 512 bytes.)

I en DIM instruktion kan mere end et område dimensioneres ved at adskille områdevariabel navnene med kommaer. Udfører et program en DIM instruktion for samme variabel mere end een gang, vises meddelelsen "RE'DIM ARRAY ERROR". Det er god programmeringspraksis at placere DIM instruktioner i programmets begyndelse.

EKSEMPEL:

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

DIRECTORY

Viser en diskettes indholdsfortegnelse på skærmen

DIRECTORY [drev nummer][,⟨ON|,⟩Uenhedsnummer][,søgekarakter]

I C128 mode viser F3 funktionstasten DIRECTORY for enhed 8, drev 0.
Visningen kan standses midlertidigt med CONTROL S eller NO SCROLL; tryk på en tilfældig tast genstarter visningen efter pausen. Tryk på Commodore tasten (C=) nedsætter hastigheden. DIRECTORY kommandoen bør ikke bruges til udskrivning på papir. Diskettens DIRECTORY indlæses med (LOAD"$",8) for at få udskrevet en kopi, og derved ødelægges et program, som måtte befinde sig i hukommelsen. Standardværdien på enhed er 8 og på drev 0.

EKSEMPLER:

DIRECTORY

Lister alle filer på disketten(erne).

DIRECTORY D1, U9, "filnavn"

Lister filen benævnt "filnavn" fra disketten i enhed 9. (Standardværdien er 8)

DIRECTORY "AB*"

Lister alle filer, som begynder med bogstaverne "AB" som ABEKAT, ABELONE, etc.

DIRECTORY D0,"fil ?.BAK"

"?" er en søgekarakter, som sammenligner enhvert enkelt tegn i denne position: fil 1.BAK, fil 2.BAK, fil 3.BAK vil alle passe.

DIRECTORY D1,U9,(A$)

Lister det filnavn som findes i variablen A$ på drev 1, enhed 9.
Husk at en variabel skal anføres i paranteser, hvis den bruges som et filnavn.

OBS! For papirudskrivning af DIRECTORY fra drev 0, enhed 8, kan følgende bruges:

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

DLOAD

Indlæser et BASIC program fra diskette

DLOAD "filnavn" [,Ddrev nummer]⟨ON|,⟩[Uenheds nummer]

Denne kommando indlæser et program fra diskette til hukommelsen. (Brug LOAD for indlæsning fra bånd.) Et programnavn på op til 16 karakterers længde skal tilføjes. DLOAD forudsætter enhed 8, drev 0.

EKSEMPLER:

DLOAD "BANKPOSTER"

Søger efter programmet "BANKPOSTER" og indlæser det.

DLOAD (A$)

Indlæser fra diskette et program, hvis navn findes i variablen A$. Hvis A$ er tom, fremkommer en fejlmelding. Husk at hvis en variabel bruges som filnavn, skal den være i paranteser.

DLOAD kommandoen kan bruges i et BASIC program for at finde og køre et andet program på en diskette. Dette kaldes kædning (chaining).

DO/LOOP/WHILE/UNTIL/EXIT

Fastlægger og styrer programløkker.

DO [UNTIL vilkår|WHILE vilkår]instruktioner [EXIT]
LOOP [UNTIL vilkår|WHILE vilkår]

Denne løkkestruktur udfører instruktionerne mellem DO instruktionen og LOOP instruktionen. Hvis ingen UNTIL eller WHILE modificerer DO eller LOOP instruktionerne, fortsætter udførelsen af de mellemliggende instruktioner i det uendelige. Hvis en EXIT instruktion opdages indenfor kroppen af en DO løkke, overgives

udførelsen til den første instruktion, som følger efter LOOP instruktionen. DO løkker kan 'nestes', når reglerne vedr. FOR-NEXT strukturen følges. Er der angivet en UNTIL parameter, fortsætter programmet i løkken, indtil vilkåret er opfyldt (bliver sandt). WHILE parameteren står stort set for det modsatte af UNTIL parameteren; programmet fortsætter med løkken, så længe vilkåret er sandt. Så snart vilkåret ikke længere er opfyldt, fortsætter programkontrollen med den instruktion, der følger umiddelbart efter LOOP instruktionen. Et eksempel på et vilkår (boolsk argument) er A = 1, eller G ⟩ 65.

EKSEMPLER:

```
10 X = 25
20 DO UNTIL X = 0
30 X = X-1
40 PRINT "X =";X
50 LOOP
60 PRINT "SLUT PÅ LØKKEN"
```

Dette eksempel udfører instruktionerne X=X-1 og PRINT "X=";X indtil X = 0. Når X = 0 fortsætter programmet med PRINT "SLUT PÅ LØKKEN" instruktionen lige efter LOOP.

```
10 DO WHILE A$ = "": GET A$: LOOP
20 PRINT "TASTEN ";A$;" ER BLEVET NEDTRYKKET"
```

A$ forbliver nul (tom) så længe ingen tast er blevet nedtrykket. Så snart der trykkes på en tast, overgår programkontrollen til den instruktion, der følger efter LOOP, PRINT instruktionen. Eksemplet udfører GET A$ så længe A$ er en tom karakter. Løkken kontrollerer konstant, om en tast på tastaturet er blevet nedtrykket. NB: GETKEY A$ har samme effekt som linie 10.

```
10 DOPEN #8, "sekvfil"
20 DO
30 GET #8,A$
40 PRINT A$
50 LOOP UNTIL ST
60 DCLOSE #8
```

Dette program åbner filen "sekvfil" og modtager data, indtil systemvariablen ST indikerer, at alle data er læst.

DOPEN

Åbner en diskettefil for læse og/eller skriveoperationer

DOPEN #logisk filnummer,"filnavn[,⟨S|P⟩"],Lpostlængde]

[,Ddrevnummer][⟨ON|,⟩Uenhedsnummer][,W]

hvor:

S = Sekventiel filtype **P** = Program filtype **L** = Længden på posten = længden af poster i en relativ fil **W** = Skriveoperation (anføres det ikke, åbnes for læsning).

Denne instruktion åbner en sekventiel, relativ eller random access fil for læsning eller skrivning. Postlængden (recordlængden) henfører til en relativ fil, som kan være så lang som 254. "W" parameteren angives kun ved skrivning (PRINT#) i en sekventiel fil. Angives den ikke, forudsætter diskettedrevet, at der skal foretages en læsning. Relative filer er åbne for såvel læsning som skrivning samtidig.

Det logiske filnummer henfører til det nummer, som senere diskette operationer skal bruge ved læse- (input#) eller skrive- (print#) operationer med denne fil. Det logiske filnummer kan variere fra 1 til 255. Logiske filnumre, som er større end 128, sender automatisk en vognretur og linieskift med hver skrive (print#) kommando. Logiske filnumre mindre end 128 sender kun en vogn RETURN, som kan undertrykkes med et semikolon efter print#-kommandoen. Standardværdi for enheden er 8, for drevet 0.

EKSEMPLER:

DOPEN#1, "ADRESSE",W

Åbner sekventiel fil nummer 1 (ADRESSE) for skrivning.

DOPEN#2, "C128",D1,U9

Åbner den sekventielle fil nummer 2 (C128) for læsning på enhed nummer 9 drev nummer 1.

DOPEN#3, "BØGER",L128

Åbner den relative fil "BØGER" for læsning eller skrivning på enhed 8 med recordlængden 128 karakterer.

DRAW

Tegner prikker, linier og figurer på specificerede skærmområder

DRAW [farveområde],X1,Y1 [TO X2,Y2]...

Denne instruktion tegner individuelle prikker, linier og figurer.

Her følger parameter værdierne:

Farveområde	0 = højopløsnings baggrund 1 = højopløsnings forgrund 2 = flerfarve 1 3 = flerfarve 2
X1, Y1	Startkoordinater (skalasat)
X2, Y2	Slutkoordinater (skalasat)

X og Y værdierne kan placere pixelmarkøren på absolutte koordinater som f.eks. (100,100) eller på koordinater, som er relative i forhold til den tidligere position (+/-x og +/-y) af pixelmarkøren som f.eks. (+20, -10). Den ene akses koordinat (x eller y) kan være relativ og den anden absolut. Her er de mulige kombinationer på måder, hvorpå x og y koordinaterne kan angives:

x,y	absolut x, absolut y
+/-x,y	relativt x, absolut y
x,+/-y	absolut x, relativt y
+/-x,+/-y	relativt x, relativt y

Se også LOCATE kommandoen for information om pixelmarkøren.

EKSEMPLER:

DRAW 1, 100, 50 Tegner en prik
DRAW , 10, 10 TO 110,60 Tegner en linie
DRAW , 10,10 TO 10,60 TO 100,60 TO 10,10 Tegner en trekant

De kan udelade en parameter, men skal stadig medtage det komma, som skulle have fulgt efter denne parameter.

DSAVE

Lagrer en BASIC programfil på diskette

DSAVE "filnavn" [,Ddrev nummer][⟨ON|,⟩Uenheds nummer]

Denne kommando lagrer et program på diskette. (Brug SAVE for at lagre programmer på bånd.) Et programnavn på op til 16 karakterers længde skal tilføjes. Standardværdi for enhed er 8, for drev 0.

EKSEMPLER:

DSAVE "BANKPOSTER"

Lagrer programmet "BANKPOSTER" på diskette.

DSAVE (A$)

Lagrer det disketteprogram, hvis navn findes i variablen A$.

DSAVE "PROG 3",D1,U9

Lagrer programmet "PROG 3" på diskettestationen med enheds nummer 9, drev nummer 1. (dobbeltdrevsenhed)

DVERIFY

Sammenligner et program i hukommelsen med et på disketten.

DVERIFY "filnavn"[,Ddrev nummer][⟨ON|,⟩Uenheds nummer]

Denne kommando får Commodore 128 til at kontrollere programmet på det specificerede drev mod det program, som findes i hukommelsen.
Standardværdien for drevnummer er 0 og for enheden 8.

OBS! Hvis et grafisk område er allokeret efter en SAVE (eller reallokeret efter en SAVE), vil verify og Dverify give en fejlmelding. Teknisk set er dette korrekt. Fordi BASIC tekst blev flyttet fra sin originale placering, ændredes kædnings byten. Derfor vil verify, som udfører byte til byte sammenligning, gå galt, selv om programmet er i orden.
Vedrørende verificering af Binære data bedes De se VERIFY "filnavn",8,1 format under beskrivelsen af VERIFY kommandoen.

EKSEMPLER:

DVERIFY "C128"

Verificerer program "C128" på drev 0, enhed 8.

DVERIFY "SPRITES",D0,U9

Verificerer program "SPRITES" på drev 0, enhed 9.

*END

Angiver afslutning af programudførelse.

END

Når programmet opdager END instruktionen standses programudførelsen omgående. CONT kommandoen kan benyttes til at genstarte programmet fra næste instruktion (hvis en sådan findes) efter END instruktionen.

ENVELOPE

Definerer et musikalsk instruments indhylningskurve.

ENVELOPE n[,atk][,dec][,sus][,rel][,wf][,pw]

hvor:

n ..... Envelope nummer (0-9)
atk ..... Attack tid (0-15)
dec ..... Decay tid (0-15)
sus ..... Sustain niveau (0-15)
rel ..... Release tid (0-15)
wf ..... Bølgeform:
0 = trekant
1 = savtak
2 = variabel puls (firkant)
3 = støj
4 = ring modulation
pw ..... Puls bredde (0-4095)

En uspecificeret parameter vil beholde sin prædefinerede eller aktuelt omdefinerede værdi. Pulsbredde henfører kun til bredden på den variable puls bølgeform (wf=2) og bestemmes af formlen:

pwout=pw/40.95 (i pct.)

Commodore 128 indeholder følgende 10 envelope definitioner:

	n	A	D	S	R	wf	pw	instrument
ENVELOPE	0,	0,	9,	0,	0,	2,	1536	piano
ENVELOPE	1,	12,	0,	12,	0,	1		harmonika
ENVELOPE	2,	0,	0,	15,	0,	0		calliope
ENVELOPE	3,	0,	5,	5,	0,	3		tromme
ENVELOPE	4,	9,	4,	4,	0,	0		fløjte
ENVELOPE	5,	0,	9,	2,	1,	1		guitar
ENVELOPE	6,	0,	9,	0,	0,	2,	512	spinet
ENVELOPE	7,	0,	9,	9,	0,	2,	2048	orgel
ENVELOPE	8,	8,	9,	4,	1,	2,	512	trompet
ENVELOPE	9,	0,	9,	0,	0,	0		xylofon

For at spille prædefinerede musikalske instrument-envelopes, kan De simpelhen anføre envelope nummeret i kommandoen (se PLAY). Det er ikke nødvendigt at bruge ENVELOPE kommandoen, med mindre man ønsker at ændre i envelopen.

FAST

Indstiller maskinen til 2MHz operations mode.

FAST

Denne kommando initierer 2MHz mode og frakobler derved VIC 40 kolonners skærmen. Alle operationer (undtagen I/O) udføres væsentligt hurtigere. Grafik kan benyttes, men vil ikke være synlig, før en SLOW kommando tilføjes.

FETCH

Henter data fra udvidelses (RAM modul) hukommelse.

FETCH #bytes,intsa,expsa,expb

hvor bytes =	det antal bytes, der skal hentes fra hukommelsesudvidelsen (0-65535).
intsa =	startadresse i generel RAM (0-65535)
expb =	bank nummer i 64k RAM udvidelsen (0-15)
expsa =	startadresse i udvidelses RAM (0-65535)

FILTER

Definerer lyd (SID chip) filter parametere.

FILTER [freq][,lp][,bp][,hp][,res]

hvor:

freq ..... Filter cut-off frekvens (0-2047)
lp ..... Low-pass filter til(1), fra(0)
bp ..... Band-pass filter til(1), fra(0)
hp ..... High-pass filter til(1), fra(0)
res ..... Resonans (015)

Uspecificerede parametere giver ingen ændring i den aktuelle værdi.

De kan benytte mere end een filtertype ad gangen. For eksempel kan både low-pass og high-pass filtrene bruges sammen for at producere en notch- (eller band-reject) virkning. For at filteret skal have en hørlig effekt, skal mindst een filtertype vælges og mindst een stemme ledes gennem dette.

EKSEMPLER:

FILTER 1024,0,1,0,2

Sætter cut-off frekvensen til 1024, vælger band-pass filteret og et resonansniveau på 2.

FILTER 2000,1,0,1,10

Sætter cut-off frekvensen til 2000, vælger både low- og high-pass filtrene (til at udføre notch-reject) og sætter resonansniveauet til 10.

*FOR/TO/STEP/NEXT

Definerer en gentagende programløkke struktur.

FOR variabel = startværdi TO slutværdi [STEP forøgelse]

Denne instruktion arbejder sammen med NEXT instruktionen for at danne et programafsnit (en løkke), som repeteres et givet antal gange.
Dette er nyttigt, når noget skal optælles eller noget skal udføres et bestemt antal gange (som udskrivning).

Instruktionen gentager alle kommandoer, der findes mellem FOR og NEXT instruktionerne, i overensstemmelse med start og slut værdierne. Startværdien og slutværdien er begyndelsen og enden på tælleren for løkke variablen. Løkkevariablen tillægges eller fratrækkes gennem FOR/NEXT løkken.

Logikken i en FOR/NEXT instruktion er følgende. Først angives løkke variablens startværdi. Når programmet når til en linie med en NEXT instruktion, lægges STEP forøgelse (standard = 1) til løkkevariablens værdi og det undersøges, om denne er højere end løkkens slutværdi. Er løkkevariablen mindre end eller lig med slutværdien, udføres løkken igen, begyndende med instruktionen, der følger umiddelbart efter FOR instruktionen. Er løkkevariablen større end slutværdien, standses udførelsen af løkken og programmet går videre med den instruktion, som følger lige efter NEXT instruktionen. Det modsatte sker, hvis STEP værdien er negativ.

EKSEMPLER:

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

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

Første program udskriver tallene fra 1 til 10 efterfulgt af meddelelsen "JEG ER FÆRDIG! L = 11". Andet program udskriver tallene fra 10 ned til 1 og derefter "JEG ER FÆRDIG! L = 0".

Løkkens slutværdi kan efterfølges af ordet STEP og et andet tal eller en anden variabel. I så fald lægges den angivne værdi til hver gang i stedet for 1. Dette giver mulighed for at tælle baglæns, med intervaller eller med forøgelse forskellig fra een.

Brugeren kan danne løkker indeni hinanden. Sådanne løkker kaldes 'nested loops'. Man skal med denne teknik passe på, at den sidste løkke som skal starte, bliver den første der slutter.

EKSEMPEL:

```
10 FOR L = 1 TO 100
20 FOR A = 5 TO 11 STEP .5
30 NEXT A
40 NEXT L
```

FOR...NEXT løkken i linierne 20 og 30 er anbragt inden i een i linierne 10 og 40. Brug af STEP værdien .5 er brugt for at illustrere den kendsgerning at decimal-punkt angivelse er mulig. Se også NEXT instruktionen.

*GET

Modtager inddata fra tastaturet, en karakter ad gangen, uden at vente på, at en tast nedtrykkes.

GET variabelliste

GET instruktionen aflæser de tastanslag, som foretages af brugeren. Efterhånden som brugeren indtaster tegn, lagres disse i computerens hukommelse i et areal, som kaldes tastaturbufferen. Op til ti tegn kan lagres her, tegn ud over dette antal vil gå tabt. GET instruktionen henter første tegn fra bufferen og flytter resten opad for at skabe plads for nye. Hvis ingen karakterer findes i bufferen, hentes et nul (en tom karakter). GET standser ikke programudførelsen, selv om der ikke findes karakterer i bufferen (se GETKEY). Ordet GET følges af et variabelnavn, enten numerisk eller streng.

Hvis C128 forventer at modtage en numerisk tekst og en ikke-numerisk tast nedtrykkes, standser programmet og en fejlmeddelelse vises. GET instruktionen kan også medtages i en løkke for at kontrollere et tomt resultat. I et sådant tilfælde kan en GETKEY instruktion også bruges. Se GETKEY for yderligere information. GET og GETKEY instruktionerene kan kun udføres i et program

EKSEMPEL:

10 DO:GETA$:LOOP UNTIL A$="A"

Denne linie venter på, at Atasten skal nedtrykkes for at fortsætte.

20 GET B,C,D

Variablerne B, C og D hentes fra tastaturet uden at der ventes på nedtrykning af en tast.

GETKEY

Modtager inddata fra tastaturet, en karakter ad gangen. Der ventes på nedtrykning af en tast.

GETKEY variabelliste

GETKEY instruktionen ligner GET instruktionen meget. Til forskel fra GET instruktionen venter GETKEY på, at brugeren indtaster en karakter på tastaturet. Dette gør, at computeren venter på, at et enkelt tegn indtastes. Instruktionen kan kun udføres i et program.

EKSEMPEL:
10 GETKEY A$

Denne linie venter på, at en tast nedtrykkes. Nedtrykning af en tilfældig tast får programmet til at fortsætte. GETKEY kan også bruges til aflæsning af numeriske taster.

OBS: GETKEY kan ikke returnere en tom karakterstreng.

10 GETKEY A$,B$,C$

Denne linie venter på, at tre alfanumeriske karakterer indtastes på tastaturet.

*GET#

Modtager inddata fra bånd, diskette eller RS232.

GET # filnummer, variabelliste

Denne instruktion modtager een karakter ad gangen fra en forud åbnet fil. Ellers arbejder den som GET instruktionen. Instruktionen kan kun benyttes i et program.

EKSEMPEL:

10 GET#1,A$

Dette eksempel modtager een karakter, som lagres i variablen A$, fra fil nummer 1. Det forudsættes i eksemplet, at fil 1 er åbnet. Se OPEN/DOPEN instruktionerne.

GO64

Skifter til C64 mode.

GO64

Instruktionen skifter fra C128 til C64 mode. Spørgsmålet "Are You Sure?" (Er De Sikker?) vises som svar på GO64 instruktionen. Tastes der Y, går det aktuelt indlæste program tabt, og kontrollen overgives til C64 mode; i modsat fald, hvis enhver anden tast nedtrykkes, forbliver computeren i C128 mode. Instruktionen kan bruges enten i direkte mode eller i et program. I programmode vises spørgsmålet ikke.

*GOSUB

Kalder en subrutine fra det angivne linienummer.

GOSUB linienummer

Denne instruktion svarer til GOTO instruktionen, bortset fra, at Commodore 128 returnerer til det sted, den kom fra, når subrutinen er afsluttet. Opdages en linie med en RETURN instruktion, hopper programmet tilbage til den instruktion, der følger umiddelbart efter GOSUB instruktionen.

Målet for en GOSUB instruktion kaldes en subrutine. En subrutine er værdifuld, hvis en opgave skal udføres flere gange i et program. I stedet for at gentage programafsnittet igen og igen, oprettes en subrutine, og GOSUB indsættes på de passende steder i programmet. Se også RETURN instruktionen.

EKSEMPEL:

```
20 GOSUB 800
:
:
799 END
800 PRINT "HEJ DER":RETURN
```

Dette eksempel kalder den subrutine, som begynder i linie 800 og udfører den. Alle subrutiner skal afsluttes med en RETURN instruktion.

*GOTO/GO TO

Overfører programudførelse til det angivne linienummer.

GOTO linienummer

Opdages en GOTO instruktion i et program, udfører computeren den instruktion, specificeret af linienummeret i GOTO instruktionen.
Bruges GOTO i direkte mode udføres det program, som begynder i det angivne linienummer, uden at slette variablerne. Dette svarer til RUN kommandoen, bortset fra, at variabelværdierne ikke slettes.

EKSEMPLER:

```
10 PRINT "COMMODORE"
20 GOTO 10
```

GOTO i linie 20 vil gentage linie 10 uendeligt, til RUN/STOP bliver nedtrykket.

GOTO 100

Starter programmet, som begynder i linie 100, uden at slette variabel lagerområdet.

GRAPHIC

Vælger en grafisk mode.

1) GRAPHIC mode [,clear][,s] eller
2) GRAPHIC CLR

Denne instruktion sætter Commodore 128 i en af de seks grafiske modes:

MODE	BESKRIVELSE
0	40 kolonners tekst
1	standard højopløsnings grafik
2	standard højopløsnings grafik (delt skærm)
3	flerfarve højopløsnings grafik
4	flerfarve højopløsnings grafik (delt skærm)
5	80 kolonners tekst

CLEAR parameteren angiver om højopløsnings skærmen slettes (lig med 1) ved programudførelse, eller lades intakt (lig med

0). S parameteren indikerer nummeret på begyndelseslinien for delt skærm i mode 2 eller 4 (flerfarve eller standard højopløsnings skærm mode). Startliniens standardværdi for delt skærm er 19.

Ved udførelse reserverer GRAPHIC 1-4 et 9K højopløsningsområde. Starten på BASIC tekstområdet flyttes ovenfor højopløsnings området og ethvert Basic program relokeres automatisk. Dette område forbliver allokeret, selv om brugeren returnerer til TEXT mode (GRAPHIC 0). Hvis clear tilføjelsen specifieres som 1, slettes skærmen. GRAPHIC CLR kommandoen frigiver 9K højopløsnings området og gør det atter tilgængeligt for BASIC tekst.

EKSEMPLER:

GRAPHIC 1,1

Vælger standard højopløsning og sletter højopløsningen.

GRAPHIC 4,0,10

Vælger delt skærm, flerfarve højopløsning. Sletter ikke højopløsningen og begynder delt skærm i linie 10.

GRAPHIC 0

Vælger 40 kolonners tekst

GRAPHIC 5

Vælger 80 kolonners tekst

GRAPHIC CLR

Sletter og frigiver højopløsnings-området.

HEADER

Formatterer en diskette

HEADER "diskettenavn" [,lid] [,Ddrev nummer] [⟨ON|,⟩Uenhed]

hvor "diskettenavn" er ethvert navn på op til 16 tegns længde.

D.v.s. alle alfanumeriske tegn, De måtte ønske at anvende. Brug ikke mellemrum.

Før en ny diskette kan bruges for første gang, skal den formatteres med HEADER kommandoen. En tidligere anvendt diskette kan slettes for genbrug, også ved brug af HEADER kommandoen.

Når De indtaster en HEADER kommando i direkte mode, fremkommer følgende spørgsmål på skærmen: "ARE YOU SURE?" (Er De Sikker?). Dette spørgsmål fremkommer ikke i programmode.

Kommandoen opdeler disketten i sektioner kaldet blokke (blocks). Den fremstiller en indholdsfortegnelse, kaldet directory eller catalog, på disketten. "id." nummeret er to vilkårlige karakterer. Giv hver diskette sin eget id. Vær varsom med anvendelsen af HEADER kommandoen, fordi den sletter alle lagrede data.

En hurtigere formattering opnås ved at undlade id., idet den gamle id. så benyttes. Dette kan naturligvis kun gøres på en tidligere formatteret diskette, da denne hurtige formattering sletter indholdsfortegnelsen i stedet for at formattere disketten. Standardværdien er drev 0, enhed 8.

Som en sikkerhedsforanstaltning spørger systemet "ARE YOU SURE?" før Commodore 128 udfører operationen. Tast "Y" for at udføre HEADER, eller en tilfældig tast for at afbryde.

HEADER kommandoen læser diskette kommandokanalen, og opdages en fejl fremkommer fejlmeldingen "?BAD DISK ERROR".

HEADER kommandoen svarer til BASIC 2.0 kommandoen:

OPEN 1,8,15,"N0:diskettenavn,id"

EKSEMPLER:

HEADER "MIN DISKETTE", I51,D0

Formatterer disketten og giver den navnet "MIN DISKETTE" med id. 51 på drev 0, enhed nr. 8 (standard).

HEADER "POSTERINGER",I45,D1,ON U9

Formatterer disketten "POSTERINGER" med id. 45 på drev 1, enhed 9.

NB: En variabelstreng kan ikke benyttes som id.

HELP

- viser fejlramt linie

HELP

HELP kommandoen bruges, hvis en fejl i et program er rapporteret.
Hvis HELP skrives i 40 kolonners mode, listes den fejlramte linie, og den fejlramte del af linien vises negativt. I 80 kolonners mode vises den fejlramte del af linien understreget. Tryk på HELP-tasten skriver automatisk HELP og giver RETURN.

*IF/THEN
IF/THEN/ELSE

Frembringer et betinget udtryk og udfører programafsnit, som er afhængige af udtrykkets resultat.

IF udtryk THEN instruktioner [:ELSE klausul]

IF...THEN instruktionen frembringer et BASIC udtryk og vælger een af to mulige aktioner, afhængig af udtrykkets resultat. Er udtrykket sandt (opfyldt) udføres de instruktioner, som følger efter THEN.
Dette kan være en hvilken, eller hvilke, som helst BASIC instruktion(er). Er udtrykket falsk (uopfyldt) fortsætter programmet med den programlinie, som følger umiddelbart efter den programlinie, der indeholder IF instruktionen, uanset om ELSE klausulen findes. Hele IF...THEN instruktionen må højst fylde 160 karakterer.

Se også BEGIN/BEND.

ELSE klausulen skal, hvis den medtages, stå på samme linie som IF...THEN delinstruktionen gør, og skal adskilles fra THEN med et kolon. Medtages en ELSE klausul, udføres denne kun, hvis udtrykket er falsk. Det dannede udtryk kan være en varia-

bel eller formel, i hvilket tilfælde det anses for værende sandt, hvis det ikke er nul, og falsk, hvis det er nul. I de fleste tilfælde indeholder udtrykket relaterede operatorer (=, <, >, <=, >=, <>).

IF...THEN instruktionen kan antage to forskellige former:

IF udtryk THEN linienummer
eller
IF udtryk GOTO linienummer

Disse udformninger overfører programudførelsen til det angivne linienummer, hvis udtrykket er sandt. I modsat fald fortsættes med den programlinie, som følger lige efter linien med IF instruktionen.

EKSEMPEL:

50 IF X >0 THEN PRINT "OK" : ELSE END

Denne linie kontrollerer værdien af X. Er X større end 0, udføres instruktionen, der følger lige efter nøgleordet THEN (PRINT "OK") og ELSE klausulen ignoreres. Er X mindre end, eller lig med, 0, udføres ELSE klausulen og instruktionen efter THEN ignoreres.

```
10 IF X = 10 THEN 100
20 PRINT "X er ikke lig med 10"
:
99 STOP
100 PRINT "X er lig med 10"
```

Dette eksempel kontrollerer værdien af X. Hvis X er lig med 10, overføres programkontrollen til linie 100 og meddelelsen "X er lig med 10" vises. Er X ikke lig med 10, fortsætter programmet med linie 20, C128 skriver "X er ikke lig med 10" og programmet stopper.

NB: ELSE udvidelsen kan ikke bruges i C64 mode.

*INPUT

Modtager en datastreng eller et tal fra tastaturet og venter på, at brugeren skal trykke på RETURN.

INPUT ["streng";] variabelliste

INPUT instruktionen beder brugeren om at indtaste data, mens programmet kører og placerer disse data i en eller flere variabler. Programmet standser, viser et spørgsmålstegn (?) på skærmen og venter på, at brugeren indtaster svaret og trykker på RETURN. Ordet INPUT følges af en 'spørgsmåls-streng' og et variabelnavn eller en liste med variabelnavne, adskilte med kommaer. Meddelelsen i strengen (i anførelsestegn) efterspørger den oplysning, brugeren skal indtaste. Medtages en sådan meddelelse, skal der være et semikolon (;) efter strengens afsluttende anførelsestegn.

Hvis INPUT består af mere end een variabel, skal de adskilles med kommaer. Computeren spørger efter de resterende værdier ved at vise to spørgsmålstegn (??). Trykkes der på RETURN, uden at der angives en inddata værdi, bibeholder INPUT variablen værdien fra tidligere INPUT. INPUT instruktionen kan kun bruges i et program.

EKSEMPEL:

```
10 INPUT "INDTAST VENLIGST ET TAL";A
20 INPUT "OG DERES NAVN";A$
30 PRINT A$", DE INDTASTEDE TALLET";A
```

*INPUT#

Indlæser data fra en fil til computerens hukommelse.

INPUT# filnummer, variabelliste

Denne instruktion arbejder som INPUT, men tager data fra en fil, som allerede er åbnet med OPEN, normalt fra diskette eller bånd i stedet for fra tastaturet. Der anvendes ingen spørgsmåls-strenge. Instruktionen kan kun bruges i et program.

EKSEMPEL:

```
10 OPEN 2,8,2,"DATAFIL,S,R"
20 INPUT#2,A$,C,D$
```

Denne instruktion indlæser de data, som er lagret i filen "DATA-FIL" og lagrer dem i variablerne A$, C og D$.

KEY

Definerer eller lister funktions tasternes indhold

KEY [tast nummer, streng]

På Commodore 128 findes 8 funktionstaster (F1-F8), som kan anvendes af brugeren: fire uskiftede og fire skiftede. (lower og upper case).
Hver gang en af disse taster nedtrykkes, kan Commodore udføre en bestemt funktion eller operation. En tasts funktion kan defineres med data eller en kommando eller serier af kommandoer. KEY uden tilføjelse af en specificeret parameter giver en listning af alle nuværende funktioner på tasterne. De data, som er tildelt tasten, skrives, når den nedtrykkes. Den maksimale længde af alle definitioner til sammen er 246 tegn.

EKSEMPLER

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

Dette får computeren til at vælge (40 kolonners) tekst skærm og liste programmet, hvis "F7" tasten nedtrykkes (i direkte mode).
CHR$(13) er ASCII karakteren for RETURN. For 'skiftet RETURN' bruges CHR$(141) og for "ESCape" bruges CHR$(27). For at indsætte anførelsestegn ("") i en KEY streng bruges CHR$(34). Det er muligt at omdefinere tasterne i et program. For eksempel:

10 KEY 2,"PRINT DS$"+ CHR$(13)

Dette får computeren til at kontrollere og vise diskettedrevs kanalernes variabler (PRINT DS$) hver gang F2 funktionstasten nedtrykkes.

For at give tasterne deres standardværdi trykkes på RESET knappen.

*LET

Tildeler en variabel en værdi

[LET] variabel = udtryk

Ordet LET bruges sjældent i programmer, da det er unødvendigt. Når som helst en variabel bliver defineret eller får tildelt en værdi, er LET altid underforstået. Det variabelnavn, som modtager resultatet af en beregning findes på lighedstegnets venstre side. Tallet, strengen eller formlen på højre side. De kan kun tildele een værdi med hver LET instruktion. For eksempel er LET A = B = 2 ugyldig.

EKSEMPEL:
10 LET A = 5

Giver den numeriske variabel A værdien 5

20 B = 6

Giver den numeriske variabel B værdien 6

30 C=A*B+3

Giver den numeriske variabel C den værdi der er resultatet af 5 gange 6 plus 3.

40 D$ = "HALLO"

Strengvariablen D$ tildeles strengen "HALLO"

*LIST

Udlister et BASIC program, som findes i hukommelsen

LIST [linie|første-|første-sidste|-sidste]

LIST kommandoen sætter brugeren i stand til at gennemse linier i et BASIC program, som er indtastet eller indlæst til Commodore 128's hukommelse. Brugt alene (uden angivne linienumre) giver Commodore 128 en komplet udlistning af programmet på skærmen. Udlistningen kan gøres langsommere ved at holde COMMODORE tasten nedtrykket, der kan holdes pause med CONTROL S eller SCROLL (pausen afbrydes ved tryk på en eller anden tast), eller standses ved at nedtrykke RUN/STOP. Hvis ordet LIST følges af et linienummer, viser Commodore 128 kun denne linie. Skrives LIST efterfulgt af to tal adskilt af en bindestreg (-), viser Commodore 128 alle linier fra det første til det sidst anførte nummer. Skrives LIST efterfulgt af et tal og en bindestreg, vises alle linier fra dette linienummer til programslut. Og hvis LIST skrives efterfulgt af en bindestreg, og derefter et tal, vises linierne fra programstart til det anførte linie-

nummer. Ved anvendelse af disse variationer, kan enhver del af et program undersøges eller på en enkel måde vises på skærmen for modifikation. I Commodore 128 mode kan LIST bruges i et program og dette program kan fortsætte med CONT kommandoen.

EKSEMPLER:

LIST

Viser et helt program

LIST 100-

Viser fra linie 100 til programslut

LIST 10

Viser kun linie 10

LIST -100

Viser linierne fra programstart til og med linie 100.

LIST 10-200

Viser linierne 10 til og med 200.

*LOAD

Indlæser et program fra en ydre enhed som f.eks. diskettedrev eller Datassette.

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

Dette er den kommando som skal anvendes, hvis et program, som er lagret på kassettebånd eller diskette, skal bruges. Filnavnet kan være et programnavn på til 16 tegns længde, angivet i anførelsestegn. Navnet skal efterfølges af et komma, efter anførelsestegnet, og et tal, der virker som et enhedsnummer og viser, hvor programmet er lagret (diskette eller bånd). Anføres intet nummer, forudsætter Commodore 128, at enhedsnummeret er 1, hvilket er båndstationen.

'Relocate flag' er et tal (0 eller 1) der er afgørende for, hvortil i hukommelsen et program indlæses. Er værdien 0, indlæser Commodore 128 programmet til begyndelsen af BASIC programområdet. Et flag på 1 gør computeren opmærksom på, at der skal indlæses fra det sted, der blev gemt. Relocate flags standardværdi er 0. Programparameteren 1 anvendes normalt, når der indlæses maskinsprogs programmer.

Den enhed, der oftest bruges sammen med LOAD kommandoen, er diskettestationen. Dette er enhed nummer 8, og derfor er DLOAD kommandoen nemmere at anvende, når der arbejdes med diskettedrev.

Hvis LOAD indtastes uden noget følgende argument, men kun af RETURN, forudsætter Commodore 128, at der skal indlæses fra bånd, og meddelelsen "PRESS PLAY ON TAPE" fremkommer på skærmen. Nedtrykkes PLAY på båndoptageren begynder Commodore 128 at lede efter et program på bånd. Når den findet et, skriver Commodore 128 FOUND "filnavn" på skærmen. Hvis det fundne program ønskes indlæst trykkes på COMMODORE tasten (C=). Hvis søgning skal fortsættes trykkes på mellemrumstangenten. Når programmet er indlæst, kan det køres, udlistes eller ændres. Tryk på mellemrumstangent virker ikke i C64 mode.

EKSEMPLER:

LOAD

Indlæser det næste program fra bånd.

LOAD "HALLO"

Gennemsøger båndet for programmet "HALLO" og indlæser dette, hvis det findes.

LOAD A$,8

Leder på diskette efter et program, hvis navn findes i variablen A$. Svarer til DLOAD(A$).

LOAD "HALLO",8

Leder på diskettestationen efter et program ved navn "HALLO". Svarer til DLOAD "HALLO".

LOAD "MASKKOD",8,1

Indlæser maskinsprogs-programmet "MASKKOD" til den adresse, hvorfra det blev lagret.

LOAD kommandoen kan bruges i et BASIC program for at finde og udføre det næste program på et bånd eller en diskette. Dette kaldes kædning (chaining).

LOCATE

Positionerer højopløsnings pixelmarkøren på skærmen

LOCATE x, y

LOCATE instruktionen kan placere pixelmarkøren (PC) på enhver specificeret pixel koordinat på skærmen.

Pixelmarkøren (PC) er den koordinat på højopløsnings-skærmen, hvorfra tegning af cirkler, kasser, linier og punkter og hvorfra farvelægning påbegyndes. PC strækker sig fra X og Y koordinaterne 0,0 til 320,200. PC er ikke synlig som tekstmarkøren, men den kan styres med de grafiske instruktioner (BOX, CIRCLE, DRAW etc.). Pixelmarkørens standardposition er den koordinat, der angives som X og Y i hver enkelt grafisk kommando. Så LOCATE kommandoen behøver ikke at angives.
X og Y værdierne kan placere pixelmarkøren på absolutte koordinater som f.eks. (100,100) eller på koordinater, som er relative i forhold til den tidligere position (+/-x og +/-y) af pixelmarkøren som f.eks. (+20, -10). Den ene akses koordinat (x eller y) kan være relativ og den anden absolut. Her er de mulige kombinationer på måder, hvorpå x og y koordinaterne kan angives:

x,y	absolut x, absolut y
+/-x,y	relativt x, absolut y
x,+/-y	absolut x, relativt y
+/-x,+/-y	relativt x, relativt y

EKSEMPEL:

LOCATE 160,100

Positionerer PC i midten af højopløsningsskærmen. Intet vil kunne ses, før noget tegnes.

PC kan lokaliseres med RDOT(0) funktionen for at finde X-koordinaten og RDOT(1) for at finde Y-koordinaten. Farvesource på PC plads kan findes med PRINT RDOT(2).

MONITOR

Adgang til Commodore 128 maskinkode monitor

MONITOR

SE APPENDIKS J OM COMMODORE 128 MASKINSPROGS MONITOR

MOVSPR

Positionerer eller flytter sprite på skærmen.

1) **MOVSPR nummer,x,y** Placerer den angivne sprite i absolut koordinat x,y.

2) **MOVSPR nummer +|-x,+|-y** Flytter sprite relativt i forhold til pixelmarkørens aktuelle position.

3) **MOVSPR nummer,x;y** Flytter sprite afstanden x i vinklen y i relation til dens aktuelle position.

4) **MOVSPR nummer,x vinkel #y fart** Flytter sprite med en vinkel (x) i relation til den originale koordinater, i urets retning og med den anførte hastighed (y).

hvor: **nummer** er spritens nummer (1-8)
〈**,x1,y1**〉 er sprite lokationens koordinat

vinkel er vinklen (0-360) på bevægelsen i urets retning i relation til spritens originale koordinat.

fart er en hastighed (0-15) hvormed spriten bevæger sig.

Denne instruktion placerer en sprite et bestemt sted på skærmen i henhold til SPRITE koordinat skemaet (ikke højopløsningsskemaet) eller igangsætter sprite bevægelse med en angivet hastighed.
Se MOVSPR i afsnit 6 for et diagram over sprite koordinat systemet.

EKSEMPLER:

MOVSPR 1,150,150

Positionerer sprite 1 tæt ved skærmens centrum, x,y koordinat 150,150.

MOVSPR 1,+20,-30

Flyt sprite 1 20 koordinater mod højre og 30 koordinater op.

MOVSPR 4,-50,+100

Flyt sprite 4 50 koordinater mod venstre og 100 koordinater nedad.

MOVSPR 5,45 #15

Flyt sprite 5 i en 45 graders vinkel i urets retning, i relation til dens originale x og y koordinater. Spriten flyttes med højeste hastighed (15).

OBS! De må indsætte en fart på 0 (nul) for at standse spritens bevægelse.

*NEW

Sletter programmer og variabler i hukommelsen

NEW

Denne kommando sletter hele det program, som findes i hukommelsen og fjerner alle variabler, som kan have været brugt. Hvis programmet ikke var lagret andetsteds, er det tabt og må indtastes igen. Vær derfor forsigtig med at anvende denne kommando. NEW kommandoen kan også bruges som en instruktion i et BASIC program. Imidlertid vil der ske det, at når Commodore 128 når til denne linie, slettes programmet og alt går i stå.

*ON

Vilkårsafhængig forgrening til en specificeret programlinie ud fra resultatet af det angivne udtryk.

ON udtryk ⟨GOTO|GOSUB⟩ linienummer 1 [,linienummer 2,...]

Denne instruktion kan få GOTO og GOSUB instruktionerne til at virke som specialversioner af IF instruktionen. Ordet ON efterfølges af et udtryk, derefter et af de to nøgleord GOTO eller GOSUB og en liste med linienumre adskilt med kommaer. Hvis resultatet af udtrykket er 1, udføres første linie i listen. Er resultatet 2, udføres anden linie o.s.v. Hvis resultatet er 0, eller større end antallet af tal i listen, fortsætter programmet med den instruktion, der følger umiddelbart efter ON instruktionen. Hvis tallet er negativt, resulterer dette i en ILLEGAL QUANTITY ERROR.

EKSEMPEL:

```
10 INPUT X:IF X<0 THEN 10
20 ON X GOSUB 30,40,50,60
25 STOP
30 PRINT "X = 1":RETURN
40 PRINT "X = 2":RETURN
50 PRINT "X = 3":RETURN
60 PRINT "X = 4":RETURN
```

Hvis X = 1 sender ON kontrollen til det første linienummer i listen (30). Hvis X = 2 sender ON kontrollen til anden linie (40) o.s.v.

*OPEN

Åbner filer for inddata eller uddata

OPEN logisk filnummer, enhedsnummer [,sekundær adresse] [<,"filnavn,filtype,mode"]|[,cmd streng]

OPEN instruktionen sætter Commodore 128 i stand til at få tilgang til (access) filer på enheder som en diskettestation, en Datassette båndstation, en printer, eller selv til Commodore 128 skærmen. Ordet OPEN efterfølges af et logisk filnummer, som er det nummer, til hvilket alle andre BASIC input/output instruktioner vil referere, som f. eks. PRINT#, INPUT#. Værdien skal ligge fra 1 til 255.

Det næste tal, kaldet enhedsnummeret, følger efter det logiske filnummer. Enhed nummer 0 er Commodore 128 tastaturet; 1 er båndstationen; 3 er Commodore 128 skærmen; 4-7 er printere; og 8-11 er reserveret for diskettestationer. Det er ofte en god ide at bruge samme filnummer som nummeret på enheden, da det så er let at huske, hvad der er hvad.

Efter enhedsnummeret kan der følge en tredie parameter, som kaldes den sekundære adresse. Bruger man kassettebånd, kan den være 0 for læsning, 1 for skrivning og 2 for skrivning med END-OF-TAPE markering efter skrivning. Bruges diskette, refererer tallet til kanal nummeret. Læs i manualen til Deres diskettestation for mere information om kanaler og kanalnumre. I forbindelse med printeren bruges den sekundære adresse til at vælge bestemte programmeringsfunktioner.

Der kan endvidere specificeres et filnavn for diskette eller bånd, ELLER en streng, efter den sekundære adresse, hvilket kunne være en kommando til diskette/bånd eller navnet på filen på bånd eller diskette. Hvis filnavnet angives, refererer type og mode udelukkende til diskettefiler. Filtyper kan være: PROGRAM, SEQUENTIAL, RELATIVE og USER; modes er READ og WRITE.

EKSEMPLER:

10 OPEN 3,3
Åbner skærmen som fil nummer 3.

20 OPEN 1,0
Åbner tastaturet som fil nummer 1.

30 OPEN 1,1,0,"DOT"
Åbner kassettestationen for læsning, som fil nummer 1, med DOT som filnavn.

OPEN 4,4

Åbner printeren som fil nummer 4.

OPEN 15,8,15

Åbner diskettestationens kommandokanal som fil nummer 15, med sekundær adresse 15. Sekundær adresse 15 er reserveret som diskettedrevets fejlkanal.

5 OPEN 8,8,12,"TESTFIL,SEQ,WRITE"

Åbner en sekventiel fil på disketten af filen ved navn TESTFIL, med fil nummer 8, sekundæradresse 12.

Se endvidere: CLOSE, CMD, GET#, INPUT# og PRINT# instruktionerne samt systemvariablerne ST, DS og DS$.

PAINT

Farvelægger et område

PAINT [farveområde],x,y[,mode]

hvor:

farveområde	0 = højopløsnings forgrund 1 = højopløsnings baggrund 2 = flerfarve 1 3 = flerfarve 2
x,y	startkoordinat, skaleret (standard ved pixelmarkør (PC).
mode	0 = farvelæg et område defineret af det valgte farveområde. (standardværdi) 1 = farvelæg et område med en tilfældig farve, som ikke er lig med baggrunden.

PAINT kommandoen udfylder et område med farve. Den udfylder området rundt om det angivne punkt, indtil en grænse med samme farve opdages. Hvis De for eksempel tegner en cirkel i forgrundsfarve, brug da PAINT i cirklen, fra hvor koordinaten antager baggrundsfarven. Commodore 128 vil kun farvelægge, hvor det farveområde, der er angivet i PAINT instruktionen, er forskelligt fra x og y pixel koordinatens område. Den kan ikke farvelægge punkter, hvor farveområderne for PAINT instruktionen og pixel koordinaten er sammenfaldende. x og y koordinaten må ligge nøjagtig på omridslinien af den figur, De vil have farvelagt, og farveområdet for den begyndende pixel koordinat og det angivne farveområde skal være forskellige.

X og Y værdierne kan placere pixelmarkøren på absolutte koordinater som f.eks. (100,100) eller på koordinater, som er relative i forhold til den tidligere position (+/-x og +/-y) af pixelmarkøren som f.eks. (+20, -10). Den ene akses koordinat (x eller y) kan være relativ og den anden absolut. Her er de mulige kombinationer på måder, hvorpå x og y koordinaterne kan angives:

x,y	absolut x, absolut y
+/-x,y	relativt x, absolut y
x,+/-y	absolut x, relativt y
+/-x,+/-y	relativt x, relativt y

Se også LOCATE kommandoen for information om pixelmarkøren.

EKSEMPLER:

eks. 1:

```
10 CIRCLE1,160,100,65,50 Tegner omridset af en cirkel
20 PAINT 1,160,100       Udfylder cirklen med farve fra område 1
                         (VIC forgrund), idet det forudsættes,
                         at punkt 160,100 er farvelagt i bag-
                         grundsfarven (område 0).
10 BOX 1,10,10,20,20     Tegner omridset af en kasse
20 PAINT 1,15,15         Udfylder kassen med farve fra område 1,
                         idet det forudsættes at punkt 15,15 er
                         farvelagt i baggrundsfarven (0).
30 PAINT 1,+10,+10       Farvelægger skærmens forgrundsområde
                         fra den koordinat, der er relativ i
                         forhold til pixelmarkørens tidligere
                         position, plus 10 i både lodret og
                         vandret plan.
```

eks. 2:

```
10 COLOR 0,1:COLOR 1,2:COLOR 2,5:COLOR 3,7
20 GRAPHIC 3,1              Flerfarvegrafik
30 CIRCLE 1, 80,100,30      Tegner cirkel
40 CIRCLE 1,80,100,35       Regner cirkel
50 BOX 2,80,100,90,110,45,1 tegner kasse
60 PAINT 3,0,100,0          farvelægger baggrund helt
```

eks. 3: som eks. 2, men udskift linie 60 med:

```
60 PAINT 3,70,100,1         farvelægger indre cirkel
```

PLAY

Definerer og spiller musikalske noder og elementer

PLAY "[Vn][On][Tn][Un][Xn][elementer]"

hvor: Vn = Stemme (n = 1-3)
On = Oktav (n = 0-6)
Tn = Instrument envelope (n = 0-9)
Standard: 0 = piano
1 = harmonika
2 = calliope
3 = tromme

4 = fløjte
5 = guitar
6 = cembalo (Se Envelope kommando)
7 = orgel
8 = trompet
9 = xylofon

Un = Volumen (n = 0-9)
Xn = Filter til (n=1), fra (n=0)
A,B,C,D,E,F,G .... Noder

Elementer:

#	kryds for noden (*)
$	.....b for noden (*)
W	.....Helnode
H	.....Halvnode
Q	.....Fjerdedelsnode
I	.....Ottendedelsnode
S	.....Sekstendedelsnode
.	.....Forlænget node (*)
R	.....Pause
M	.....Vent til alle stemmer er færdige med takt

(*) Skal foranstilles de enkelte noder.

PLAY instruktionen giver Dem mulighed for at vælge stemme, oktav og envelope (incl. de ti foruddefinerede instrument envelopes), volumen og de noder, De ønsker at spille. Alle disse kontroller skal angives i anførelsestegn.

Alle elementer med undtagelse af R og M sættes foran noden i en PLAY streng.

EKSEMPLER:

PLAY "V1O4T0U5X0CDEFGAB"

Spiller noderne C, D, E, F, G, A og B i stemme 1, oktav 4, envelope 0 (piano) med styrke 5, filter fraslået.

PLAY "V1O4T0U5X0CQD#E$FHGIAB"

Spiller noderne: C, kvartnoderne D og kryds for E, b for F, halvnoden G og ottendedelsnoderne A og B.

*POKE

Ændrer indholdet af en RAM hukommelses adresse.

POKE adresse, værdi

POKE instruktionen tillader ændring af enhver værdi i Commodore 128 RAM, og tillader modifikation af mange af Commodore 128 input/output registrene. Nøgleordet POKE følges altid af to parametere. Den første angiver en adresse i Commodore 128's hukommelse. Denne oplysning kan have en værdi fra 0 til 65535. Anden parameter er en værdi fra 0 til 255, som bliver placeret i den angivne adresse, og udskifter enhver værdi, som måtte være der i forvejen. Hukommelsesadressens indhold er afgørende for bitmønstret i den aktuelle adresse (C128 mode). POKE sker til den aktuelt valgte RAM bank. POKE adressen afhænger af BANK nummeret. Se BANK for eksempler på passende BANK konfigurationer.

EKSEMPEL:

10 POKE 53280,1

Ændrer kantfarve (bank 15 i C128 mode)

OBS! PEEK, en funktion beslægtet med POKE, som returnerer indholdet af en angiven hukommelsesadresse, findes under FUNKTIONER.

*PRINT

Uddata til tekstskærmen.
PRINT [liste]
PRINT instruktionen er den vigtigste uddata instruktion i BASIC. Da PRINT instruktionen er den første BASIC instruktion, de fleste mennesker lærer at bruge, findes der mange variationer af denne instruktion. Ordet PRINT kan efterfølges af følgende:

Karakterer i anførelsestegn ("tekst")
Variabelnavne (A, B, A$, X$)
Funktioner (SIN(23),ABS(33))
Skilletegn (;,)

Karakterer i anførelsestegn udskrives i nøjagtig den rækkefølge, hvormed de er anført i strengen. Variabelnavne får den værdi,

de inholder, udskrevet. Dette kan være et tal eller en streng. Funktioner får ligeledes udskrevet deres talværdi.

Skilletegn bruges som hjælp til at få data placeret pænt på skærmen.
Kommaet tabulerer til nærmeste 10-tegns kolonne, mens semikolon udskriver individerne lige ved siden af hinanden (se også sektion 3, Udskrivning af tal). Hvert af skilletegnene kan anføres som sidste symbol i en instruktion. Dette resulterer i, at næste PRINT instruktion opfører sig, som var den en fortsættelse af den foregående.

EKSEMPLER:

10 PRINT "HALLO"	HALLO
20 A$ = " DER":PRINT"HALLO";A$	HALLO DER
30 A = 4:B = 2: PRINT A + B	6
40 J = 41:PRINT J:PRINT J-1	41 40
50 PRINT A;B;:D=A+B:PRINT D;A-B	**4 2 6 2**

Se endvidere POS, SPC og TAB funktionerne.

*PRINT#

Der er kun meget lidt forskel på denne instruktion og PRINT.

Vigtigst er, at ordet PRINT# efterfølges af et tal, som refererer til en i forvejen åbnet datafil. Tallet er fulgt af et komma og en liste indeholdende individer, som skal være uddata til filen. Komma og semikolon virker på samme måde, med dannelse af mellemrum, på printere, som de gør i PRINT instruktionen. Visse enheder fungerer ikke med TAB og SPC.

EKSEMPEL:

```
10 OPEN 4,4
20 PRINT#4,"HALLO DER",A$,B$
```

Udskriver "HALLO DER" og variablerne A$ og B$ på printeren.

```
10 OPEN 2,8,2,"DATAFIL,S,W"
20 PRINT#2,A,B$,C,D
```

Udskriver datavariablerne A, B$, C og D på diskette, fil nr. 2.

OBS! PRINT# kommandoen kan bruges alene til at klargøre kanalen til enheden, efter udskrivning via CMD, før filen lukkes, som her:

```
OPEN 4,4
CMD4
LIST
PRINT#4
CLOSE4
```

Se endvidere CMD kommandoen.

PRINT USING

Udskrivning i defineret format.

PRINT [#filnummer]USING "formatliste";udskriv liste

Denne instruktion definerer udskrivningsformatet af strenge og numeriske individer til tekstskærm, printer eller andre enheder. Formatet angives i anførelsestegn. Dette er formatlisten. Tilføj derefter et semikolon og en liste over, hvad der skal udskrives i det anførte format. Liste kan bestå af variabler eller aktuelle værdier adskilte med kommaer.

EKSEMPEL:

```
5 X=32: Y=100.23: A$="CAT":B$="COMPUTER"
6 F$="* #=##########  * #$##.## *"+
CHR$(13):REM CHR$(13) ER RETURN
10 PRINT USING "$##.##";13.25,X,Y
20 PRINT USING "###〉#";"CBM",A$
30 PRINT USING F$;A$,X,B$,Y;
```

Når dette program køres, udskriver linie 10:

$13.25 $32.00 $*****	Der udskrives fem asterisker (*****) istedet for Y's værdi, fordi Y har fem cifre og dets format derfor ikke stemmer overens med formatlisten.

Linie 20 udskriver:

CBM CAT	Laver, som beskrevet i format-listen, to mellemrum, før udskrivning.

Formatstreng KARAKTER	Bruges med NUMERISK	STRENG
Nummer tegn (#)	x	x
Plus tegn (+)	x	
Minus tegn (-)	x	
Decimalpunktum (.)	x	
Komma (,)	x	
Dollartegn ($)	x	
Fire piltegn (↑↑↑↑)	x	
Ligheds tegn (=)		x
Større end tegn (>)		x

Nummertegnet (#) reserverer plads til en enkelt karakter i udskrivningsfeltet. Hvis dataindividet indeholder flere karakterer end der er angivet #-tegn i formatfeltet, fyldes hele feltet med asterisker (*); ingen karakterer udskrives.

EKSEMPEL:

10 PRINT USING "# # # #";X

Dette format udskrives for følgende X værdier:

X = 12.34	12
X = 567.89	568 (Obs: Tallet oprundes)
X = 123456	****

Ved et STRENGindivid forkortes strengens data indenfor feltets grænser. Der udskrives kun så mange karakterer, som der er nummertegn (#) i formatet. Afskæring sker fra højre.
Plus og minustegnene (+ -) kan benyttes enten i første eller sidste position i formatfeltet, men ikke begge steder samtidig. Plustegnet udskrives, hvis tallet er positivt. Minustegnet udskrives, hvis tallet er negativt.

Hvis der bruges minustegn, og tallet er positivt, vises en 'blank' karakter på minustegnets plads.
Hvis hverken plus- eller minustegn anvendes i et numerisk dataindivids formatfelt, udskrives et minustegn før første ciffer eller dollartegn, hvis tallet er negativt. Er tallet positivt, udskrives intet tegn. Dette betyder, at en ekstra karakter, minustegnet, udskrives, hvis tallet er negativt. Er der flere karakterer, end

der er plads til i feltet, som er defineret med pundtegn og plus/minus tegn, indtræffer 'overflow' og feltet udfyldes med asterisker (*).

Et decimalpunkt symbol angiver positionen af decimalpunktet i tallet. Der kan kun være et decimalpunkt i et formatfelt. Angives der ikke noget decimalpunkt, afrundes der til nærmeste heltal og der udskrives uden decimalpladser.

Hvis et decimalpunkt defineres, må antallet af cifre (incl. minustegn, hvis værdien er negativ) ikke overstige det antal nummertegn, der er afsat før decimalpunktet. Er der for mange cifre, indtræffer overflow og feltet udfyldes med asterisker (*).

Et komma (,) tillader placering af kommaer i numeriske felter. Kommaets position i formatlisten indikerer, hvor kommaer vil vise sig i et udskrevet tal. Kun kommaer i et tal bliver udskrevet. Ubrugte kommaer til venstre for første ciffer, vises som fyldkarakterer. Mindst eet nummertegn skal være foranstillet første komma i et felt.

Såfremt kommaer specificeres i et felt, og tallet er negativt, udskrives et minustegn som første karakter, selv om karakterpositionen er specificeret som et komma.

EKSEMPLER:

FELT	UDTRYK	RESULTAT	KOMMENTAR
##.#	-.1	-0.1	Foranstillet 0 tilføjes
##.#	1	1.0	Decimalnul tilføjet
####	-100.5	-101	Afrunding, ingen decimalpladser
####	-1000	****	Overflow fordi der ikke er plads til fire cifre og et minustegn.
###.	10	10.	Decimalpunkt tilføjet
#$##	1	$1	$-tegn foranstillet

Et dollartegn ($) symbol viser, at et dollartegn vil blive udskrevet med tallet. Skal dollartegnet være flydende (altid være placeret før tallet), skal mindst et nummertegn specificeres foran dollartegnet.
Specificeres dollartegnet uden foranstillet nummertegn, udskrives dollartegnet i den position, hvori det er anført i formatfeltet. Hvis kommaer og plus/minustegn specificeres i et formatfelt med et dollartegn, udskriver programmet et komma eller tegn

foran dollartegnet. PIL op (↑) eller indskudstegn bruges til at specificere, at tallet skal udskrives i E + format (videnskabelig notation). Sammen med de fire piltegn skal et nummertegn bruges for at angive feltbredden. Indskudstegnene skal stå efter nummertegnet i feltet. Der skal angives fire indskudstegn, hvis et tal skal udskrives i E notation. Hvis færre end fire indskudstegn specificeres, optræder en syntaksfejl. Specificeres mere end 4 indskudstegn, bruges kun de fire første. Det femte indskudstegn oversættes som et ikke-tekst symbol. Man kan angive + eller - tegn efter indskudstegnene, hvis dette ønskes. Lighedstegn (=) bruges for at centrere en streng i et felt. Feltbredden angives med antallet af karakterer (nummertegn og lighedstegn) i formatfeltet. Hvis strengen indeholder færre karakterer end feltbredden, centreres strengen i feltet. Indeholder strengen flere karakterer, end der er plads til i feltet, afskæres karaktererne fra højre og hele feltet udfyldes af strengen. Brug af større-end tegn (>) højrejusterer en streng i et felt.

PUDEF

Omdefinerer symboler i PRINT USING instruktionen

PUDEF "nnnn"

Hvor "nnnn" er enhver kombination af karakterer, op til fire ialt. PUDEF sætter Dem i stand til at omdefinere ethvert af de følgende fire symboler i PRINT USING instruktionen: blanke, kommaer, decimalpunktummer og dollartegn. Disse fire symboler kan ændres til andre tegn ved at placere det nye tegn i den korrekte position i PUDEF kontrolstrengen.

Position 1 er fyldkarakteren. Standard er en 'blank'. Her sættes en ny karakter, som ønsker vist i stedet for en blank position.

Position 2 er kommategnet. Standardværdi er komma.

Position 3 er decimalpunktummet. Standardværdi er punktum.

Position 4 er dollartegnet. Standardværdi er dollartegn.

EKSEMPLER:

10 PUDEF "*" Skriver * i stedet for blanke.

20 PUDEF "〈" Skriver 〈 i stedet for kommaer.

OBS: Alle positioner frem til den, der skal ændres, skal angives.

*READ

Læser data fra DATA instruktioner og sender disse til computerens hukommelse (mens programmet udføres med RUN).

READ variabelliste

Denne instruktion henter information fra DATA instruktioner og lagrer dem i variabler, hvorfra data kan bruges af et kørende program. READ instruktionens variabelliste kan indeholde såvel strenge som tal. Pas på ikke at læse strenge, når READ instruktionen forventer et tal. Dette vil give en TYPE MISMATCH fejlmeddelelse.

Data i DATA instruktioner læses sekventielt. Hver READ instruktion kan læse eet eller flere dataindivider. Hver variabel i READ instruktionen kræver et dataindivid. Findes det ikke, optræder en OUT OF DATA fejl.

I et program kan De READ data og derefter genlæse dem ved at tilføje RESTORE instruktionen. RESTORE stiller den sekventielle datapointer tilbage til start, hvorefter data atter kan indlæses. Se RESTORE.

EKSEMPLER:

```
10 READ A, B, C                  Læser de tre første dataindivider,
20 DATA 3, 4, 5                  som skal være numeriske.
10 READ A$,B$,C$                 Læser de første tre streng-
20 DATA JOHN, POUL, HANS         variabler fra DATAinstruktionen.
10 READ A,B$,C                   Læser en numerisk værdi, en streng
20 DATA 1200,BENTE,345           variabel og en anden numerisk
                                 variabel.
```

RECORD

Positionerer relative filpointere.

RECORD# logisk filnummer, postnummer [,bytenummer]

Denne instruktion positionerer en relativ filpointer for at udvælge en angiven byte (karakter) fra en post i en angiven relativ fil. Det logiske filnummer kan ligge i området fra 1 til 255. Postnummeret kan ligge i området 1 - 65535. Bytenummeret i området 1 t.o.m. 254. Se Deres diskettestations manual for detaljer om relative filer.

Sættes værdien på postnummeret højere end sidste postnummer i filen, sker følgende:

Ved en skrive (PRINT#) operation skabes ekstra poster for at udvide filen til det ønskede postnummer.

Ved en læse (INPUT#) operation, returneres en tom post og meddelelsen "RECORD NOT PRESENT ERROR" fremkommer.

EKSEMPEL:

```
10 DOPEN#2,"KUNDER"
20 RECORD#2,10,1
30 PRINT#2,A$
40 DCLOSE#2
```

Dette eksempel åbner, i linie 10, en eksisterende relativ fil ved navn KUNDER som fil nummer 2. Linie 20 positionerer den relative filpointer på første byte i post nummer 10. Linie 30 udskriver oplysningen, A$, til filen.

RECORD kommandoen accepterer variabler som parametere. Det er ofte hensigtsmæssigt at placere en RECORD kommando i en FOR...NEXT eller DO løkke. Se endvidere DOPEN.

REM

Kommentarer eller bemærkninger om programliniens funktion.

REM kommentar

REM instruktionen er en oplysning til den, der læser en pro-

gramliste. REM kan forklare en programdel, give oplysninger om forfatter etc. REM instruktioner har ingen indflydelse på programmets virkemåde, bortset fra, at programmet bliver længere og derved bruger mere hukommelsesareal. Hvad der står til højre for nøgleordet REM bliver ikke oversat af computeren til en udførbar instruktion. Derfor kan ingen udførbar instruktion følge efter en REM i samme linie.

EKSEMPEL:

10 NEXT X : REM Denne linie forøger X

RENAME

Ændrer navnet på en diskettefil

RENAME D0,"gammelt navn"TO"nyt navn"[Uenhedsnummer]

Denne kommando bruges til at ændre navnet på en diskettefil fra et gammelt filnavn til et nyt. Diskettedrevet ændrer ikke filnavn på en fil, hvis den er åben.

EKSEMPLER:

RENAME D0,"TEST" TO "SLUTTEST"

ændrer filnavnet "TEST" til "SLUTTEST"

RENAME D0,(A$) TO (B$),U9

ændrer det filnavn, der specificeres i A$ på drev 0, enhed 9. Husk, at hvis et variabelnavn anvendes som filnavn, skal det omgives af paranteser.

RENUMBER

Omnummererer linierne i et BASIC program

RENUMBER [nyt startlinienummer][,forøgelse] [,gl. startlinienummer]

Den nye startlinie er nummeret på programmets første linie efter omnummerering. Dens standardværdi er 10. 'forøgelse' er det antal ubrugte linier, der er mellem linienumrene, f. eks. 10, 20, 30, etc. Også her er standardværdien 10. Det gamle startlinie nummer er det linienummer, hvorfra omnummereringen skal be-

gynde. Dette tillader omnummerering af en del af et program. Dette nummers standardværdi er programmets første linienummer. Kommandoen kan kun bruges i direkte mode.

Hvis der angives en reference til et ikke eksisterende linienummer, vil en "UNRESOLVED REFERENCE ERROR" meddelelse fremkomme. Hvis omnummerering udvider programmet ud over dets grænser viser meddelelsen "OUT OF MEMORY" sig. Hvis RENUMBER prøver at generere et linienummer på 64000 eller derover, fremkommer fejlmeldingen "LINE NUMBER TOO LARGE". Disse fejltyper ødelægger ikke programmet.

EKSEMPLER:

RENUMBER

Omnummererer programmet begyndende i 10, med en linienummerforøgelse på 10.

RENUMBER 20, 20, 1

Programmet omnummereres, begyndende i linie 1. Linie 1 ændres til linie 20 og andre linier nummereres med spring på 20.

RENUMBER, , 65

Omnummererer med spring på 10, begyndende i linie 65. Linie 65 ændres til linie 10. Udelades en parameter, skal der være et komma på dens plads. Der må ikke findes linier mellem 10 og 64, incl.

Gem altid Deres programmer, før RENUMBER bruges, fordi meget lange programmer kan forårsage systemfejl, hvis der omnummereres med store linienumre. Vær endvidere opmærksom på, at store programmer bør omnummereres i FAST mode, da omnummerering kan tage meget lang tid.
(op til 30 minutter for et 55K program i FAST mode).

Hvis De har en 40-kolonners skærm bruges FAST:RENUMBER---- RETURN og derefter indtastes SLOW RETURN . Mens systemet arbejder vil man ikke kunne se, at der sker noget til jobbet er fuldført.

Har De en 80-kolonners skærm skal 80-kolonners skærm vælges

før der tastes FAST.

*RESTORE

Flytter READ pointeren tilbage, så DATA kan genlæses.

RESTORE [linienummer]

Under udførelse af et program tilbagestilles den pointer, der peger på det individ i en DATA instruktion, som skal læses, til første individ i instruktionen. Dette giver mulighed for at genlæse data. Følger der et linienummer efter RESTORE instruktionen, sættes pointeren på første dataindivid efter den anførte linie. Ellers tilbagestilles pointeren til BASIC programmets start. I C64 findes ingen mulighed for at angive linienummeret, så her kan kun RESTOREs til programmets begyndelse.

EKSEMPLER:

```
10 FOR I = 1 TO 3
20 READ X
30 T = X + T
40 NEXT
45 PRINT T
50 RESTORE
60 GOTO 10
70 DATA 10,20,30
```

Dette eksempel læser data i linie 70 og lagrer dem i den numeriske variabel X. Den beregner summen af alle numeriske dataindivider. Når alle data er læst, tre gange gennem løkken, bliver pointeren RESTOREt til programstart og vender tilbage til linie 10 for gentaget udførelse.

```
10 READ A,B,C
20 DATA 100,500,750
30 READ X,Y,Z
40 DATA 36,24,38
50 RESTORE 40
60 READ S,P,Q
70 PRINT A,B,C
80 PRINT X,Y,Z
90 PRINT S,P,Q
```

Dette eksempel RESTOREr DATA pointeren til det første individ i linie 40. Når linie 60 er udført, vil 36,24,38 læses fra linie 40, da det ikke er nødvendigt at læse linie 20's data igen.

RESUME

Definerer hvorfra et program skal fortsætte efter at en fejl er fundet.

RESUME [linienummer|NEXT]

Denne instruktion bruges for at genstarte et program efter fejlsøgning med TRAP. Angives ingen parametere, vil RESUME genudføre den linie, i hvilken fejlen optrådte. RESUME NEXT fastlægger, at der skal begyndes med udførelse af den instruktionslinie, der følger umiddelbart efter den, der var fejlbehæftet. Hvis RESUME efterfølges af et linienummer, vil der gås til (GOTO) den specificerede linie og programudførelsen vil genoptages herfra. RESUME kan kun bruges i program mode.

EKSEMPEL:

```
10 TRAP 100
20 INPUT "hjerteINDTAST ET TAL";A (hjerte = SHIFT/CLR)
30 B=100/A
40 PRINT "RESULTATET =";B: PRINT "SLUT"
50 PRINT "VIL DE PRØVE IGEN (J/N?)":
   GETKEY Z$:IF Z$="J" THEN 20
60 STOP
100 INPUT "INDTAST ET NYT TAL (IKKE NUL)";A
110 RESUME
```

Dette eksempel fanger en "division med nul-fejl" i linie 30, hvis 0 indtastes i linie 20. Indtastes et 0, går programmet til linie 100, hvor De af computeren bedes om at indtaste et tal forskelligt fra 0.
Linie 110 går tilbage til linie 30 for at udføre beregningen. Linie 50 spørger, om De vil gentage programmet. Hvis det ønskes, skal der indtastes et J.

*RETURN

Returnerer fra en subrutine.

RETURN

Denne instruktion sættes altid sammen med GOSUB instruktionen. Når programmet opdager en RETURN instruktion, returneres til den instruktion, der følger umiddelbart efter den sidste GOSUB kommando, der er blevet udført. Hvis en sådan GOSUB ikke findes, vises fejlmeddelelsen RETURN WITHOUT GOSUB ERROR og programmet stopper. Alle subrutiner skal slutte med en RETURN instruktion.

EKSEMPEL:

```
10 PRINT "GÅ TIL SUBRUTINE"
20 GOSUB 100
30 PRINT "SLUT PÅ SUBRUTINE
.
.
.
90 STOP
100 PRINT "SUBRUTINE 1"
110 RETURN
```

Dette eksempel kalder subrutinen i linie 100, som udskriver meddelelsen "SUBRUTINE 1" og returnerer til linie 30, resten af programmet.

***RUN** - udfører BASIC program

1) RUN [linienummer]

2) RUN "filnavn" [,Ddrev nummer] [,Uenhedsnummer]
(kun C128 mode)

Når et program er indlæst (med LOAD) eller indtastet til hukommelsen, eksekverer RUN kørslen af programmet. Før programafviklingen påbegyndes, sletter RUN alle variabler i programmet. Følger et nummer efter RUN, begynder kørslen ved dette linienummer. Efterfølges RUN kommandoen af et filnavn, indlæses den angivne fil fra diskette og eksekveres, uden at brugeren skal foretage sig yderligere. RUN kan bruges i et program. Standardværdien på drev er 0 og på enhed 8.

EKSEMPLER:

RUN	**Starter programkørsel fra laveste nummer i det aktuelle program.**
RUN 100	**Starter programkørsel fra linie 100**
RUN"PRG1"	**DLOAD'er "PRG1" fra diskenhed #8 og kører dette program fra starten**
RUN (A$)	**Indlæser det program, som er navngivet af variablen A$ og udfører det.**

*SAVE

Lagrer program fra hukommelsen på diskette eller bånd.

SAVE ["filnavn"][,enheds nummer][,EOT flag]

Denne kommando lagrer det program, som netop ligger i hukommelsen, på kassettebånd eller diskette for senere genindlæsning. Hvis kun ordet SAVE skrives, efterfulgt af RETURN, vil en unavngivet fil blive lagret på kassettebåndet. Da bånd er et sekventiel system, er det op til brugeren at sikre sig, før SAVE benuyttes, at det anvendte bånd ikke indeholder vigtige oplysninger. (se VERIFY). Skal programmet have et navn, skal dette angives i anførelsestegn (eller som en strengvariabel) umiddelbart efter ordet SAVE. Et filnavn kan være op til 16 karakterer langt.

Obs: Hvis der skal lagres på diskette, SKAL der angives et filnavn.
I modsat fald fremkommer fejlmeldingen "MISSING FILE NAME ERROR".

For angivelse af enhedsnummeret (1 for bånd) indtastes et komma efterfulgt af enhedsnummeret umiddelbart efter det sidste anførelsestegn i filnavnet.

Den sidste parameter (EOT) tillader enhedsnummeret igen, adskilt med et komma. Den har ingen betydning i forbindelse med lagring på diskette og kan have en af fire følgende værdier (0-3), når den bruges ved kassettebåndslagring.

0 standardværdi

1 gør at relocate funktionen i LOAD ikke virker, d.v.s. at filen altid vil blive genindlæst til den adresse, hvorfra den blev gemt.

2 Skriver en End Of Tape markering efter filens afslutning. Forsøg på LOAD efter filslut på en fil, der er lagret på denne måde, vil frembringe meldingen "FILE NOT FOUND ERROR".

3 Lagrer i ikke relokerbart format (1) og skriver EOT markering (2).

OBS: Hvis enhedsnummer eller EOT parameter angives, skal filnavn medtages. På bånd kan dette være et 'tomt' navn (""). Se eksempler.

EKSEMPLER:

SAVE	**Lagrer et program på bånd uden at give det et navn.**
SAVE "HALLO"	**Lagrer på bånd under navnet HALLO.**
SAVE A$,8	**Lagrer på diskette med navnet i variablen A$.**
SAVE "HALLO",8	**Lagrer på diskette med navnet HALLO. (svarer til DSAVE "HALLO")**
SAVE "HALLO",1,2	**Lagrer på bånd med navnet HALLO og sætter et END-OF-TAPE mærke efter programmet.**
SAVE "",1,3	**Lagrer på bånd, uden navn, sætter en EOT markering efter programmet og tillader ikke relokering af programmet under indlæsning.**

SCALE

Ændrer skalastørrelsen i grafisk mode.

SCALE n [,xmax,ymax]

hvor:

n = 1 (til) eller 0 (fra)

Standard højopløsning:
xmax ligger i området 320 - 32767, standardværdi 1023.
ymax ligger i området 200 - 32767, standardværdi 1023.

Flerfarve mode:
xmax ligger i området 160 - 32767, standardværdi 2047.
ymax ligger i området 200 - 32767, standardværdi 1023.

Ændrer skalaen på højopløsnings-skærmens koordinater i både flerfarve og højopløsning. MOVSPR kommandoens koordinater bliver også skalasat. Samler mange logiske punkter til eet fysisk punkt.
Dette er nyttigt, hvis De har brug for at udlægge data over et stort værdiområde; det vil ikke være til hjælp, hvis De har en stort datasamling, som udelukkende består af høje værdier.

Da flerfarve bruger 2 fysiske pixels på x-aksen pr. punkt er dens normale skærm:

x = 0 til 159 y = 0 til 199

i modsætning til højopløsning:

x = 0 til 319 y = 0 til 199

Hvis De ønsker af bruge samme koordinater for flerfarve og højopløsning benyt så SCALE 1,640,200 efter oprettelse af en flerfarveskærm eller benyt standardværdierne for begge.

OBS: GRAPHIC kommandoen slår skalasætning fra, hvilket svarer til GRAPHIC...:SCALE 0.

EKSEMPEL:

```
10 GRAPHIC 1 :GOSUB 50
20 SCALE 1 :GOSUB 50
30 SCALE 1,5000,5000 :GOSUB 50
40 END
50 CIRCLE 1,160,100,60:RETURN
```

SCNCLR

Sletter skærmindhold.

SCNCLR modenummer

Numrene er som følger:

MODE NUMMER	MODE
0	40 kolonners (VIC) tekst
1	højopløsning
2	højopløsning, delt skærm
3	flerfarve højopløsning
4	flerfarve højopløsning, delt skærm
5	80 kolonners (8563) tekst

Denne instruktion, brugt uden argumenter, sletter skærmens indhold, uanset om der er tale om et grafisk eller et tekstbillede. Bitarealet er det det samme for både højopløsning og flerfarve, de forskellige modenumre vælger, hvad ellers kræver sletning. F.eks. 40-kolonners tekst (2 og 4) farve (3 og 4).

EKSEMPLER:

SCNCLR 5	Sletter 80 kolonners tekstskærm
SCNCLR 1	Sletter (VIC) højopløsningsskærm
SCNCLR 4	Sletter (VIC) flerfarve højopløsnings delt skærm

SCRATCH

Fjerner fil fra diskettens indholdsfortegnelse

SCRATCH "filnavn"[,Ddrev nummer][⟨ON|,⟩Uenheds nummer]

Fjerner en fil fra diskettens indholdsfortegnelse (directory). Som en sikkerhedsforanstaltning spørger systemet "Are you sure?", før Commodore 128 foretager sletningen. Såfremt sletningen ønskes, foretaget tastes "Y", fortrydes, tastes "N". Brug denne kommando til at fjerne uønskede filer og skab derved mere plads på disketten.

Obs! Filnavnet må gerne indeholde søgekarakterne * eller ?. Standardværdi for drevnummer er 0 og for enhed 8.

EKSEMPEL:

SCRATCH "MIT PROGRAM",D1

Fjerner filen MIT PROGRAM fra disketten i drev 1, enhed 8.

SLEEP

Forsinker programudførsel i et specificeret tidsrum.

SLEEP N

hvor N er sekunder 0 ⟨ N ⟨ 65535

SLOW

Returnerer Commodore 128 til 1 MHz tilstand.

Commodore 128 er i stand til at bruge 8502 mikroprocessoren med en hastighed på 1 eller 2 megahertz.

SLOW kommandoen nedsætter mikroprocessoren fra 2 til 1 megahertz.
FAST kommandoen virker den modsatte vej. Commodore 128 kan udføre kommandoer væsentligt hurtigere ved 2 MHz end ved 1 MHz.
Imidlertid kan 40-kolonners skærm ikke bruges ved 2MHz.

SOUND

Udsender lydeffekter og musikalske noder.

SOUND v,f,d[,dir][,m][,s][,w][,p]

hvor:
- v = stemme (1..3)
- f = frekvensregister værdi (0..65535)
- d = varighed (0..32767)
- dir = trinretning (0(op),1(ned) eller 2 (osciller) standardværdi=0.
- m = minimum frekvens (hvis sweep bruges) (0..65535) standardværdi=0.
- s = trinværdi for effekter (0..65535) standardværdi=0.
- W = bølgeform (0=trekant, 1=savtak, 2=firkant, 3=støj) Standardværdi=2.
- p = pulsbredde (0..4095) Standard=2048

SOUND kommandoen er en hurtig og let måde at danne lydeffekter og musikalske toner på. De tre krævede parametere v, f og d indstiller stemme, frekvens og varighed på lyden. Varigheden måles i enheder kaldet 'jiffies'. Tres 'jiffies' er lig med 1 sekund.

SOUND kommandoen kan 'sweepe' gennem en serie frekvenser, som lader lydeffekterne passere gennem et nodeområde. Retningen af 'sweep' bestemmes med parameteren DIR. Minimumfrekvensen på sweep sættes med M og trinværdien med S. Vælg den passende bølgeform med W og specificer P som bredden af den variable puls bølgeform, hvis den er valgt med W.

EKSEMPLER:

SOUND 1,40960,60

Spiller en lyd i frekvensen 40960 med stemme 1 i 1 sekund

SOUND 2,20000,50,0,2000,100

Udsender en lyd, der sweeper gennem frekvenser begyndende i 2000 og går trinvist opad i enheder på 100.

SOUND 3,5000,90,2,3000,500,1

Dette eksempel udsender et lydområde begyndende med en minimumsfrekvens på 3000, til 5000, i trin på 500. Sweep's retning er svingende (oscillerende). Den valgte bølgeform er savtak og som stemme er valgt nummer 3.

SPRCOLOR

Sætter flerfarve 1 og/eller flerfarve 2 for alle sprites.

SPRCOLOR [smcr-1] [,smcr-2]

hvor

[smcr-1] sætter flerfarve 1 for alle sprites og
[smcr-2] sætter flerfarve 2 for alle sprites.

Hver af disse parametre kan være enhver farve fra 1 - 16.

EKSEMPLER:

SPRCOLOR 3,7

Sætter sprite flerfarve 1 til rød og flerfarve 2 til blå.

SPRCOLOR 1,2

Sætter sprite flerfarve 1 til sort og flerfarve 2 til hvid.

SPRDEF

Går til SPRite DEFinitions mode for at skabe og redigere sprites.

SPRDEF

Kommandoen definerer sprites interaktivt.

Når man bruger SPRDEF kommandoen, vises et sprite arbejdsareal på skærmen i størrelsen 24 karakterer bredt og 21 karakterer højt. Hver karakterposition i denne gitterstruktur svarer til en sprite pixel i den sprite, der vises til højre for arbejdsarealet. Her er en summering af SPRite DEFinitions mode operationerne og de taster, der udfører dem:

BRUGER INPUT	BESKRIVELSE
RETURN tast	Forlader sprite designer mode, men kun ved SPRITE NUMBER?
1-8	Vælger et sprite nummer
A	Slukker og tænder aut. markørbevægelse
CRSR taster	Flytter markør
RETURN tast	Flytter markør til næste linies begyndelse
HOME tast	Flytter markør til øverste venstre af gitter
CLR tast	Sletter hele skemaet
1-4	Vælger farveområde
⟨CTRL⟩ 1-8	Vælger sprite forgrundsfarve (1-8)
Commodoretast 1-8	Vælger sprite forgrundsfarve (9-16)
STOP tast	Sletter ændringer og returnerer til SPRITE NUMBER?
SHIFT RETURN	Gemmer sprites og returnerer SPRITE NUMBER?
X	Udvider sprite i X retningen
Y	Udvider sprite i Y retningen
M	Flerfarve sprite
C	Kopierer spritedata fra en sprite til en anden.

NB: Brug af denne kommando vil slette højopløsningsskærm.

SPRITE

Tænder og slukker, farvelægger, udvider og sætter skærmprioriteter for en sprite.

SPRITE ⟨nummer⟩[,on/off][,fgnd][,prioritet][,x-udv] [,y-udv][,mode]

SPRITE instruktionen styrer de fleste karakteristika ved en sprite.

PARAMETER	BESKRIVELSE
nummer	Spritenummer (1-8)
on/off	Tænd eller sluk sprite
fgnd	Sprite forgrundsfarve (1-16)
prioritet	Prioritet skal være 0, hvis spriten skal gå foran objekter på skærmen. 1 hvis spriten skal gå bag objekter på skærmen.
x-udv	Horisontal udvidelse TIL (1), FRA (0)
y-udv	Vertikal udvidelse TIL (1), FRA (0)
mode	Standard sprite (0) eller flerfarve sprite (1)

Uspecificerede parametere i efterfølgende sprite instruktioner antager foregående SPRITE instruktions karakterer. En sprites karakteristika kan kontrolleres med RSPRITE funktionen.

EKSEMPLER:

SPRITE 1,1, 3	Tænd sprite 1 og farv den rød.
SPRITE 2,1,7,1,1,1	Tænd sprite 2, farv den blå, lad den passere bag objekter på skærmen og udvid den horisontalt og vertikalt.
SPRITE 6,1,1,0,0,1,1	Tænd sprite 6 og farv den sort. Det første 0 fortæller computeren, at spriten skal vises foran objekter på skærmen. Det næste 0 og det følgende 1 udvider spriten vertikalt. Det sidste 1-tal specificerer flerfarve mode. Brug SPRCOLOR kommandoen for valg af spritens flerfarver.
SPRITE 7,,,,1	Sætter horisontal udvidelse for sprite 7 og andre parametere bevarer deres værdier.

SPRSAV

Lagrer spritedata fra en tekststreng variabel til et sprite område eller omvendt.

SPRSAV original,destination

Denne kommando overfører en sprite's udseende fra en strengvariabel til et sprite lagerområde. Den kan også overføre data fra lagerområdet til en strengvariabel. Enten originalen eller destinationen kan være et spritenummer eller en strengvariabel, men de kan ikke begge være strengvariabler. Skal De flytte en streng til en sprite, bruges kun de første 63 bytes af dataarealet. Resten ignoreres, da en sprite højst kan indeholde 63 databytes.

EKSEMPLER:

SPRSAV 1,A$
Overfører mønstret fra sprite 1 til en streng kaldet A$.

SPRSAV B$,2
Overfører data fra strengvariablen B$ til sprite 2.

SPRSAV 2,3
Overfører data fra sprite 2 til sprite 3.

OBS: SPRSAV spritestreng danner en streng i samme format som SSHAPE for at få en sprite til at passe til en højopløsningsskærm. Strengen bliver 67 karakterer lang.

SSHAPE/GSHAPE

Lagrer/genindlæser figurer til/fra strengvariable SSHAPE og GSHAPE bruges til at lagre og indlæse rektangulære arealer med flerfarve eller højopløsnings skærmbilleder til/fra BASIC strengvariabler. Den kommando, der skal benyttes for at lagre et skærmområde i en strengvariabel er:

SSHAPE strengvariabel,x1,y1[,x2,y2]

hvor:

strengvariabel	Navnet på den streng, der skal lagres i.
x1,y1	Hjørnekoordinat (0,0 til 319,199) (skaleret)
x2,y2	Modsatte hjørnekoordinat (standardværdi er PC.

Fordi BASIC begrænser strenge til 255 karakterer, er det areal, der kan gemmes, også begrænset. Den krævede strengstørrelse kan beregnes med een af følgende (uskalerede) formler:

L(flerfarve)=INT ((ABS(x1-x2) + 1)/4 + .99) * (ABS(y1-y2) +1) + 4
L(højopl)=INT ((ABS(x1-x2) + 1)/8 + .99) * (ABS(y1-y2) + 1) + 4

NB: Under forudsætning af, at x2-x1 er 23, og at grafisk mode er højopløsning (mode 1 eller 2), kan en streng dannet af SSHAPE bruges til fremstilling af en sprite. (Se SPRSAV).

Den kommando, der skal anvendes for at genindlæse data fra en streng variabel og vise den på et specificeret skærmområde er:

GSHAPE strengvariabel [x,y][,mode]

hvor:

strengvariabel	Den streng, der indeholder figuren
x,y	Øverste venstre koordinat (0,0 til 319,199) som fortæller hvor på skærmen figuren skal tegnes. (skaleret - standardværdi er PC)
mode	Genplacerings mode: 0: placer figuren som den er (standard) 1: figur i reverse (negativ) 2: OR figur med areal 3: AND figur med areal 4: XOR figur med areal

Genplacerings mode tillader Dem at ændre data i strengvariablen, så De kan spejlvende den, udføre en logisk OR, eksklusiv OR eller AND operation med figurens udseende. X og Y værdierne kan placere pixelmarkøren på absolutte koordinater som f.eks. (100,100) eller på koordinater, som er relative i forhold til den tidligere position (+/x og +/-y) af pixelmarkøren som f.eks. (+20, -10). Den ene akses koordinat (x eller y) kan være relativ og den anden absolut. Her er de mulige kombinationer på måder, hvorpå x og y koordinaterne kan angives:

x,y	absolut x, absolut y

+/-x,y	relativt x, absolut y
x,+/-y	absolut x, relativt y
+/-x,+/-y	relativt x, relativt y

Se også LOCATE kommandoen for information om pixelmarkøren.

EKSEMPLER:

SSHAPE A$,10,10	Lagrer et rektangulært areal fra koordinaterne 10,10 til pixelmarkørens position i strengvariablen A$.
SSHAPE B$,20,30,47,51	Lagrer et rektangulært område fra øverste venstre koordinat (20,30) til nederste højre koordinat (47,51) i strengvariablen B$.
GSHAPE A$,120,20	Genfinder figuren, der er i strengvariablen A$ og viser den fra øverste venstre hjørne i koordinat 120,20.
GSHAPE B$,30,30,1	Genfinder figuren i strengvariablen B$ og viser den fra øverste venstre koordinat 30,30. Da genplacerings mode er angivet med 1, vises figuren i reverse (negativ).

OBS! Undgå at bruge mode 1 til 4 med flerfarvede figurer.
De vil få uforudsigelige resultater.

STASH

Flytter indhold fra vært hukommelse til udvidelses RAM
STASH byteantal,intsa,expsa,expb
Se under FETCH kommandoen for beskrivelse af parametere.

*STOP

Standser programudførelse.

Denne instruktion standser programmet. En meddelelse "BREAK LINE IN XXX" viser sig (kun i program mode), hvor XXX er det linienummer, der indeholder STOP kommandoen. Programmet kan genstartes fra den instruktion, som måtte følge efter STOP, hvis CONT kommandoen benyttes øjeblikkeligt, uden at nogen form for redigering foretages. STOP instruktionen bruges ofte i forbindelse med fejlsøgning i et program.

SWAP

Udskifter indholdet af vært RAM med indholdet fra udvidelses RAM.

SWAP byteantal,intsa,expsa,expb

Se under FETCH kommandoen for beskrivelse af parametere.

*SYS

Kalder og eksekverer en maskinsprogs subrutine i den angivne adresse

SYS adresse [,a][,x][,y][,s]

Denne instruktion udfører et kald af en maskinsprogs-subrutine fra en given adresse i en hukommelseskonfiguration, som er etableret i henhold til BANK kommandoen. Det er muligt, før subrutinen kaldes, at indlæse argumenterne a, x, y og s til akkumulatoren, x, y og statusregistrene. Adresseområdet er 0 - 65535. Programmet påbegynder udførelsen af maskinkodeprogrammet, som starter i adressen. Se også BANK kommandoen.

EKSEMPLER:

SYS 40960	Kalder og udfører maskinsprogsrutinen i adresse 40960.
SYS 8192,0	Kalder og udfører maskinsprogsrutinen i adresse 8192 og indlæser 0 til akkumulatoren.

TEMPO

Fastlægger hastigheden på afspilningen af en melodi.

TEMPO n

hvor n er en relativ varighed mellem (1 og 255).

En hel nodes aktuelle varighed afgøres ved anvendelse af nedenfor anførte formel:

hel nodes varighed = 23.66/n sekunder

Standardværdien er 8 og nodens varighed forøges med n.

EKSEMPLER:

TEMPO 16	Fastlægger tempo til 16.
TEMPO 1	Fastlægger tempo til langsomste fart
TEMPO 250	Fastlægger tempo til 250.

TRAP

Opdager og håndterer programfejl i et BASIC program under udførelse

TRAP [linienummer]

Når den er i brug, opdager TRAP de fleste fejlbetingelser (excl. DOS fejlmeddelelser men incl. STOP tasten). I tilfælde af, at en fejludførelse opdages, sættes 'fejl-flaget', og programudførelsen overføres til det linienummer, som er angivet i TRAP instruktionen. Det linienummer, i hvilket fejlen blev opdaget, kan findes ved brug af systemvariablen EL. I systemvariablen ER opbevares det angivne fejlvilkår. Strengfunktionen ERR$ (ER) afgiver den fejlmeddelelse, som svarer til fejltypen.

Til genoptagelse af programudførelsen kan RESUME instruktionen bruges. TRAP uden angivelse af linienummer afbryder fejlkontrollen. En fejl i en TRAP rutine kan ikke kontrolleres. Se også systemvariablerne ST, DS og DSS$.

EKSEMPEL:

100 TRAP 1000	Opdages en fejl, gås til linie 1000.
1000 ?ERR$(ER);EL	Udskriv fejlmeddelelsen og fejlnummeret.
1010 RESUME	Genoptag programudførelse.

TROFF

Denne instruktion afbryder fejlsøgnings mode. (TRON)

TRON

Starter fejlsøgnings mode.
TRON bruges ved fejlfinding i programmer. Instruktionen starter fejlsøgnings mode. Når programmet udføres, vises programmets linienumre i skarpe paranteser, før linien udføres. Bemærk, at tegnene [og] bliver til Æ og Å, hvis dansk tegnsæt er monteret.
Hvis De har linier indholdende flere instruktioner, vil linienummeret blive udskrevet før hver enkelt instruktion behandles.

*VERIFY

Sammenligner program i hukommelsen med eet på diskette eller bånd.

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

Denne kommando forårsager, at Commodore 128 kontrollerer programmet på bånd eller diskette med eet, som ligger i hukommelsen. Derved kan kontrolleres, at et netop lagret program virkelig er lagret, hvis man har mistanke om, at båndet er defekt eller noget ikke fungerer, som det skal. Denne kommando er også meget nyttig i forbindelse med positionering af et bånd, således at Commodore 128 vil skrive efter det sidste program på båndet. Den vil finde positionen og derefter meddele brugeren, at programmerne ikke stemmer overens. Nu er båndet imidlertid korrekt positioneret, og det næste program kan lagres uden fare for at slette det gamle.

Bruges VERIFY alene, uden tilføjelse af argumenter, vil Commodore 128 teste det næste program på båndet, uanset dets navn, mod det program, som befinder sig i hukommelsen. Såfremt VERIFY efterfølges af et programnavn i anførelsestegn, eller en strengvariabel, gennemsøges båndet for dette program for, når det er fundet, at kontrollere det med programmet i hukommelsen. VERIFY, fulgt af et navn, et komma og et nummer (1 for bånd, 8 for diskette) kontrollerer programmet på enheden med den anførte nummer. Relocate flag har her samme betydning som under LOAD kommandoen. Det kontrollerer, at programmet ligger fra den hukommelsesposition, hvorfra det blev lagret.

EKSEMPLER:

VERIFY	kontrollerer det næste program på båndet
VERIFY "HALLO"	Gennemsøger båndet for HALLO og kontrollerer det mod hukommelsen.
VERIFY "HALLO",8,1	Søger efter HALLO på diskette og kontrollerer det mod hukommelsen.

OBS! Hvis et grafisk område er allokeret efter Save (eller reallokeret), Verify eller Dverify, vil en fejlmelding fremkomme. Teknisk set er dette korrekt. Som følge af, at BASIC tekst er blevet flyttet fra sin originale (SAVEede) plads, er kædningsbytene ændret. Derfor vil Verify, som udfører byte mod byte sammenligning, vise fejl, selvom programmet er korrekt.

VOL

Fastlægger udgangsstyrken på lyd.

VOL lydstyrke

Denne instruktion indstiller volumen for SOUND og PLAY instruktionerne. VOLUME niveauet kan indstilles fra til til 15, hvor 15 er højeste styrke, og 0 er fraslået. VOL påvirker alle stemmer.

EKSEMPLER:

VOL 0 Ingen lyd.

VOL 1 Sætter volumen på laveste niveau.

VOL 15 Sætter volumen for SOUND og PLAY på højeste styrke.

*WAIT

Pause i programudførelse til en data-betingelse opfyldes.

WAIT ⟨lokation⟩,⟨mask-1⟩ [,mask-2⟩]

WAIT instruktionen forårsager, at programudførelse tilbageholdes, til en given hukommelsesadresse genkender et specificeret bitmønster eller en værdi. Med andre ord kan WAIT bruges for at standse programmet, til en eller anden ydre ting sker. Dette gøres ved at kontrollere status på bittene i input/output

registrene. Dataindivider som bruges i forbindelse med WAIT kan antage enhver værdi. De fleste programmører bruger overhovedet ikke denne instruktion. Den får programmet til at standse, til en bestemt bit i en bestemt adresse ændres. Dette anvendes ved visse I/O operationer og ellers næsten aldrig. WAIT instruktionen henter værdien i hukommelsesadressen og udfører en logisk AND operation med værdien i mask-1. Angives mask-2, bliver resultatet af første operation udelukkende OR'ed med mask-2.
Med andre ord 'bortfiltrerer' mask-1 de bits, som ikke skal testes.
Hvis bitten i mask-1 er 0, vil den tilsvarende bit i resultatet altid blive 0. Værdien i mask-2 gør, at en uopfyldt betingelse kan kontrolleres, såvel som en opfyldt. I mask-2 vil enhver bit, som bliver testet for 0, have et 1-tal i tilsvarende position. Hvis de modsvarende bits i mask-1 og mask-2 operanderne ikke er i overensstemmelse, giver OR operationen et bit resultat på 1. Giver den modsvarende bit samme resultat, er bitten 0. Det er muligt at indgive en ubetinget pause med WAIT instruktionen, i hvilket tilfælde RUN/STOP og RESTORE tasterne kan benyttes for at gå videre. WAIT kan kræve en BANK kommando, hvis den hukommelse, De ønsker at få adgang til, ikke findes i den aktuelt valgte bank.

Det første eksempel nedenunder venter med at forsætte programudførelse, til en tast på båndstationen nedtrykkes. Det andet eksempel vil vente, til SHIFT-tasten nedtrykkes og slippes igen.

EKSEMPLER:

WAIT 1, 32, 32
WAIT 211,1:WAIT 211,1,1
WAIT 36868, 144, 16

(144 og 16 er binære masker. 144 = 10010000 binært og 16 = 10000 binært.)

WIDTH

Sætter bredden på tegnede linier.

WIDTH n

Denne kommando sætter bredden på tegnede linier ved anven-

delse af BASIC's grafiske kommandoer. Der kan vælges mellem enkelt (1) og dobbelt (2) bredde.

EKSEMPLER:

WIDTH 1 Sæt enkelt bredde for grafiske kommandoer.

WIDTH 2 Sæt dobbelt bredde for tegnede linier.

WINDOW

Definerer et skærmvindue.

WINDOW øverste venstre kolonne, øverste venstre række, nederste højre kolonne, nederste højre række [,clear]

Denne kommando definerer et logisk vindue på en 40 eller 80 kolonners tekst skærm. Koordinaterne skal ligge i kolonneområderne 0 39/79 og rækkeområdet 0 - 24. Hvis clear (1) medtages, udføres en sletning af skærmbilledet (men kun indenfor rammerne af det netop beskrevne vindue.

EKSEMPLER:

WINDOW 5,5,35,20	Definerer et vindue med øverste venstre hjørne i koordinaten 5,5 og nederste højre hjørne i koordinaten 35,20.
WINDOW 10,2,33,24,1	Definerer et vindue med øverste venstre hjørne i koordinaten 10,2 og venstre nederste hjørne i koordinaten 33,24. Sletter også den del af skærmbilledet som ligger indenfor vinduets rammer. (Gøres af 1).

Obs: Hvis der angives en kolonne bredere end 39 på en 40-kolonners skærm, vil meldingen "ILLEGAL QUANTITY ERROR" fremkomme.

AFSNIT 18
Basic funktioner

BASIC FUNKTIONER

Funktionsbeskrivelsens format er:

FUNKTION (argument)

hvor argument kan være en numerisk værdi, variabel eller streng.

Hver funktionsbeskrivelse efterfølges af et EKSEMPEL. Linierne efter eksemplet er computerens svar.

*ABS

Returnerer absolut værdi

ABS (X)

Den absolutte værdi funktion returnerer den positive værdi af argumentet.

EKSEMPEL:

```
PRINT ABS (7*(-5))
35
```

*ASC

Returnerer CBM ASCII karakterkode.

ASC(X$)

Denne funktion returnerer den ASCII kode, der tilhører den første karakter i X$. I C128 mode er det ikke nødvendigt at tilføje CHR$(0) til en nulstreng. ILLEGAL QUANTITY ERROR fremkommer ikke.

EKSEMPEL:

```
X$="C128":PRINT ASC(X$)
67
```

*ATN

Returnerer vinklen, hvis tangent er X

ATN (X)

Denne funktion returnerer vinklen, hvis tangent er X, målt i radianer.

EKSEMPEL:
PRINT ATN (3)
1.24904577

*BUMP

Returnerer information om sprite kollision.

BUMP (N)

For at afgøre, hvilke sprites der er kollideret siden sidste kontrol, bruges BUMP funktionen: BUMP(1) registrerer, hvilke sprites der er kollideret med hinanden og BUMP(2) registrerer, hvilke sprites, der er kollideret med andre genstande på skærmen. COLLISION behøver ikke at være aktiv for at kunne bruge BUMP. Bit positionerne (0-7) i BUMP værdien svarer til nummeret på sprites fra 1 til 8. BUMP(n) sættes til nul (0) efter hvert kald.

Værdien som returneres med BUMP er to opløftet i potens, svarende til bitpositionen. Hvis f.eks. BUMP returnerer en værdi på 16 har sprite 4 været involveret i en kollision, da 2 opløftet til 4. potens er 16.

EKSEMPLER:

PRINT BUMP(1) indikerer at sprite 2 og 3 har kollideret.
12

PRINT BUMP(2) indikerer at sprite 5 har kollideret med en genstand på skærmen.
32

*CHR$

Returnerer ASCII karakteren for specificeret CBM ASCII kode.

CHR$(X)

Dette er modsætningen til ASC og returnerer den strengkarakter, hvis CBM ASCII kode er X. Se tabel i appendiks E med CHR$ koder.

EKSEMPLER:

PRINT CHR$(65) Skriver et A
A

PRINT CHR$(147) Sletter skærmen.

*COS

Returnerer cosinus for vinklen af X målt i radianer.

COS(X)

Denne funktion returnerer værdien på cosinus af X, hvor X er en vinkel målt i radianer.

EKSEMPEL:

PRINT COS (pi)
-1

*DEC

Returnerer decimalværdien af en heksadecimal talstreng.

DEC (heksadecimal streng)

Denne funktion returnerer decimalværdien af en heksadecimal streng.

EKSEMPEL:

PRINT DEC ("D020")
53280

*ERR$

Returnerer en streng der beskriver fejltilstand

ERR$(N)

Denne funktion returnerer en streng der beskriver en fejltilstand. Se også systemvariablerne EL og ER samt appendiks A, der indeholder en liste med BASIC fejlmeddelelser.

EKSEMPEL:

PRINT ERR$(ER)
NEXT WITHOUT FOR

*EXP

Returnerer værdien af e (2.7182813) opløftet til X potens EXP(X)

EKSEMPEL:

PRINT EXP(1)
2.7182813

*FNxx

Returnerer værdien af brugerdefineret funktion.

FNxx(x)

Denne funktion returnerer værdien fra den brugerdefinerede funktion xx frembragt med en DEF FNxx instruktion.

EKSEMPEL:

10 DEF FNAA(X)=(X-32)*5/9
20 INPUT X
30 PRINT FNAA(X)
RUN
?**40** (? er input spørgsmålet)
4.44444445

*FRE

Returnerer antallet af ubrugte bytes i hukommelsen.

FRE (X)

hvor X er bank nummeret. X=0 for ledig BASIC programlager og X=1 til undersøgelse af ledigt BASIC variabel lager.

EKSEMPLER:

PRINT FRE(0) Returnerer antal ledige bytes til BASIC programmer.
48893

PRINT FRE(1) Returnerer antal ledige bytes til BASIC variabel lager.
64256

HEX$

Returnerer heksadecimal talstreng fra decimalt tal.

HEX$ (X)

Denne funktion returnerer en firetegns streng, der indeholder den heksadecimale repræsentation af værdien X. (0⟨=X⟨65535). Den decimale modpart til denne funktion er DEC.

EKSEMPEL:

PRINT HEX$(53280)
D020
Obs: HEX$(0) ="000"

INSTR

Returnerer position af streng 1 i streng 2.

INSTR (streng 1, streng 2 [,startposition])

INSTR funktionen søger efter den første forekomst af streng 2 inden for streng 1, og returnerer positionen i strengen, hvor det søgte blev fundet. Det valgfri parameter for STARTPOSITION angiver den position i streng 1, hvor søgningen starter. STARTPOSITION skal have en værdi mellem 1 og 255. Findes ingen overensstemmelse, er STARTPOSITION større end længden af streng 1, eller hvis streng 1 er nul, returneres værdien 0. Hvis streng 2 er nul, returneres værdien 0.

EKSEMPEL:

PRINT INSTR("COMMODORE 128","128")
11

*INT

Returnerer heltalsværdien af en decimalværdi.

INT (X)

Denne funktion returnerer heltalsværdien af udtrykket. Hvis udtrykket er positivt, udelades en evt. brøk. Er udtrykket negativt, vil enhver brøkdel resultere i, at det nærmeste lavere heltal returneres.

EKSEMPLER:

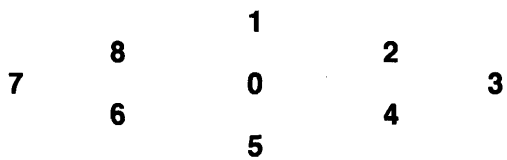
PRINT INT(3.14)
3
PRINT INT(-3.14)
-4

JOY

Returnerer position af joystick samt affyringsknaps tilstand.

JOY(N)
Hvis N = 1 fås joystick 1's position
Hvis N = 2 fås joystick 2's position

Enhver værdi på 128 eller mere betyder, at affyringsknappen også er nedtrykket. Retningen indikeres som vist nedenunder:



EKSEMPLER:

JOY(2) = 135
Joystick 2 skyder mod venstre

IF (JOY(1)AND128)=128 THEN PRINT "FYR"
Afgør om affyringsknappen er nedtrykket.

*LEFT$

Returnerer venstrestillede tegn af streng.

LEFT$ (streng,heltal)

Denne funktion returnerer en streng indeholdende strengens venstrestillede (heltal) karakterer. Værdien af heltalsargumenter skal ligge i intervallet 0 til 255. Er heltallet større end strengens længde vil hele strengen blive returneret. Bruges en heltalsværdi på nul, returneres en nulstreng (med længden 0).

EKSEMPEL:

PRINT LEFT$("COMMODORE",5)
COMMO

*LEN

Returnerer en strengs længde.

LEN (streng)

Denne funktion returnerer antallet af karakterer i et strengudtryk. Blanke og karakterer, som ikke skal udskrives, medregnes også.

EKSEMPEL:

PRINT LEN("COMMODORE128")
12

*LOG

Returnerer naturlig logaritme af X

LOG(X)

Denne funktion returnerer den naturlige log af X. Denne baserer sig på log af e (se EXP(X)). For at konvertere til 10 talslogaritme 10 divideres med LOG(10).

EKSEMPEL:

PRINT LOG(37/5)
2.00148

*MID$

Returnerer delstreng fra en større streng eller lægger en delstreng ind i en større streng (overlay).

MID$ (streng,startposition[,længde])

Denne funktion returnerer en delstreng, specificeret af længden, startende med karakteren specificeret af startpositionen. Delstrengens startposition bestemmes af den første kararkter, hvor delstrengen begynder. længden af delstrengen angives af længdeargumentet. Begge disse argumenter kan værdimæssigt ligge fra 0 til 255. Hvis startpositionens værdi er større end strengens længde, eller værdien for længden er 0, vil MID$ returnere en nul streng. Udelades længdeargumentet, vil alle karakterer til højre for startpositionen blive returneret.

EKSEMPEL:

```
PRINT MID$("COMMODORE 128",3,5)
MMODO
```

EKSEMPEL med overlay:

```
A$="123456":MID$(A$,3,2)="ABCDE":PRINT A$
12AB56
```

Obs: Overlay kan ikke bruges for udvidelse af størrelsen på en streng, derfor vil f. eks. MID$(A$,3,5) ikke være mulig i ovenstående eksempel.

*PEEK

Returnerer indholdet af en specificeret lokation i hukommelsen.

PEEK(X)

Funktionen giver indholdet af lokation X i hukommelsen, hvor X skal ligge i værdiområdet 0 til 65535, idet et resultat fra 0 til 255 returneres. Dette er modparten til POKE instruktionen. Data vil komme fra den bank, som vælges ud fra den seneste BANK kommando. Se BANK kommando.

EKSEMPEL:

```
10 BANK 15:VIC=DEC("D000")
20 FOR I=1 TO 47
30 PRINT PEEK(VIC+1),
40 NEXT
```

Dette eksempel udskriver indholdet af registrene i VIC chip'en.

*PEN

Returnerer X og Y koordinater fra lyspen.

PEN(n)

hvis n = 0 fås position af lyspennens X-koordinat
n = 1 fås position af lyspennens Y-koordinat
n = 2 fås position af X-koordinat på 80 kolonners skærm
n = 3 fås position af Y-koordinat på 80 kolonners skærm
n = 4 fås lyspennens triggerværdi

Observer, som ved sprite koordinater, at PEN værdien ikke er skalasat og bruger reelle koordinater, ikke grafisk bit map koordinater. X-koordinaten opgives som et tal liggende fra omtrent 60 til 320, medens Y positionen kan være et hvilket som helst tal fra ca. 50 til 250, liggende indenfor det omgivende kantareal. Disse er områder for de synlige skærmarealer, mens alle andre værdier ikke er synlige på skærmen.

Værdien nul for den ene eller anden position betyder, at lyspennen er udenfor skærmen og ikke siden sidste måling er blevet trykket. For at bruge PEN behøver COLLISION ikke at være aktiv. For det bedste måleresultat anbefales det at bruge hvid som baggrundsfarve. PEN værdier varierer fra system til system.

Til forskel fra 40 kolonners (VIC) skærmen, er 80 kolonners (8563) koordinaterne række og kolonnepositioner og ikke pixelkoordinater som på VIC skærmen. Både 40 og 80 kolonners skærmens koordinatværdier er tilnærmede og varierer på grund af lyspennens natur. De læste værdier er ikke gældende, før PEN(4) er korrekt.

EKSEMPLER:

10 PRINT PEN(0);PEN(1) Viser lyspennens X og Y koordinater.

10 DO UNTIL PEN(4):LOOP Sikrer rigtige læseværdier.
20 X=PEN(2)
30 Y=PEN(3)
40 REM:RESTEN AF PROGRAMMET

*(π)

Udskriver værdien af pi som 3.14159265.
(pi)

EKSEMPEL:

PRINT(π) Herved udskrives resultatet 3.14159265

POINTER

Returnerer adressen på et variabelnavn

POINTER(variabelnavn)

EKSEMPEL:

PRINT POINTER(Z)

Dette eksempel returnerer adressen på variablen Z

*POS

Returnerer den aktuelle kolonneposition i det aktuelle skærmvindue.

POS(X)

POS funktionen indikerer hvor markøren er inden for et defineret skærmvindue. X er et "dummy" argument der skal angives, men værdien ignoreres.

EKSEMPEL:

```
PRINT "0123456789" POS(1)
10
```

Dette viser den aktuelle markørposition inden for det definerede tekstvindue, i dette tilfælde 10.

POT

Returnerer paddle-potentiometrets værdi.

POT(n)

hvis n = 0 fås paddle #1 position
n = 1 fås paddle #2 position
n = 2 fås paddle #3 position
n = 3 fås paddle #4 position

Værdiområdet for POT ligger fra 0 til 255. En værdi på 256 eller mere betyder, at affyringsknappen også er nedtrykket.

EKSEMPEL:

```
10 PRINT POT(1)
20 IF POT(1) > 256 THEN PRINT "SKYD"
```

Dette eksempel viser paddle 1's værdi.

Obs: Hvis paddles ikke er tilsluttet, returneres værdien 255.

RCLR

Returnerer farven fra farvekilden.

RCLR(N)

Funktionen returnerer den farve (1 - 16), som er knyttet til kilden N (0 ⟨ N ⟨ 6).

hvor 0 = 40 kolonners baggrund
1 = bit map forgrund
2 = flerfarve 1
3 = flerfarve 2
4 = 40 kolonners kant
5 = 40 eller 80 kolonners karakterfarve
6 = 80 kolonners baggrundsfarve

Modparten til RCLR funktionen er COLOR kommandoen.

EKSEMPEL:

```
10 FOR I=0 TO 6
20 PRINT"KILDEN";I;"ER FARVE KODE";RCLR(I)
30 NEXT
```

Dette eksempel udskriver farvekoder fra alle syv farvekilder.

RDOT

Returnerer pixelmarkørens aktuelle placering eller farve.

RDOT (N)

hvor N = 0 for X koordinat i pixelmarkør
= 1 for Y koordinat i pixelmarkør
= 2 for pixelmarkørs farvekilde.

Denne funktion returnerer pixelmarkørens (PM) position, eller pixelmarkørens aktuelle farvekilde.

EKSEMPEL:

```
PRINT RDOT(0) Returnerer X position af PC
PRINT RDOT(1) Returnerer Y position af PC
PRINT RDOT(2) Returnerer PC's farvekilde. (0-3)
```

RGR

Returnerer aktuel grafisk mode.

RGR (X)

Funktionen returnerer aktuel grafisk mode. X er et dummy argument der skal angives. Modparten til RGR funktionen er GRAPHIC kommandoen. Værdien der returneres af RGR(X) svarer til følgende modes:

VÆRDI	GRAFIKMODE
0	40 kolonner (VIC) tekst
1	Standard bit map
2	Delt skærm bit map
3	Flerfarvet bit map
4	Delt skærm flerfarvet bit map
5	80 kolonner (8563) tekst

EKSEMPEL:

```
PRINT RGR(0)
1
```

Viser det aktuelle grafikmode; i dette tilfælde standard bit map mode.

*RIGHT$

Returnerer en substreng med en strengs højrestillede tegn.

RIGHR$ (⟨streng⟩,⟨længde⟩)

Denne funktion returnerer en delstreng taget fra strengargumentets højre side. Længden af delstrengen angives af længdeargumentet, som kan være ethvert heltal fra 0 til 255.
Er det numeriske udtryks værdi nul, returneres en nulstreng.
Er det numeriske udtryks værdi større end strengens længde, returneres hele strengen. Se også LEFT$ og MID$ funktionerne

EKSEMPEL:

```
PRINT RIGHT$("BASEBALL",5)
EBALL
```

*RND

Returnerer tilfældigt (random) tal.

RND (X)

Denne funktion returnerer et tilfældigt tal X som 0 ⟨= X ⟨1. Funktionen er nyttig i forbindelse med spil for at simulere terningkast og andre chancebetonede elementer, og kan endvidere bruges i visse statistiske opgaver.

Hvis X = 0	RND returnerer et tilfældigt tal, baseret på hardwares tidsenhed.
Hvis X ⟩ 1	RND skaber et reproducerbart pseudo-tilfældigt tal, baseret på en efterfølgende seeded værdi.
Hvis X ⟨ 0	Danner et tilfældigt tal, der anvendes som en base der kaldes en seedning.

Til simulering af terningkast bruges formlen INT(RND(1)*6+1). Først ganges det tilfældige tal med 6, hvilket udvider området til 0-6 (i virkeligheden mindre end seks). Derefter tillægges 1, hvilket nu giver et område på mindre end syv.
INT funktionen afskærer alle evt. decimaler og afgiver resultatet som et tal fra 1 til 6.

EKSEMPEL:

PRINT RND(0)	Viser et tilfældigt tal
.507824123	mellem 0 og 1
PRINT INT(RND(1)*100+1)	Viser et tilfældigt positivt
89	tal mindre end 100.

RSPCOLOR

Returnerer sprite flerfarve værdi.

RSPCOLOR (register)

Når:

X = 1 RSPCOLOR returnerer sprite flerfarve 1.
X = 2 RSPCOLOR returnerer sprite flerfarve 2.

Den farveværdi der returneres har værdier mellem 1 og 16. Modparten til RSPCOLOR funktionen er SPRCOLOR instruktionen.

Se også SPRCOLOR instruktionen.

EKSEMPEL:

```
10 SPRITE 1,1,2,0,1,1
20 SPRCOLOR 5,7
30 PRINT"SPRITE FLERFARVE 1 ER";RSPCOLOR(1)
40 PRINT"SPRITE FLERFARVE 2 ER";RSPCOLOR(2)
RUN

SPRITE FLERFARVE 1 ER 5
SPRITE FLERFARVE 2 ER 7
```

I dette eksempel tænder linie 10 sprite 1, farver den hvid, udvider den i både X og Y retningen, og viser den i flerfarve mode. Linie 20 vælger sprite flerfarver 1 og 2. Linie 30 og 40 udskriver RSPCOLOR værdier for flerfarve 1 og 2.

RSPPOS

Kontrollerer en sprites hastighed og position.

RSPPOS (sprite nummer,x)

hvor "spritenummer" er den sprite, der skal kontrolleres og "x" angiver x og y koordinaterne eller spritens hastighed.
Hvor positionen svarer til:

0 RSPPOS returnerer en specificeret sprites X position.
1 RSPPOS returnerer en specificeret sprites Y position.
2 RSPPOS returnerer hastigheden (0-15) for den specificerede sprite.

EKSEMPEL:

```
10 SPRITE 1,1,2
20 MOVSPR 1,43#13
30 PRINT RSPPOS(1,0);RSPPOS(1,1);RSPPOS(1,2)
```

Dette eksempel returnerer den aktuelle sprites X og Y koordinater og hastigheden (13).

RSPRITE

Returnerer sprite attributter.

RSPRITE (sprite nummer,attributter)

RSPRITE returnerer de attributter der blev specificeret i SPRITE

kommandoen. Sprite nummer specificerer den sprite der kontrolleres og karakteristikken, der angiver spritens egenskaber er som følger:

Karakteristik:	**RSPRITES returnerer disse værdier:**	
0	virksom(1) / uvirksom (0)	
1	Sprite farve (1-16)	
2	Sprite vises foran (0) eller bag (1) genstande på skærmen	
3	Forøgelse i X retning	ja = 1, nej = 0
4	Forøgelse i Y retning	ja = 1, nej = 0
5	Flerfarve	ja = 1, nej = 0

EKSEMPEL:

```
10 FOR I=0 TO 5
20 PRINT RSPRITE(1,I)
30 NEXT
```

Dette eksempel udskriver alle karakteristika for sprite 1.

RWINDOW

Returnerer oplysninger om størrelsen på det aktuelle skærmvindue.

RWINDOW (n)

Hvor n svarer til:

0 antal linier i aktuelt vindue
1 antal kolonner i aktuelt vindue
2 returnerer værdien 40 eller 80, afhængig af hvilket skærmformat der anvendes.

Modparten til RWINDOW funktionen er WINDOW kommandoen.

EKSEMPEL:

```
10 WINDOW 1,1,10,10
20 PRINT RWINDOW(0);RWINDOW(1);RWINDOW(2)
RUN
 9  9  40
```

Dette eksempel returnerer antallet af linier (9) og kolonner (9) i det aktuelle vindue. Dette eksempel forudsætter at vinduet vises i 40 kolonners format.

*SGN

Returnerer fortegn for argument X.

SGN (X)

Denne funktion viser X's fortegn (positivt, negativt eller nul). Resultatet er 1 hvis X > 0, 0 hvis X = 0 og -1 hvis X < 0.

EKSEMPEL:

```
PRINT SGN(4.5);SGN(0);SGN(-2.3)
 1  0 -1
```

*SIN

Returnerer sinus af argument.

SIN (X)

Dette er den trigonometriske sinusfunktion. Resultatet er sinus af X. X måles i radianer.

EKSEMPEL:

```
PRINT SIN(pi/3)
 .866025404
```

*SPC

Overspringer mellemrum (space) på skærmen.

SPC (X)

SPC funktionen bruges i PRINT og PRINT# kommandoer til kontrol af dataformattering, enten som output til skærmen eller output til en logisk fil. Det tal, som angives af X, afgør antallet af karakterer der skal udfyldes med mellemrum (space) over skærmen eller i en fil. For skærm og båndfiler ligger argumentets værdiområde fra 0 til 255 og for diskettefiler er maximum 254. Ved printerfiler vil en automatisk vognretur og linieskift blive udført af printeren, hvis et mellemrum (space) skrives i sidste karakterposition i en linie. Ingen mellemrum skrives på den følgende linie.

EKSEMPEL:

```
PRINT"COMMODORE";SPC(3);128"
COMMODORE 128
```

*SQR

Returnerer kvadratroden af argumentet.

SQR (X)

Denne funktion giver som resultat kvadratroden af X, hvor X er et positivt tal eller 0. Argumentets værdi må ikke være negativ, i så fald vil BASIC fejlmeldingen ?ILLEGAL QUANTITY ERROR fremkomme.

EKSEMPEL:

PRINT SQR(25)
5

*STR$

Returnerer strengrepræsentation af et tal STR$ (X) Funktionen returnerer strengrepræsentationen af X's numeriske værdi. Når STR$ værdien oversættes til hver variabel, som er repræsenteret i det numeriske argument, har hvert vist tal efterfulgt af et mellemrum og, hvis det er positivt. Et foranstillet mellemrum. Negative tal foranstilles et "-"-tegn.
Modparten til STR$ funktionen er VAL funktionen.

EKSEMPEL:

PRINT STR$(123.45)
123,45

PRINT STR$(-89.03)
-89.03

PRINT STR$(1E20)
1E+20

*TAB

Flytter markøren til tab-position i denne instruktion.

TAB (X)

Denne funktionen flytter, hvis det er muligt, markøren til en relativ position på tekstskærmen, angivet af argumentet X, idet der begyndes med den aktuelle liniens yderste venstre position. Argumentets værdi kan variere fra 0 til 255. Hvis den aktuelle position allerede er efter position X, vil TAB kommandoen blive

ignoreret. TAB funktionen kan kun anvendes sammen med PRINT instruktionen, da den har forskellig effekt sammen med PRINT# til en logisk fil, afhængende af den anvendte enhed.

EKSEMPEL:

```
10 PRINT"COMMODORE"TAB(25)"128"
COMMODORE                 128
```

*TAN

Returnerer tangenten til argumentet.

TAN (X)

Funktionen returnerer tangenten til X, hvor X er en vinkel i radianer.

EKSEMPEL:

```
PRINT TAN(.785398163)
1
```

*USR

Kalder brugerdefineret subfunktion.

USR(X)

Når denne funktion bruges, springer programmet til et maskinsprogsprogram, hvis startadresse ligger i hukommelsesadresserne 4633 ($1219) og 4634 ($121A),(og 785 ($0311) og 786 ($0312) i C64 mode).Parameteren X overgives til maskinsprogsprogrammet i akkumulatoren med flydende kommaplacering. En værdi sendes tilbage til BASIC programmet via kaldevariablen. De skal omdirigere værdien til en variabel i Deres program for at kunne modtage værdien fra akkumulatoren med flydende kommaplacering. Hvis denne variabel ikke specificeres, vil det resultere i en ILLEGAL QUANTITY ERROR. Dette tillader brugeren at udveksle en variabel mellem maskinkode og BASIC.

EKSEMPEL: (kun C128)

```
10 POKE 4633,0
20 POKE 4634,192
30 A=USR(X)
40 PRINT A
```

NB: Standardværdi 128 er bank 15.
Anbring startadresserne ($C000=49152:$00=0:$C0=192) af maskinsprogsrutinen i adresse 4633 og 4634. Linie 30 lagrer den returnerede værdi fra akkumulatoren.

*VAL

Returnerer en talstrengs numeriske værdi.

VAL (X$)

Denne funktion konverterer strengen X$ til et tal og virker faktisk modsat STR$. Strengen undersøges fra venstre mod højre, så længe der er karakterer i et genkendeligt talformat. Hvis Commodore 128 finder ugyldige karakterer, konverteres kun den del af strengen, der ligger foran dette punkt. Er der ingen numeriske karakterer i strengen, vil VAL returnere et 0.

EKSEMPEL:

```
10 A$="120"
20 B$="365"
30 PRINT VAL (A$+B$)
RUN
485
```

XOR

Returnerer eksklusiv OR

XOR(n1,n2)

Denne funktion giver eksklusive OR af argumentværdierne n1 og n2.

x = XOR (n1,n2)

hvor n1, n2, er værdier uden fortegn (0-65535).

EKSEMPEL:

```
PRINT XOR(128,64)
192
```

NB: n1 og n2 behøver ikke at være heltal.

AFSNIT 19
VARIABLER OG OPERATORER

VARIABLER

Commodore 128 bruger tre variabeltyper i BASIC. Disse er: normale numeriske, heltals numeriske og strengvariabler (alfanumeriske).

Normale NUMERISKE VARIABLER, også kaldet variabler med flydende kommaplacering, kan have en hvilken som helst eksponentiel værdi fra -10 til +10, med op til ni cifres nøjagtighed. Hvis et tal bliver større end, at det kan skrives med 9 cifre, vises det af computeren i videnskabelig notations form med tallet normaliseret til eet ciffer med otte decimaler, efterfulgt af bogstavet E samt potensen 10, hvormed tallet multipliceres. For eksempel udtrykkes tallet '12345678901' som: 1.23456789E+10.

HELTALS VARIABLER kan bruges, når tallet er fra +32767 til -32768 og uden decimaler. En heltalsvariabel er et tal som 5, 10 eller -100.
Heltal fylder mindre end variabler med flydende komma, især når de bruges i et område.

STRENGVARIABLER benyttes til karakterdata, som kan indeholde tal, bogstaver og andre tegn, som Commodore 128 kan fremstille. Et eksempel på en strengvariabel kan være "Commodore 128".

VARIABELNAVNE kan bestå af et enkelt bogstav, et bogstav efterfulgt af et tal eller to bogstaver. Variabelnavne kan være længere end to tegn, men kun de to første bruges. Et heltal angives ved at sætte procent (%) tegn efter variabelnavnet. Strengvariabler navne efterfølges af et dollartegn ($).

EKSEMPLER:
Numeriske variabelnavne:A, A5, BZ
Heltal variabelnavne:A%, A5%, BZ%
Streng variabelnavne:A$, A5$, BZ$

ARRAYS (områder) er lister indeholdende variabler med samme navn, som bruger eet eller flere tal for at angive et element i området.

Områderne defineres ved anvendelse af DIM instruktionen og kan være flydende komma-, heltals- eller strengvariabelområder. Navnet på områdevariablen følges af et sæt paranteser (), der indeholder nummeret på variablen i listen.

EKSEMPEL:
A(7), BZ%(11), A$(87)

Tabelområder kan have mere end een dimension. En todimensional tabel kan betragtes som indeholdende rækker og kolonner, hvor det første tal definerer rækkenummeret og det andet tal definerer kolonnen. (På samme måde, som når man specificerer et bestemt punkt på et kort).

EKSEMPEL:
A(7,2), BZ%(2,3,4), Z$(3,2)

RESERVEREDE VARIABELNAVNE er navne, som er reserveret til Commodore 128's brug og ikke må anvendes til andre formål. Det er variablerne DS, DS$, ER, ERR$, EL, ST, TI og TI$. Nøgleord som TO og IF eller andre navne, der indeholder nøgleord som RUN, NEW eller LOAD, kan heller ikke anvendes.

ST er en statusvariabel for input og output (bortset fra normale skærm/tastatur operationer). Værdien af ST afhænger af den sidste I/O operations resultater. Normalt var operationen vellykket, hvis værdien af ST = 0.

TI og TI$ er variabler relaterede til det realtidsur, der er indbygget i Commodore 128. Systemuret opdateres tres gange i sekundet.
Når der tændes for Commodore 128, startes der ved 0 og kan derefter kun ændres ved at udskifte værdien i TI$. Variablen TI giver urets nuværende værdi i tresindstyvendele sekunder. TI$ er en streng, som læser realtidsurets værdi som et 24 timers ur. De to første tegn af TI$ indeholder timen, tredie og fjerde tegn er minutterne, mens femte og sjette tegn er sekunderne. Denne variabel kan sættes til en hvilket som helst værdi (så længe alle karakterer er tal) og opdateres automatisk som et 24 timers ur.

EKSEMPEL:
TI$ = "101530" indstiller uret til 10:15 og 30 sekunder (AM).

Urets indstillede værdi går tabt, når der slukkes for Commodore 128.
Når der tændes for Commodore 128 startes ved 0, og der gås atter til 0, når urets værdi overstiger 235959 (23 timer, 59 minutter, 59 sekunder).

DS er en variabel, som aflæser diskettedrevets kommandokanal og returnerer drevets aktuelle status. For at få denne information udskrevet bruges PRINT DS$. Disse statusvariabler bruges efter disketteoperationer som DLOAD og DSAVE for at finde ud af grunden, når den røde fejllampe på diskettedrevet blinker.

ER, EL og ERR$ er variabler til brug i fejlsøgningsrutiner. De egner sig normalt kun til anvendelse i et program. ER viser den sidst fundne fejl efter kørsel af programmet. EL viser den linie, hvori fejlen optrådte. ERR$ er en funktion, som tillader programmet at udskrive een af BASIC fejlmeddelelserne. PRINT ERR$(ER) udskriver den rigtige fejlmelding.

OPERATORER

BASIC OPERATORER omfatter ARITMETISKE, RELATIONELLE og LOGISKE OPERATORER. De ARITMETISKE operatorer omfatter følgende tegn:

+ addition

- subtraktion

*** multiplikation**

/ division

↑ potensopløftning (eksponentiering)

I en linie, som indeholder mere end een operator, findes en fast rækkefølge, hvori operationerne udføres. Hvis flere operatorer bruges i sammenhæng, prioriteres de af computeren som følger: først foretages potensopløftning, derefter multiplikation og division og til sidst addition og subtraktion. Hvis to operatorer er prioriteret ens, udføres beregningerne i rækkefølge fra venstre mod højre. Hvis operationsrækkefølgen skal være en anden, tillader Commodore 128 BASIC, at en kalkulation prioriteres højere ved at sætte den i paranteser. Operationer angivet i paranteser vil blive udført før de øvrige operationer. Man må sikre sig,

at beregningsformlerne indeholder lige mange venstre- og højre-paranteser, da der i modsat fald vil blive afgivet en SYNTAX ERROR fejlmelding, når programmet køres.
Der findes også operatorer for ligheder og uligheder. Disse operatorer kaldes RELATIONELLE operatorer. Aritmetiske operatorer vil altid være højere prioriterede end relationelle.

=	**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 forskellig fra fra**

Endelig findes tre LOGISKE operatorer, som alle har lavere prioritet end såvel aritmetiske som relationelle operatorer.

AND
OR
NOT

Disse anvendes oftest til at forbinde flere formler i IF ... THEN instruktioner. Når de bruges med aritmetiske operatorer, udføres de sidst (d.v.s. efter + og -). Er relationen i udtrykket sandt, bliver resultatet en heltalsværdi på -1. Er den falsk gives værdien 0.

EKSEMPLER:

IF A = B AND C = D THEN 100	kræver at både A=B og C=D er sande
IF A = B OR C = D THEN 100	tillader at enten A=B eller C=D er sand.
A = 5:B = 4:PRINT A = B	viser værdien 0
A = 5:B = 2:PRINT A〉3	viser værdien -1
PRINT 123 AND 15:PRINT 5	viser 11 og 7 OR 7

AFSNIT 20
Reserverede ord og symboler

RESERVEREDE ORD OG SYMBOLER

Dette afsnit indeholder ord og symboler der udgør BASIC 7.0 sproget.
Disse ord og symboler kan ikke anvendes i et program som andet end dele af BASIC sproget. Den eneste undtagelse er, at de kan anvendes inden for anførelsestegn i en PRINT sætning.

ABS	AND	APPEND	ASC	ATN
AUTO	BACKUP	BANK	BEGIN	BEND
BLOAD	BOOT	BOX	BSAVE	BUMP
CATALOG	CHAR	CHR$	CIRCLE	CLOSE
CLR	CBM	COLLECT	COLLISION	COLOR
CONCAT	CONT	COPY	COS	DATA
DCLEAR	DCLOSE	DEC	DEF FN	DELETE
DIM	DIRECTORY	DLOAD	DO	DOPEN
DRAW	DS	DSAVE	DS$	DVERIFY
EL	ELSE	END	ENVELOPE	ER
ERR$	EXIT	EXP	FAST	FETCH
FN	FOR	FRE	GET	GETKEY
GET#	GO64	GOSUB	GOTO	GO TO
GRAPHIC	GSHAPE	HEADER	HELP	HEX$
IF	INPUT	INPUT#	INSTR	INT
JOY	KEY	LEFT$	LEN	LET
LIST	LOAD	LOCATE	LOG	LOOP
MID$	MONITOR	MOVSPR	NEW	NEXT
NOT	ON	OPEN	OR	PAINT
PEEK	PEN	PLAY	POINTER	POKE
POS	POT	PRINT	PRINT#	PUDEF
RCLR	RDOT	READ	RECORD	REM
RENAME	RENUMBER	RESTORE	RESUME	RETURN
RGR	RIGHT$	RND	RREG	RSPCOLOR
RSPPOS	RSPRITE	RUN	RWINDOW	SAVE
SCALE	SCNCLR	SCRATCH	SGN	SIN
SLEEP	SLOW	SOUND	SPC	SPRCOLOR
SPRDEF	SPRITE	SPRSAV	SQR	SSHAPE
ST	STASH	STEP	STOP	STR$
SWAP	SYS	TAB	TAN	TEMPO
THEN	TI	TI$	TO	TRAP
TRON	TROFF	UNTIL	USING	USR
VAL	VERIFY	VOL	WAIT	WHILE
WIDTH	WINDOW	XOR		

RESERVEREDE SYSTEM SYMBOLER

Følgende karakterer er reserveret som system symboler:

	SYMBOL	ANVENDELSER
+	Plus tegn	Aritmetisk addition; sprite bevægelse; erklærer decimaltal i maskinsprogs monitor.
-	Minus tegn,	Aritmetisk subtraktion, negativt tal; unary minus; relativ sprite bevægelse.
*	Stjerne	Aritmetisk multiplikation
/	Skråstreg	Aritmetisk division
↑	Pil opad	Aritmetisk eksponentiering
	Mellemrum	Adskiller nøgleord og variabelnavne
=	Lighedstegn	Værdiangivelse; test af relation
<	Mindre end	Test af relation
>	Større end	Test af relation
,	Komma	Formatter output i variable lister; kommando/erklæring funktions-parametre
.	Punktum	Decimalpunkt i flydende decimal-konstanter
;	Semikolon	Formatter output i variable lister;
:	Kolon	Deler flere BASIC kommandoer på en programlinie.
" "	Anførelsestegn	Omslutter streng konstant.
?	Spørgsmålstegn	Forkortelse for nøgleordet PRINT
(	Venstre parantes	Udtryksevaluering og funktioner
)	Højre parantes	Udtryksevaluering og funktioner
%	Procent	Erklærer et variabelnavn som heltal; Erklærer binært tal i maskinsprogs-monitor.

#	Nummertegn	Indleder det logiske filnummer i input/output kommandoer.
$	Dollartegn	Erklærer et variabelnavn som en streng og erklærer hexadecimale tal i maskinsprogsmonitor.
&	Et tegn	Erklærer oktalt tal i maskinsprogs monitor
π	Pi	Erklærer den numeriske konstant ca. 3.14159265

TILLÆG

BASIC FEJLMEDDELELSER

De følgende fejlmeddelelser vises af BASIC. Fejlmeddelelser kan også vises ved brug af ERR$() funktionen. Nedennævnte fejlnummer henviser kun til tallet tilknyttet fejlen, ved brug af ERR$() funktionen.

FEJL NR.	FEJLMEDDELELSE	BESKRIVELSE
1	TOO MANY FILES	BEGRÆNSNINGEN PÅ 10 SAMTIDIG ÅBNE FILER ER OVERSKREDET
2	FILE OPEN	DER ER FORETAGET FORSØG PÅ AT ÅBNE EN ALLEREDE ÅBNET FIL
3	FILE NOT OPEN	DET FILNUMMER SOM SPECIFICERES I EN I/O INSTRUKTION SKAL ÅBNES FØR BRUG.
4	FILE NOT FOUND	ENTEN FINDES INGEN FIL MED DET ANGIVNE NAVN (DISK) ELLER DER ER FUNDET ET END=OF=TAPE MÆRKE. (KASSETTE)
5	DEVICE NOT PRESENT	DEN ANGIVNE I/O ENHED ER IKKE TILSLUTTET ELLER BUFFER ER IKKE ALLOCERET (KASSETTE). UNDERSØG OM ENHEDEN ER FORBUNDET OG TÆNDT
6	NOT INPUT FILE	DER ER GJORT FORSØG PÅ GET ELLER INPUT MED DATA FRA EN FIL, SOM ER SPECIFICERET SOM UDELUKKENDE OUTPUT.
7	NOT OUTPUT FILE	DER ER GJORT FORSØG PÅ AT SENDE DATA TIL EN FIL, SOM ER SPECIFICERET SOM INPUTFIL.
8	MISSING FILE NAME	FILNAVNET MANGLER I KOMMANDO
9	ILLEGAL DEVICE NUMBER	FORSØG PÅ AT BRUGE EN ENHED TIL NOGET FORKERT (SAVE TIL SKÆRM ELLER LIGNENDE)

FEJLNR.	FEJLMEDDELELSE	BESKRIVELSE
10	NEXT WITHOUT FOR	ENTEN ER LOOP FORKERT OPBYGGET ELLER DER ER ET VARIABELNAVN I EN NEXT-INSTRUKTION, SOM IKKE PASSER TIL NAVNET I FOR.
11	SYNTAX	EN INSTRUKTION, SOM BASIC IKKE KAN GENKENDE. DETTE KAN SKYLDES MANGLENDE ELLER EKSTRA PARANTESER, FORKERT STAVEMÅDE, ETC.
12	RETURN WITHOUT GOSUB	EN RETURN-INSTRUKTION ER OPDAGET, HVOR INGEN GOSUB-INSTRUKTION VAR AKTIV.
13	OUT OF DATA	EN READ-INSTRUKTION ER OPDAGET, HVOR ALLE DATA ER OPBRUGT.
14	ILLEGAL QUANTITY	ET TAL ANVENDT SOM ARGUMENT I EN FUNKTION ELLER INSTRUKTION LIGGER UDENFOR DET TILLADTE OMRÅDE.
15	OVERFLOW	RESULTATET AF EN BEREGNING 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 GOSUB INSTRUKSER I BRUG.
17	UNDEF'D STATEMENT	HENVISNING TIL ET LINIENUMMER, IKKE FINDES I PROGRAMMET.
18	BAD SUBSCRIPT	PROGRAMMET FORSØGTE AT HENVISE TIL ET ELEMENT I ET OMRÅDE, SOM LIGGER UDEN FOR DET, SOM ER SPECIFICERET I DIM-INSTRUKTIONEN.
19	REDIM'D ARRAY	ET OMRÅDE KAN KUN DIMENSIONERES EEN GANG.
20	DIVISION BY ZERO	DIVISION MED 0 ER IKKE TILLADT
21	ILLEGAL DIRECT	INPUT OG GET ELLER INPUT# GET# INSTRUKTIONER ER KUN TILLADT I ET PROGRAM.

FEJLNR.	FEJLMEDDELELSE	BESKRIVELSE
22	TYPE MISMATCH	FOREKOMMER HVIS ET TAL ER BRUGT I STEDET FOR EN STRENG ELLER OMVENDT.
23	STRING TOO LONG	EN STRENG KAN HØJST INDEHOLDE 255 TEGN.
24	FILE DATA	FEJLRAMTE DATA INDLÆST FRA EN BÅND- ELLER DISKFIL.
25	FORMULA TOO COMPLEX	COMPUTEREN KUNNE IKKE FORSTÅ UDTRYKKET. GØR DET ENKLERE (DEL DET I TO DELE ELLER BRUG FÆRRE PARANTESER).
26	CAN'T CONTINUE	CONT-KOMMANDOEN VIRKER IKKE, HVIS PROGRAMMET IKKE KØRTES, HVIS DER VAR EN FEJL ELLER HVIS EN LINIE ER ÆNDRET.
27	UNDEFINED FUNCTION	DER BLEV REFERERET TIL EN BRUGEREEFINERET FUNKTION DER IKKE VAR DEFINERET.
28	VERIFY	PROGRAMMET PÅ BÅND ELLER DISK SVARER IKKE TIL PROGRAMMET I HUKOMMELSEN.
29	LOAD	PROBLEMER MED INDLÆSNINGEN. PRØV IGEN.
30	BREAK	DER ER TRYKKET PÅ STOPTASTEN UNDER PROGRAMUDFØRELSEN.
31	CAN'T RESUME	EN RESUME-INSTRUKTION OPDAGET, UDEN AT DER FINDES EN VIRKSOM TRAP INSTRUKTION.
32	LOOP NOT FOUND	PROGRAMMET HAR MØDT EN DO-INSTRUKTION UDEN AT FINDE DEN TILHØRENDE LOOP-INSTRUKTION.
33	LOOP WITHOUT DO	PROGRAMMET INDEHOLDER EN LOOP-INSTRUKTION UDEN EN AKTIV DO-INSTRUKTION.
34	DIRECT MODE ONLY	KOMMANDOEN ER KUN TILLADT I DIREKTE TILSTAND, IKKE I ET PROGRAM.

FEJLNR.	FEJLMEDDELELSE	BESKRIVELSE
35	NO GRAPHIC AREA	EN KOMMANDO (DRAW, BOX, ETC.) TIL FREMSTILLING AF GRAFIK, ER MØDT FØR UDFØRELSE AF GRAPHIC-KOMMANDOEN.
36	BAD DISK	ENTEN ER DISKETTEN DEFEKT ELLER FORSØG PÅ AT FORMATTERE EN DISKETTE SLOG FEJL, FORDI DEN HURTIGE FORMATTERINGSMETODE (UDEN ID) BLEV FORSØGT PÅ EN UFORMATTERET DISKETTE.
37	BEND NOT FOUND	PROGRAMMET OPDAGEDE EN "IF..THEN BEGIN" eller "IF....THEN ELSE BEGIN" OPBYGNING OG KUNNE IKKE FINDE ET BEND - NØGLEORD DER SSER TIL BEGIN.
38	LINE # TOO LARGE	DER ER OPSTÅET EN FEJL UNDER NUMMERERINGEN AF ET BASIC PROGRAM. DE GIVNE PARAMETRE RESULTERER AT ET LINIENUMMER) 999 DANNES. DERFOR BLEV RENUMMERERINGEN IKKE UDFØRT.
39	UNRESOLVED REFERENCES	EN FEJL ER OPDAGET VED RENUMMERERING AF ET BASIC PROGRAM. ET LINIENUMMER DER HENVISES TIL MED EN KOMMANDO (F. EKS. GOTO 999) EKSISTERER IKKE. DERFOR BLEV RENUMMERERINGEN IKKE UDFØRT.
40	UNIMPLEMENTED COMMAND	EN KOMMANDO SOM IKKE UNDERSTØTTES AF BASIC 7.0 ER MØDT.
41	FILE READ	EN FEJL ER OPSTÅET UNDER INDLÆSNING ELLER LÆSNING AF ET PROGRAM ELLER EN FIL FRA DISKEN (F. EKS HVIS LÅGEN TIL DISKETTESTATONEN HAR VÆRET ÅBNET UNDER INDLÆSNINGEN.

APPENDIX B
DOS FEJLMEDDELELSER

De følgende DOS fejlmeddelelser returneres gennem DS og DS$ variable. DS variablen indeholder kun fejlnummeret og DS$ variablen indeholder fejlnummeret, fejlmeddelelsen og det tilhørende spor og sektor nummer.
NB.: Fejlmeddelelser med numre mindre end 20 skal ignoreres med undtagelse af 01, der giver information om det antal filer, der er slettet med SCRATCH kommandoen.

FEJLNR	BESKRIVELSE
20	READ ERROR (block header ikke fundet). Diskcontrolleren er ikke i stand til at lokalisere block headeren på den forlangte datablok. Skyldes et ulovligt sektornummer, eller at header er ødelagt.
21	READ ERROR (ingen sync. karakter). Controlleren er ikke i stand til at finde et synkroniseringsmærke på det ønskede spor. Skyldes fejljustering af læse/skrivehovedet, manglende diskette, uformatteret eller ukorrekt isat diskette. Kan også forekomme ved hardwarefejl.
22	READ ERROR (manglende datablok). Controlleren er blevet bedt om at læse eller verificere en datablok, der ikke er korrekt skrevet. Denne meddelelse fremkommer i forbindelse med BLOCK kommandoerne og indikerer et ulovligt spor/sektorforlangende.
23	READ ERROR (kontrolsumfejl i datablok). Denne fejlmeddelelse angiver, at der er fejl i en eller flere databytes. Data er indlæst i DOS hukommelsen, men kontrolsummen over data er forkert. Kan også betyde problemer med jordforbindelsen.
24	READ ERROR (byte fortolkningsfejl). Data eller header er blevet indlæst i DOS hukommelsen, men en hardwarefejl er fremkaldt på grund af et ugyldigt bitmønster i data-byten. Kan også betyde problemer med jordforbindelsen.
25	WRITE ERROR (skrive/verificerings fejl). Denne meddelelse fremkommer, hvis controlleren opdager en uoverensstemmelse mellem de skrevne data og data i DOS hukommelsen.

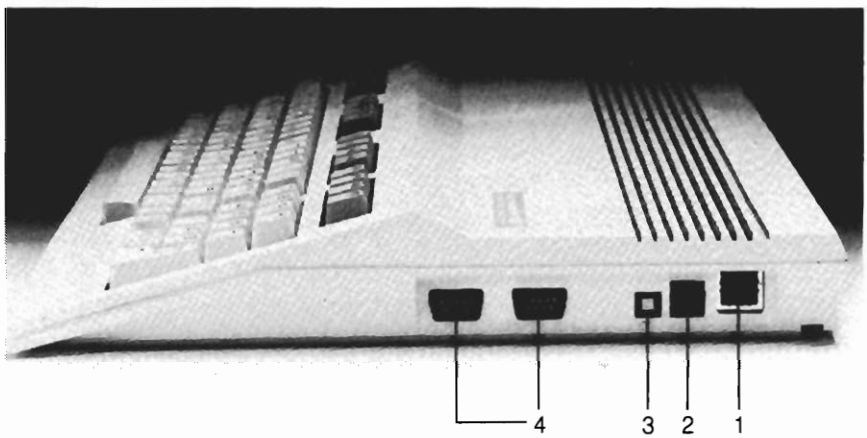
FEJLNR	BESKRIVELSE
26	WRITE PROTECT ON (diskette skrivebeskyttet). Denne meddelelse fremkommer, når controlleren er bedt om at skrive en datablok mens skrivebeskyttelseskontakten er sluttet. Dette sker typisk, når der benyttes en diskette, hvor skriveslidsen er lukket af en mærkat
27	READ ERROR (kontrolsumfejl) Denne meddelelse vises, hvis der er blevet opdaget en kontrolsumfejl i headeren til en forlangt datablok. Blokken er ikke blevet indlæst i DOS hukummelsen.
28	WRITE ERROR Denne meddelelse fremkommer, hvis en datablok er for lang og overskriver synkroniseringsmærket i næste header. Blokken indlæses ikke i DOS hukommelsen.
29	DISK ID MISMATCH Denne meddelelse fremkommer, når controlleren er blevet bedt om at gå ind på en diskette, der ikke er blevet initialiseret. Kan også fremkomme, hvis en diskette har en forkert header.
30	SYNTAX ERROR (generel syntaksfejl). DOS kan ikke oversætte den kommando, der er sendt til kommandokanalen. Dette skyldes typisk et ulovligt antal filnavne, eller at mønstre bruges forkert. F. eks. hvis filnavne er placeret til venstre for COPY kommandoen.
31	SYNTAX ERROR (ulovlig kommando). DOS kender ikke den afgivne kommando. Kommandoen skal begynde i første position.
32	SYNTAX ERROR (for lang linie). Den afgivne kommando er længere end 58 tegn. Brug forkortede diskkommandoer.
33	SYNTAX ERROR (ulovligt filnavn). Forkert mønstersammenligning ved anvendelse af OPEN eller SAVE kommando, forkert stavning af filnavn.
34	SYNTAX ERROR (ingen fil angivet). Filnavnet er udeladt i en kommando, eller DOS genkender det ikke som et sådant. Typisk mangler et kolon (:) i kommandoen.
39	SYNTAX ERROR (ulovlig kommando). Denne meddelelse er resultatet hvis den kommando, som er sendt til kommandokanalen (sekundær adresse 15), ikke kendes af DOS.
50	RECORD NOT PRESENT - Resultat efter forsøg på at læse på disketten forbi sidste post ved brug af INPUT# eller GET# kommandoen. Meddelelsen vil også fremkomme, hvis der positioneres til en post efter filslut i en relativ fil. Hvis hensigten er at tilføje en ny post (med en PRINT# kommando), kan meddelelsen ignoreres. Uden først at ompositionere bør INPUT eller GET ikke forsøges efter denne fejlmeddelelse.

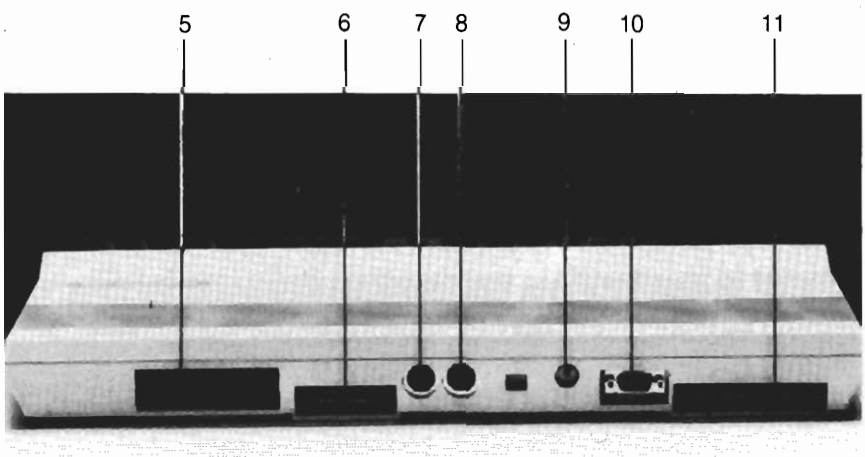
FEJLNR	BESKRIVELSE
51	OVERFLOW IN RECORD - PRINT# instruktion overskrider grænsen for en post. Oplysning forkortes. Da vognretur, der sendes som slut på posten, tælles med i poststørrelsen, vil denne meddelelse fremkomme, hvis det samlede antal tegn (incl. sidste vognretur) overskrider den definerede 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, hvis en skrive-fil der ikke er blevet lukket, prøves forsøgt åbnet for indlæs-ning.
61	FILE NOT OPEN - Denne meddelelse fremkommer, hvis man prøver at gå ind til en fil, som ikke er blevet åbnet i DOS. I dette til-fælde sker det somme tider, at der ikke vises nogen meddelelse. Anmodningen ignoreres simpelthen.
62	FILE NOT FOUND - Den forlangte fil findes ikke på disketten i det angivne drev.
63	FILE EXISTS - Navnet på den fil, der prøves gemt, findes alle-rede på disketten.
64	FILE TYPE MISMATCH - Den anmodede filtilgang er ikke mulig ved anvendelse af denne type fil. Læs atter kapitlet, der beskriver denne filtype.
65	NO BLOCKS - Denne meddelelse kommer i forbindelse med Block Allocation kommandoen og indikerer, at den blok, der skal allo-keres, er allokeret tidligere. Det spor og sektornummer der meldes tilbage, er det næstfølgende højere ledige spor og sektornummer. Hvis det spornr. der meldes tilbage er nul (0), er alle blokke med højere numre i brug. Hvis disketten endnu ikke er fuld, prøv da med et lavere spor og sektornummer.
66	ILLEGAL TRACK AND SECTOR - DOS har forsøgt at gå ind på et spor eller i en blok, som ikke eksisterer i det anvendte format. Dette kan indikere et problem med at læse adressen til næste blok.
67	ILLEGAL SYSTEM T OR S - Denne specielle fejlmeddelelse indike-rer et ulovligt systemspor eller -sector.

FEJLNR	BESKRIVELSE
70	NO CHANNEL (ingen ledig kanal) - Enten er den forlangte kanal ikke til rådighed, eller samtlige kanaler er i brug. Maksimalt fem bufferarealer er til rådighed. En sekventiel fil kræver to buffere, en relativ fil kræver tre buffere, og fejl/kommando kanalen kræver en buffer. Man kan anvende enhver kombination af disse, blot ikke kombinationen kræver mere end fem buffere.
71	DIRECTORY ERROR - BAM (Block Availability Map) på disketten svarer ikke til dennes kopi i diskhukommelsen. Disketten skal ominitialiseres for at bringe dette i orden.
72	DISK FULL - Enten er blokkene på disketten opbrugt eller indholdsfortegnelsen har nået sin indtastningsgrænse. For at tillade lukning af den igangværende fil, sendes DISK FULL når der er to blokke til rådighed på disketten.
73	DOS VERSION NUMBER - DOS 1 og DOS 2 er læse-, men ikke skrivekompatible. Disketter kan skiftevis læses med begge DOS versioner, men en diskette formatteret med den ene version, kan der ikke skrives på med den anden version, fordi formatteringen er forskellig. Denne meddelelse vises, hvis der forsøges skrivning på en diskette, der er formatteret i et ikke kompatibelt format. Meddelelsen fremkommer også ved opstart af systemet og er i dette tilfælde ikke en fejl.
74	DRIVE NOT READY - Forsøg på at bruge diskettestationen uden diskette, eller når lågen til disketten er åben.

APPENDIKS C:
TILSLUTNINGER/PORTE FOR YDRE ENHEDER

(KONTAKT INFORMATIONER)



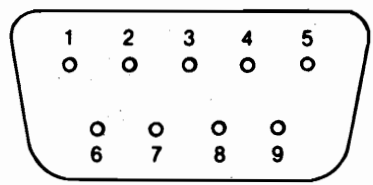


Side panel forbindelser

1. Strømsokkel - Den fri ende af kablet fra strømforsyningen tilsluttes til dette stik.
2. Strømkontakt - Tilslutter strømmen fra transformatoren.
3. Reset kontakt - Nulstiller computeren (varm start).
4. Spil port - Der findes to spilporte, nummerede 1 og 2. I hver port kan tilkobles et joystick eller en paddle. En lyspen kan tilsluttes port 1, det stik som er nærmest computerens forside. Brug af portene er normalt beskrevet til den aktuelle software.

Control Port 1

Pin	Type	Note
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	BUTTON A/LP	
7	+5V	MAX. 50mA
8	GND	
9	POT AX	



Control Port 2

Pin	Type	Note
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	BUTTON B	
7	+5V	MAX. 50mA
8	GND	
9	POT BX	

Bagsidens tilslutninger:

5. Programkapsel sokkel - Denne rektangulære sokkel er en parallel port, hvori programmer og spillekapsler kan isættes. Kan også bruges i forbindelse med specielle interface kapsler.

Cartridge Expansion Slot

Pin	Type
12	BA
13	DMA
14	D7
15	D6
16	D5
17	D4
18	D3
19	D2
20	D1
21	D0
22	GND

Pin	Type
1	GND
2	+5V
3	+5V
4	IRQ
5	R/W
6	Dot Clock
7	I/O 1
8	GAME
9	EXROM
10	I/O 2
11	ROML

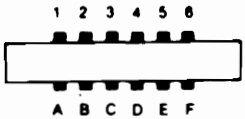
Pin	Type
N	A9
P	A8
R	A7
S	A6
T	A5
U	A4
V	A3
W	A2
X	A1
Y	A0
Z	GND

Pin	Type
A	GND
B	ROMH
C	RESET
D	NMI
E	S 02
F	A15
H	A14
J	A13
K	A12
L	A11
M	A10

6. Kassette stik - Her kan en Datassette båndstation model 1530 tilkobles for lagring og indlæsning af programmer og data.

Cassette

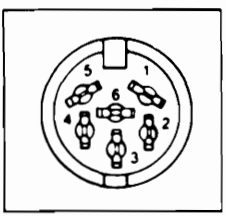
Pin	Type
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SENSE



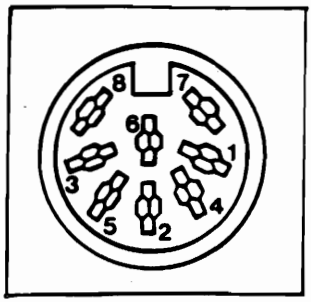
7. Seriel port - En seriel Commodore printer eller diskettestation kan tilsluttes direkte til Commodore 128 via denne port.

Serial I/O

Pin	Type
1	SERIAL SRQIN
2	GND
3	SERIAL ATN IN OUT
4	SERIAL CLK IN OUT
5	SERIAL DATA IN OUT
6	RESET



8. Video tilslutning - Denne DIN tilslutning afgiver direkte audio og composite video signaler. Disse kan forbindes med Commodore monitoren eller bruges med separate komponenter. Dette er en 40 kolonners output forbindelse.

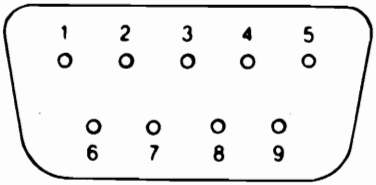


Pin	Type	Note
1	LUM/SYNC	Luminance/SYNC output
2	GND	
3	AUDIO OUT	
4	VIDEO OUT	Composite signal output
5	AUDIO IN	
6	COLOR OUT	Chroma signal output
7	NC	No connection
8	NC	No connection

9. HF tilslutning (antennetilslutningen) - Denne forbindelse forsyner Deres TV-apparat med såvel lyd som billede. (Et fjernsyn kan kun vise et 40 kolonners skærmbillede).

10. RGBI tilslutning - Fra denne nibenede sokkel kommer lyden og et RGBI(Red/Green/Blue/Intensity) signal. Dette er 80 kolonners output til monitor.

11. Brugerstik - I dette stik kan monteres forskellige typer af interface kapsler, blandt andet et Commodore modem.

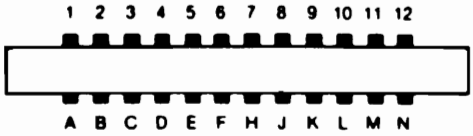


Pin	Signal
1	Ground
2	Ground
3	Red
4	Green
5	Blue
6	Intensity
7	Monochrome
8	Horizontal Sync
9	Vertical Sync

User I/O

Pin	Type	Note
1	GND	
2	+5V	MAX. 100 mA
3	$\overline{\text{RESET}}$	
4	CNT1	
5	SP1	
6	CNT2	
7	$\overline{\text{SP2}}$	
8	$\overline{\text{PC2}}$	
9	SER. ATN IN	
10	9 VAC	MAX. 100 mA
11	9 VAC	MAX. 100 mA
12	GND	

Pin	Type	Note
A	$\overline{\text{GND}}$	
B	FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



APPENDIKS D
SKÆRMBILLEDE KODER

SKÆRMBILLEDE KODER 40 KOLONNER

De følgende opstillinger viser alle de tegn, som er indbygget i et Commodore karaktersæt. Det viser hvilke værdier der skal POKEs til skærmens hukommelse (adresserne 1024 til 2023) for at få et bestemt tegn, på en 40 kolonners skærm. (Husk også at indstille farve hukommelsen - 55296 til 56295). Der vises også, hvilke tegn der korresponderer til en værdi, som PEEKes fra skærmen.

Der findes to tilgængelige tegnsæt. De er begge til tilgængelige på samme tid i 80 kolonners - mode. I 40 kolonners mode er de kun tilgængelige eet ad gangen. Der skiftes mellem sættene ved samtidig at holde SHIFT og COMMODORE tasterne nedtrykket.

Fra BASIC vil PRINT$(142) skifte til uppercase/grafik mode og PRINT$(14) skifter til uppercase/lowercase mode.

Enhver værdi i oversigterne kan også vises 'negativt'. Koden for negativ skrift fås ved at addere 128 til de viste værdier.

Kodeoversigt findes på de følgende 2 sider.

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	↑		30	<		60
A	a	1	←		31	=		61
B	b	2	SPACE		32	>		62
C	c	3	!		33	?		63
D	d	4	"		34			64
E	e	5	#		35		A	65
F	f	6	$		36		B	66
G	g	7	%		37		C	67
H	h	8	&		38		D	68
I	i	9	'		39		E	69
J	j	10	(		40		F	70
K	k	11	)		41		G	71
L	l	12	*		42		H	72
M	m	13	+		43		I	73
N	n	14	,		44		J	74
O	o	15	–		45		K	75
P	p	16	.		46		L	76
Q	q	17	/		47		M	77
R	r	18	0		48		N	78
S	s	19	1		49		O	79
T	t	20	2		50		P	80
U	u	21	3		51		Q	81
V	v	22	4		52		R	82
W	w	23	5		53		S	83
X	x	24	6		54		T	84
Y	y	25	7		55		U	85
Z	z	26	8		56		V	86
[		27	9		57		W	87
£		28	:		58		X	88
]		29	;		59		Y	89

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
	Z	90			103			116
		91			104			117
		92			105			118
		93			106			119
		94			107			120
		95			108			121
SPACE		96			109			122
		97			110			123
		98			111			124
		99			112			125
		100			113			126
		101			114			127
		102			115			

Koder fra 128-255 er det omvendte billede
af koderne fra 0-127

APPENDIKS E
ASCII og CHR$ KODER

ASCII OG CHR$ KODER

Dette appendiks viser Dem, hvilke karakterer, der vil vise sig, hvis De bruger PRINT CHR$(X), for alle mulige værdier af X. Det vil også vise hvilke værdier der fås ved indtastning af PRINT ASC ("x"), hvor x er en vilkårlig karakter, som kan udskrives. Dette er nyttig i forbindelse med evaluering af tegn modtaget i en GET instruktion, ved konvertering af upper/lower case og skrivning af tegnbaserede kommandoer (f. eks. skift mellem upper og lower case), som ikke kunne omsluttes af anførelsestegn.

Kodeoversigt findes på de følgende 3 sider.

Kodeoversigt

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	0		16	SPACE	32	ø	48
	1	CRSR ↓	17	!	33	1	49
	2	RVS ON	18	"	34	2	50
	3	CLR HOME	19	#	35	3	51
	4	INST DEL	20	$	36	4	52
WHT	5		21	%	37	5	53
	6		22	&	38	6	54
	7		23	'	39	7	55
DISABLES SHIFT C=	8		24	(	40	8	56
ENABLES SHIFT C=	9		25	)	41	9	57
	10		26	*	42	:	58
	11		27	+	43	;	59
	12	RED	28	,	44	<	60
RETURN	13	CRSR →	29	—	45	=	61
SWITCH TO LOWER CASE	14	GRN	30	.	46	>	62
	15	BLU	31	/	47	?	63

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
@	64	U	85		106		127
A	65	V	86		107		128
B	66	W	87		108	Orange	129
C	67	X	88		109		130
D	68	Y	89		110		131
E	69	Z	90		111		132
F	70	[	91		112	f1	133
G	71	£	92		113	f3	134
H	72	]	93		114	f5	135
I	73	↑	94		115	f7	136
J	74	←	95		111	f2	137
K	75		96		117	f4	138
L	76		97		118	f6	139
M	77		98		119	f8	140
N	78		99		120	SHIFT RETURN	141
O	79		100		121	SWITCH TO UPPER CASE	142
P	80		101		122		143
Q	81		102		123	BLK	144
R	82		103		124	CRSR	145
S	83		104		125	RVS OFF	146
T	84		105		126	CLR HOME	147

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
INST DEL	148	CYN	159		170		181
Brun	149	SPACE	160		171		182
Lys rød	150		161		172		183
Mørk blå	151		162		173		184
Grå	152		163		174		185
Lys grøn	153		164		175		186
Lys blå	154		165		176		187
Lys grå	155		166		177		188
PUR	156		167		178		189
CRSR	157		168		179		190
YEL	158		169		180		191

Kode 192-223 samme som 96-127
Kode 224-254 samme som 160-190
Kode 255 samme som 126

Se iøvrigt Appendix I

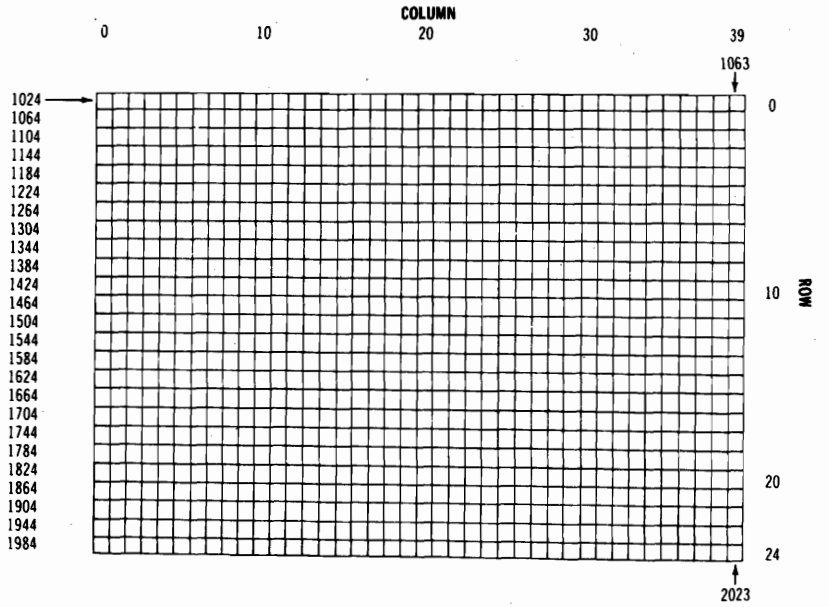
APPENDIKS F
SKÆRM OG FARVE MEMORY MAPS

SKÆRM OG FARVE MEMORY MAPS I C128 MODE, 40 KOLONNER og C64 MODE

De følgende MAPS viser hukommelsesadresserne i 40-kolonners mode (C128 og C64) som skal bruges for at identificere tegn og deres farver på skærmen. Hver MAP kontrolleres separat og består af 1000 positioner.

De karakterer som vises på MAPen kan kontrolleres direkte med POKE kommandoen.

SKÆRM MEMORY MAP

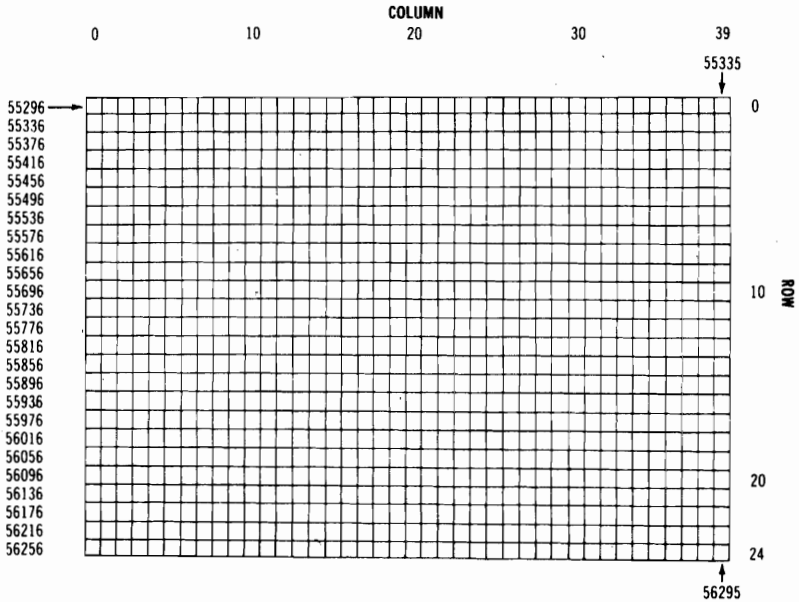


Skærmbilledet er POKEt med en værdi fra skærmkoder, appendiks D:

POKE 1024,13

og vil vise bogstavet M i skærmens øverste venstre hjørne.

FARVE MEMORY MAP



Farve MAPen POKEs med en farveværdi; dette ændrer tegnets farve. Tastes

POKE 55296,1

ændres bogstavet M's farve fra lysegrøn til hvid.

FARVE KODER

0 Sort	8 Orange
1 Hvid	9 Brun
2 Rød	10 Lyserød
3 Cyanblå	11 Mørkegrå
4 Purpurrød	12 Mellemgrå
5 Grøn	13 Lysegrøn
6 Blå	14 Lyseblå
7 Gul	15 Lysegrå

Kantfarve kontrol memory 53280
Baggrundsfarve kontrol memory 53281

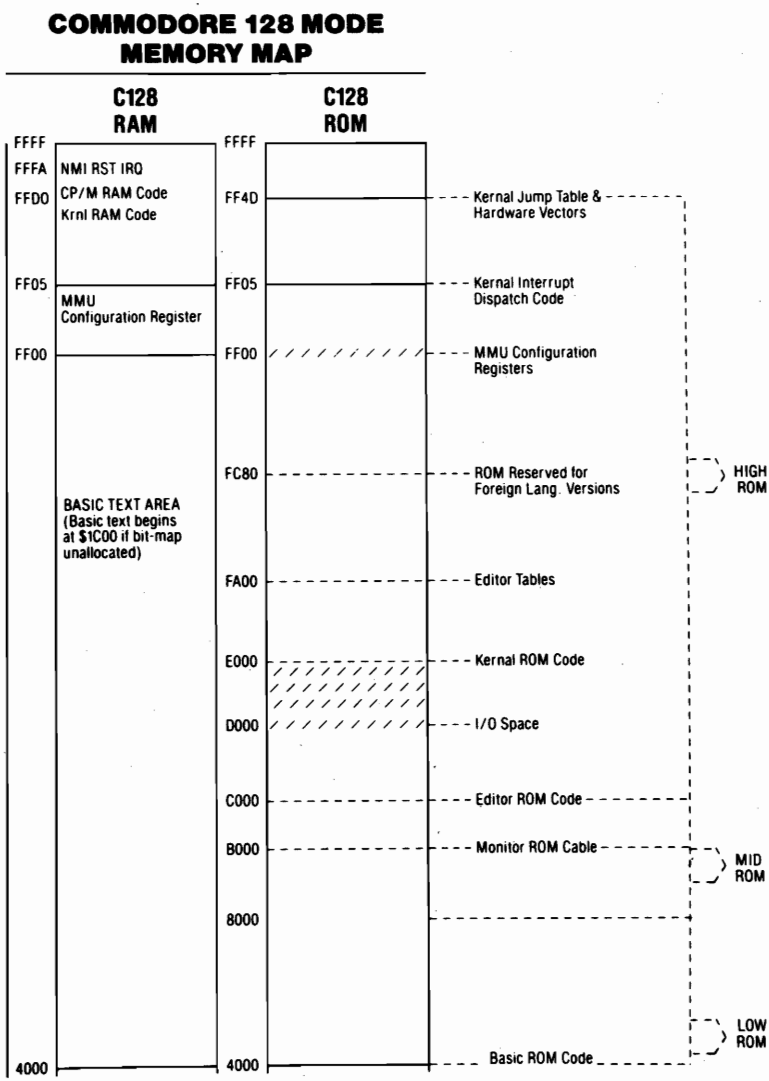
APPENDIKS G
AFLEDEDE MATEMATISKE FUNKTIONER

FUNKTION:	BASIC UDTRYK:
SECANT	SEC(X) = 1/COS(X)
COSECANT	CSC(X) = 1/SIN(X)
COTANGENS	COT(X) = 1/TAN(x)
ARC SINUS	ARCSIN(X) = ATN(X/SQR(-X*X+1))
ARC COSINUS	ARCCOS(X) = -ATN(X/SQR(-X*X+1))+ pi/2
ARC SECANT	ARCSEC(X) = ATN(X/SQR(-X*X+1))
ARC COSECANT	ARCCSC(X) =ATN(X/SQR(-X*X+1))+(SGN(X)-1*pi/2
ARC COTANGENS	ARCOT(X) = ATN(X)+ pi/2
HYPERBOLSK SINUS	SINE(X) = (EXP(X)-EXP(-X))/2
HYPERBOLSK COSINUS	COSH(X) = (EXP(X)+EXP(-X))/2
HYPERBOLSK TANGENS	TANH(X) = EXP(-X)/EXP(X)+EXP(-X))*2+1
HYPERBOLSK SECANT	SECH(X) = 2/(EXP(X)+EXP(-X))
HYPERBOLSK COSECANT	CSCH(X) = 2/(EXP(X)-EXP(-X))
HYPERBOLSK COTANGENS	COTH(X) = EXP(-X)/(EXP(X)-EXP(-X))*2+1
ARC HYPERBOLSK SINUS	ARCSINH(X) = LOG(X+SQR(X*X+1))
ARC HYPERBOLSK COSINUS	ARCCOSH(X) = LOG(X+SQR(X*X-1))
ARC HYPERBOLSK TANGENS	ARCTANH(X) = LOG(1+X)/(1-X)/2
ARC HYPERBOLSK SECANT	ARCSECH(X) = LOG((SQR(-X*X+1)+1)/X
ARC HYPERBOLSK COSECANT	ARCCSCH(X) = LOG((SGN(X)*SQR(X*X+1)/X)
ARC HYPERBOLSK COTANGENS	ARCCOTH(X) = LOG(X+1)/(X-1)/2
ARC HYPERBOLSK COTANGENS	ARCCOTH(X) = LOG(X+1)/(X-1)/2

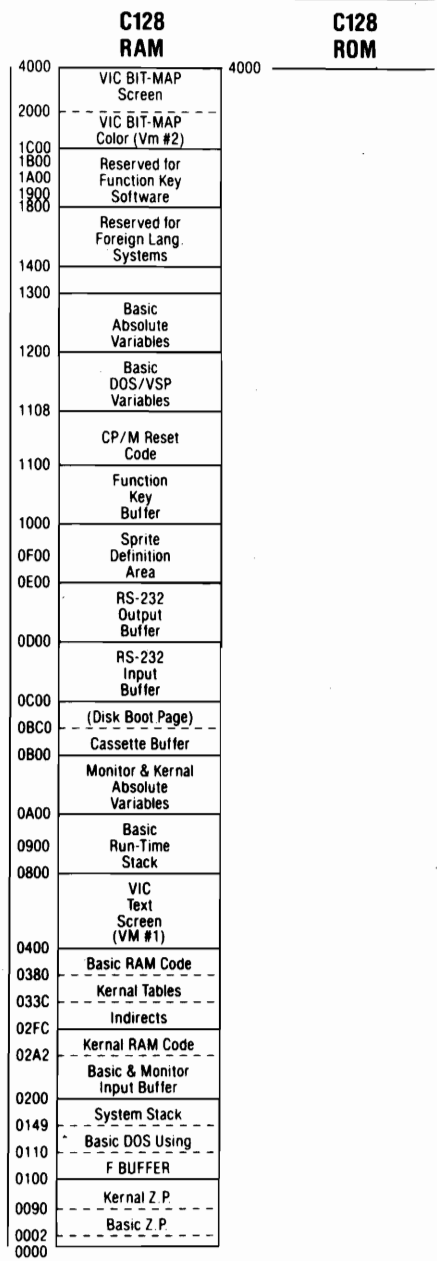
APPENDIKS H
MEMORY MAP

System Memory Map

Commodore 128 memory map vises herunder.



COMMODORE 128 MODE MEMORY MAP



APENDIX I
KONTROL OG ESCAPE KODER

Kontrol Koder

CHR$	Indtastning	Funktion	Effektiv i mode C64	C128
CHR$(2)	CTRL B	Understregning (80)		*
CHR$(5)	CTRL 2/CTRL E	Sæt tegnfarve til hvid(40) og (80)	*	*
CHR$(7)	CTRL G	Giver klokketone		*
CHR$(8)	CTRL H	Undertrykker skift af karaktersæt	*	
CHR$(9)	CTRL I	Muliggør skift af karaktersæt	*	
		Flytter markøren til næste angivne tabulatorposition		*
CHR$(10)	CTRL J	Send en vognretur med linieskift	*	
		Send et linieskift		*
CHR$(11)	CTRL K	Undertrykker skift af karaktersæt		*
CHR$(12)	CTRL L	Muliggør skift af karakter - mode		*
CHR$(13)	CTRL M	Send en vognretur og linieskift til computeren og start en linie i BASIC	*	*
CHR$(14)	CTRL N	Sæt karaktersættet til store/små bogstaver	*	*
CHR$(15)	CTRL O	Start bink (80)		*
CHR$(17)	CRSR DOWN/CTRL Q	Flyt markør en linie ned	*	*
CHR$(18)	CTRL 9	Bevirker at tegnet skrives som negativ	*	*

BEMÆRK: (40)...Kun skærm med 40 kolonner.
(80)...Kun skærm med 80 kolonner.

CHR$	Indtastning	Funktion	Effektiv i mode C64	C128
CHR$(19)	HOME/CTRL S	Flytter markøren til "home" position,øverst til venstre på skærmen (det aktuelle vindue)	*	*
CHR$(20)	DEL/CTRL T	Slet sidst indtastede karakter og flyt alle karakterer til højre for den slettede karakter en plads til venstre.	*	*
CHR$(24)	CTRL X	Sæt/slet tabulator	*	*
CHR$(27)	ESC/CTRL!	Send en ESC karakter		*
CHR$(28)	CTRL 3/CTRL 1	Sæt tegnfarven til rød (40) og (80)	*	*
CHR$(29)	CRSR /CTRL!	Flyt markøren en kolonne til højre	*	*
CHR$(30)	CTRL 6/CTRL ↑	Sæt tegnfarven til grøn (40) og (80)	*	*
CHR$(34)	"	Udskriv et dobbelt anførelsestegn på skærmen og sæt editor i anførelses - mode	*	*
CHR$(129)	C= 1	Sæt tegnfarven til orange (40) mørk purpur (80)		
CHR$(130)		Understregning fra (80)		*
CHR$(131)		Kør et program. Denne CHR$ kode virker ikke i PRINT CHR$(131),men virker fra tastaturbuffer.	*	*
CHR$(133)	F1	Reserveret CHR$ kode for F1 - tasten	*	*
CHR$(134)	F3	Reserveret CHR$ kode for F3 - tasten	*	*
CHR$(135)	F5	Reserveret CHR$ kode for F5 - tasten	*	*
CHR$(136)	F7	Reserveret CHR$ kode for F7 - tasten	*	*

BEMÆRK: (40)...Kun skærm med 40 kolonner.
(80)...Kun skærm med 80 kolonner.

CHR$	Indtastning	Funktion	Effektiv i mode C64	C128
CHR$(137)	F2	Reserveret CHR$ kode for F2 - tasten	*	*
CHR$(138)	F4	Reserveret CHR$ kode for F4 - tasten	*	*
CHR$(139)	F6	Reservetet CHR$ kode for F6 - tasten	*	*
CHR$(140)	F8	Reserveret CHR$ kode for F8 - tasten	*	*
CHR$(141)	SHIFT/RETURN	Send en vognretur og et linieskift uden at gå ind i en BASIC linie	*	*
CHR$(142)		Sæt karaktersættet til store bogstaver/grafik	*	*
CHR$(143)		Afbryd blink (80)		*
CHR$(144)	CTRL 1	Sæt tegnfarve til sort (40) og (80)	*	*
CHR$(145)	CRSR OP	Flyt markøren eller udskriften en linie op	*	*
CHR$(146)	CTRL 0	Afbryd negativ udskrift på skærmen	*	*
CHR$(147)	HOME	Slet skærmbilledet og flyt markøren til øverste venstre position	*	*
CHR$(148)	INST	Flyt karakter fra markørens position een plads til højre.	*	*
CHR$(149)	CTRL 2	Sæt tegnfarve til brun (40) mørkegul (80)	*	*
CHR$(150)	CTRL 3	Sæt tegnfarve til lyserød (40) og (80)	*	*
CHR$(151)	CTRL 4	Sæt tegnfarve til mørk grå (40) og mørk cyan (80)	*	*
CHR$(152)	CTRL 5	Sæt tegnfarve til mellemgrå (40) og (80)	*	*
CHR$(153)	CTRL 6	Sæt tegnfarve til lysegrøn (40) og (80)	*	*

BEMÆRK: (40)...Kun skærm med 40 kolonner.
(80)...Kun skærm med 80 kolonner.

CHR$	Indtastning	Funktion	Effektiv i mode C64	C128
CHR$(154)	C=7	Sæt tegnfarve til lyse-blå (40) og (80)	*	*
CHR$(155)	C=8	Sæt tegnfarve til lyse grå (40) og (80)	*	*
CHR$(156)	CTRL 5	Sæt tegnfarven til purpur (40) og (80)	*	*
CHR$(157)	CRSR LEFT	Flyt markøren en position til venstre	*	*
CHR$(158)	CTRL 4	Sæt tegnfarve til cyan (40) lys cyan (80)	*	*

BEMÆRK: (40)...Kun skærm med 40 kolonner.
(80)...Kun skærm med 80 kolonner.

Escape koder

Her følger en beskrivelse af escape sekvenser, der er til rådighed på Commodore 128. Escape sekvenserne fremkommer, når man trykker ESC - tasten ned og udløser den, hvorefter man trykker på en af de nævnte taster.

ESCAPE FUNKTION	ESCAPE TASTE
Annuller anførelses- og indsæt mode	ESC O
Slet til slutning af aktuel linie	ESC Q
Slet til start af aktuel linie	ESC P
Slet til slutning af skærm	ESC @
Flyt til start af aktuel linie	ESC J
Flyt til slutning af aktuel linie	ESC K
Start auto-indsæt funktion	ESC A
Slut auto-indsæt funktion	ESC C
Slet aktuel linie	ESC D
Indsæt linie	ESC I
Sæt standard tabulatorstop (8 mellemrum)	ESC Y
Slet alle tabulatorer	ESC Z
Muliggør rulning af skærmbillede	ESC L
Undertryk rulning af skærmbillede	ESC M
Rul skærmbillede op	ESC V
Rul skærmbillede ned	ESC W
Muliggør klokke (med kontrol-G)	ESC G

ESCAPE FUNKTION	ESCAPE TASTE
Undertryk klokke	ESC H
Undertryk markørens blinken	ESC E
Muliggør markørens blinken	ESC F
Sæt bund af skærmvindue ved markørposition	ESC B
Sæt top af skærmvindue ved markørposition	ESC T
Skift mellem 40/80 kolonners outputenhed	ESC X

De følgende ESCapekoder kan kun anvendes på en 80-kolonners skærm. (Se afsnit 8 , der indeholder pålysninger om 80-kolonners skærme).

Skift til understregnings-markør	ESC U
Skift til blok-markør	ESC S
Sæt skærm til negativ udskrift	ESC R
Sæt skærm tilbage til normal (ikke negativ) udskrift	ESC N

BEMÆRK....(40) kun 40 kolonners skærm
(80) kun 80 kolonners skærm

APPENDIKS J
MASKINSPROGS MONITOR

Commodore 128 har en indbygget maskinsprogs monitor program, der gør det let for brugeren at skrive og undersøge maskinsprogsprogrammer. Commodore 128 MONITOR indeholder en maskinsprogs monitor, en mini assembler og en disassembler. Den indbyggede monitor fungerer kun i C128 - mode, enten 40 eller 80 kolonner. Bemærk at udtrykket MONITOR i dette afsnit henviser til de omtalte programfunktioner og IKKE skærm. (to monitor,eng. = at undersøge, at se)

Maskinsprogsprogrammer, skrevet med Commodore 128 MONITOR kan køre selvstændigt, eller anvendes som meget hurtige subrutiner i et BASIC program, da COMMODORE 128 MONITOR er istand til at arbejde fredeligt sammen med BASIC.

Der skal udvises omtanke ved positionering af assembler programmer i hukommelsen, således at BASIC programmet ikke overskriver dem.

Man kommer i MONITOR fra BASIC ved at taste:

MONITOR RETURN

Opsummering af Commodore 128 Monitor kommandoer.

NØGLEORD	FUNKTION	FORMAT
ASSEMBLE	Assemblerer en 8502 kodelinie	A ⟨start adresse⟩⟨op.kode⟩ [operand]
COMPARE	Sammenligner to sektioner af hukommelsen og meddeler om afvigelser.	C ⟨start adresse⟩⟨slut adresse⟩ ⟨ny start adresse⟩
DISASSEMBLE	Disassemblerer en eller flere linier 8502 kode	D [⟨start adresse⟩⟨slut adresse⟩]

NØGLEORD	FUNKTION	FORMAT
FILL	Fylder et område af hukommelsen med en specificeret byte	F ⟨start adresse⟩⟨slut adresse⟩ ⟨byte⟩
GO	Starter udførelse på en specificeret adresse	G [adresse]
HUNT	Søger i hukommelsen inden for et angivet område alle forekomster af en speciel byte-kombination	H ⟨start adresse⟩⟨slut adresse⟩ ⟨byte 1⟩[⟨byte n⟩] H ⟨start adresse⟩⟨slut adresse⟩ ⟨ascii streng⟩
JUMP	Hop til en subrutine	J [adresse]
LOAD	Indlæs en fil fra bånd eller diskette	L "⟨filnavn ⟩"[⟨apparat #⟩]⟨load adresse⟩]
MEMORY	Viser de hexadecimale i hukommelsesområdet	M [⟨start adresse⟩][slut adresse]
REGISTERS	Viser 8502 registrene	R
SAVE	Gemmer på bånd eller diskette	S "⟨filnavn⟩"⟨apparat nr⟩⟨start adresse⟩⟨sidste adresse +1⟩
TRANSFER	Overfører koder fra en del af hukommelsen til en anden	T ⟨start adresse⟩⟨slut adresse⟩ ⟨ny start adresse⟩
VERIFY	Sammenligner hukommelse V med bånd eller disk	"⟨filnavn⟩"[⟨apparat nr.⟩] [⟨load adresse⟩]
EXIT	Afslutter Commodore 128 MONITOR	X
(punktum)	Assemblerer en linie 8502 kode	.
(større end)	Retter hukommelsen	⟩
(semikolon)	Retter 8502 register display	;

NØGLEORD	FUNKTION	FORMAT
(alfa-tegn)	Viser diskstatus, sender diskkommandoer, viser indholdsfortegnelse	@
	Disk status	@[apparat nr.]
	Disk kommando	@[apparatnr.],⟨kommandostreng⟩
	Disk katalog	@[apparat nr.],$[⟨drev⟩]:[⟨filspecifikation⟩]

BEMÆRK: ⟨ ⟩ indeholder krævede parametre.
[ ] indeholder valgfri parametre.

Commodore 128 viser 5 cifrede hexadesimale adresser i maskinsprogs monitor. Normalt har et hexadecimalt tal kun 4 cifre, der angiver adressen i det tilladte område. Cifret yderst til venstre angiver BANK konfigurationen (på det tidspunkt en given kommando udføres) i overensstemmelse med den følgende memory konfigurationstabel:

0 - RAM 0 alene
1 - RAM 1 alene
2 - RAM 2 alene
3 - RAM 3 alene
4 - INT ROM, RAM 0,I/O
5 - INT ROM, RAM 1,I/O
6 - INT ROM, RAM 2,I/O
7 - INT ROM, RAM 3,I/O
8 - EXT ROM, RAM 0,I/O
9 - EXT ROM, RAM 1,I/O
A - EXT ROM, RAM 2,I/O
B - EXT ROM, RAM 3,I/O
C - KERNAL + INT (lo), RAM 0,I/O
D - KERNAL + EXT (lo), RAM 1,I/O
E - KERNAL + BASIC, RAM 0, CHARROM
F - KERNAL + BASIC, RAM 0,I/O

OPSUMMERING AF MONITOR FELT BESKRIVELSER.

Følgende skilletegn står foran monitor data-felter (f.eks. memory dumps).

Når disse fremkommer som en kommando, vil skilletegnene give monitoren ordre om at ændre hukommelsen eller registret, ved brug af de givne data.

. ⟨punktum⟩ indleder en linie disassembleret kode.
) ⟨vinkel til højre⟩ indleder linier af memory dump.
; ⟨semikolon⟩ indleder linie af register dump.

Følgende skilletegn står foran nummerfelter (f.eks. adresser) og angiver grundtallet for værdien. Indtastet som kommandoer vil disse skilletegn ganske simpelt instruere monitoren om at vise værdierne i hver af de angivne grundtal.

⟨nul⟩ (standardværdi) indleder hexadecimale værdier.
$ ⟨dollar⟩ indleder hexadecimale (16-tals) værdier.
+ ⟨plus⟩ indleder decimale (10-tals) værdier.
& ⟨et-tegn⟩ indleder oktale (8-tals) værdier.
% ⟨procent⟩ indleder binære (2-tals) værdier.

Monitoren anvender de følgende tegn som feltafgrænsninger eller linieadskillelser (med mindre de findes i en ASCII streng).
⟨mellemrum⟩ afgrænsning - deler to felter.
, ⟨komma⟩ afgrænsning - deler to felter.
: ⟨kolon⟩ adskillelse - logisk linieslut.
? ⟨spørgsmålstegn⟩ adskillelse logisk linieslut.

COMMODORE 128 MONITOR KOMMANDOER

Med undtagelse af det, som er nævnt tidligere, er der ingen ændringer i MONITOR kommandoernes virkemåde. Vær imidlertid opmærksom på, at ethvert nummerfelt (f.eks. adresser, apparatnr. og databytes) skal angives som et grund nummer. Dette påvirker også operand feltet i ASSEMBLER kommandoen.

Bemærk også tilføjelsen i directory syntaksen til disk kommandoen.
Sem en yderligere hjælp til programmører, er faciliteten med Kernal fejlmeddelelser automatisk tilsluttet når man er i Monitoren. Det betyder at Kernal vil vise "I/O ERROR #" og fejlkoden, hvis der skulle være nogle mislykkede I/O forsøg fra MONITOR. Denne facilitet afbrydes, når man forlader MONITOR.

KOMMANDO: **A**
FORMÅL: Indtastning af en linie assembler kode.
SYNTAKS: **A** ⟨adresse⟩ ⟨mnemoteknisk kode⟩ ⟨operand⟩

⟨adresse⟩ En værdi som angiver, hvor i hukommelsen en opkode skal placeres.

⟨mnemoteknisk kode⟩ Et standard MOS teknologi assembler mnemoteknisk udtryk, f.eks. LDA, STX, ROR

⟨operand⟩ Operanden kan, når den kræves, være enhver af de gyldige adresserings modes.

Tryk på **RETURN** bruges for at indikere afslutning på en assembler linie. Hvis der er fejl i linien, vises et spørgsmålstegn for at gøre opmærksom på fejlen, og markøren flyttes til næste linie. Til rettelse af fejl kan skærmeditoren bruges på denne linie.

Eksempel: A 1200 LDX #$00
A 1202

Obs! Punktum (.) svarer til ASSEMBLE kommandoen.

Eksempel: . 2000 LDA #$23

KOMMANDO: **C**
FORMÅL: Sammenligning af to arealer i hukommelsen.
SYNTAKS: C ⟨adresse 1⟩ ⟨adresse 2⟩ ⟨adresse 3⟩

⟨adresse 1⟩ er et tal, som angiver begyndelsen på det hukommelsesareal, der skal sammenlignes med.

⟨adresse 2⟩ er et tal, som angiver afslutningen på det hukommelsesareal, der skal sammenlignes med.

⟨adresse 3⟩ er et tal, som angiver begyndelses på det andet hukommelsesareal, som skal kontrolleres.

Adresser, som ikke er i overensstemmelse, bliver udskrevet på skærmen.

KOMMANDO: **D**
FORMÅL: Disassemblering af maskinkode til assembler mnemotekniske udtryk og operander.
SYNTAKS: **D** [⟨adresse⟩] [⟨adresse 2⟩]

⟨adresse⟩ Et tal, som angiver i hvilken adresse disassemblering skal begynde.

⟨adresse 2⟩ En valgfri slutadresse på kode, som skal disassembleres.

Formatet på disassemblering er kun en smule forskelligt fra inputformatet på en assemblering. Forskellen er, at det første tegn ved disassemblering er et punktum i stedet for et A og at kodens hexadecimale værdi også vises.

En disassemblerings listning kan modificeres ved brug af skærmeditoren. Foretag først en ændring af mnemotekniske udtryk eller operander på skærmen, og tryk derefter på **RETURN**. Dette sender linien og kalder assembleren for flere modifikationer.

En disassemblering kan sideopdeles. Ved at taste et D **RETURN** kan næste side af disassemblering vises.

EKSEMPEL: D 3000 3003
.3000 A900 LDA #$00
.3002 FF ???
.3003 D0 2B BNE $3030

KOMMANDO: **F**
FORMÅL: Udfylder et adresseområde med en specificeret byte.
SYNTAKS: **F** ⟨adresse 1⟩ ⟨adresse 2⟩ ⟨byte⟩

⟨adresse 1⟩ Den første lokation som skal udfyldes.

⟨adresse 2⟩ Sidste lokation som skal udfyldes.

⟨byte værdi⟩ Et tal som skal bruges til udfyldning.

Denne kommando er nyttig ved initialisering af datastrukturer eller alle andre RAM områder.

EKSEMPEL: F 0400 0518 EA
Udfylder hukommelsesadresserne fra $0400 til $0518 med $EA (en NOP instruktion).

KOMMANDO: **G**
FORMÅL: Begynder udførelse af et program fra en specificeret adresse.
SYNTAKS: **G** [⟨adresse⟩]
⟨adresse⟩ Den adresse, hvor programudførelse skal begynde. Hvis adressen udelades, vil udførelsen begynde ved aktuelt PC. (Denne PC kan ses ved hjælp af R kommandoen).

GO kommandoen nulstiller alle registre (kan ses med R kommandoen) og påbegynder udførelse på den angivne startadresse. Det anbefales at være forsigtig med anvendelse af GO kommandoen. For, efter kørsel af et maskinsprogs program, at vende tilbage til C-128 MONITOR mode, bruges BRK instruktionen til sidst i programmet.

EKSEMPEL: G 140C
Programudførsel starter i adresse $140C.

KOMMANDO: **H**
FORMÅL: Gennemsøgning af hukommelsen indenfor et bestemt område for alle forekomster af et datasæt.
SYNTAKS: **H** ⟨adresse 1⟩ ⟨adresse 2⟩ ⟨data⟩

⟨adresse 1⟩ Begyndelsesadressen for søgning.

⟨adresse 2⟩ Slutadresse for søgning.

⟨data⟩ Et datasæt, som skal anvendes til søgning kan være tal eller en ASCII streng.

EKSEMPEL: H A000 A101 A9 FF 4C
Søgning efter data $A9, $FF, $4C fra A000 til A101.
H 2000 9800 'CASH
Søgning efter alfa-streng "CASH"

KOMMANDO: **L**
FORMÅL: Indlæsning af en fil fra kassettebånd eller disk
SYNTAKS: **L** ⟨"filnavn"⟩[,⟨device⟩] [,alt-load-adresse]

⟨"filnavn"⟩ ethvert gyldigt C128 filnavn.

⟨device⟩ tal til angivelse af, hvilken enhed, der skal læses fra. (1 = kassette, 8 = disk (evt. 9, A, o.s.v.)
(alt-load-adresse) mulighed for at indlæse en fil til en specificeret adresse.

LOAD kommandoen forårsager at en fil bliver indlæst i hukommelsen. Startadressen er indeholdt i filens to første bytes (en programfil) LOAD kommandoen indlæser med andre ord altid en fil til samme område, hvorfra den blev gemt. I maskinkode arbejde er dette meget vigtigt, da få programmer er komplet relokerbare. Indlæsning af en fil fortsætter, til filens slutmarkering (EOF) findes.

EKSEMPEL: L "PROGRAM 1",8
Indlæs filen benævnt PROGRAM1 fra diskette

KOMMANDO: **M**

FORMÅL: For visning af hukommelsen som hexadecimale og ASCII dump indenfor et specificeret adresseområde.

SYNTAKS: **M** [⟨adresse 1⟩] [⟨adresse 2⟩]

⟨adresse 1⟩ Første adresse i hukommelses-udlæsning. Hvis adresse 1 udelades, vises 1 side. Første byte er det 'bank nummer' som skal vises, de næste fire bytes er den første adresse som skal vises.

⟨adresse 2⟩ Sidste adresse som skal udlæses. Hvis den udelades, vises 1 side. Første byte er det 'bank nummer' som skal vises, de næste fire bytes er slutadressen.

Hukommelsen vises i følgende format:
⟩1A048 41 21 00 AA AA 00 58 32 :A[.**.X2

Indholdet af hukommelsen kan ændres ved brug af skærmeditoren. Flyt markøren til de data, der skal ændres, indtast den ønskede ændring og tryk på **RETURN**. Hvis der forefindes en forkert RAM adresse eller der gøres forsøg på at ændre ROM, vises en fejlmarkering (?). Udlæsning af ASCII data vises negativt (for at adskille dem fra de øvrige data på skærmen) til højre for de hexadecimale værdier. Kan et tegn ikke vises korrekt, vises det som et punktum i negativt (.). Som med disassemblerings kommandoen fås sidefremføring ved at taste M **RETURN**.

EKSEMPEL:
```
M F41F1 F4201
⟩F41F1 20 43 4F 4D 4D 4F 44 4F: COMMODO
⟩F41F9 52 45 20 45 4C 45 43 54:RE ELECT
⟩F4201 52 4F 4E 49 43 53 2C 20:RONICS,
```
NB: Ovenstående er dannet på en 40 - kolonners skærm.

KOMMANDO: **R**

FORMÅL: Viser vigtige 8502 registre. Program Status Register, Programtæller, regneakkumulator, X og Y indeks registre og Stack Pointer Register vises.

SYNTAKS: **R**

EKSEMPEL:
```
R
PC SR AC XR YR SP
; 1002 01 02 03 04 F6
```

Obs! ; (semikolon) kan bruges for at ændre register visning på samme måde som ⟩ kan bruges for at ændre hukommelses registrene.

KOMMANDO: **S**
FORMÅL: Gemmer en del af hukommelsen på bånd eller disk.
SYNTAKS: **S** ⟨"filnavn"⟩,⟨apparat⟩,⟨adresse 1⟩,⟨adresse 2⟩

⟨"filnavn"⟩ Ethvert lovligt C-128 filnavn. For at gemme data skal filnavnet omgives af anførelsestegn. Apostrof kan ikke bruges.

⟨apparat⟩ Enhedens nr. (kassette = 1, diskettestation = 8, o.s.v.).

⟨adresse 1⟩ Startadresse på det hukommelsesområde, der skal gemmes.

⟨adresse 2⟩ Slutadresse på det hukommelsesområde der skal gemmes + 1. Alle data op til men ikke inclusive databyten på denne adresse gemmes.

Den fil som dannes af denne kommando er en programfil. De første to bytes indeholder startadressen ⟨adresse 1⟩ på data. Filen kan genindlæses med L kommandoen.

EKSEMPEL: S "GAME",8,0400,0000
Gemmer indholdet af hukommelsen fra adresse $0400 til $0000 på diskette.

KOMMANDO: **T**
FORMÅL: Overfører hukommelses-segmenter mellem forskellige hukommelsesområder.
SYNTAKS: **T** ⟨adresse 1⟩ ⟨adresse 2⟩ ⟨adresse 3⟩

⟨adresse 1⟩ Startadresse for data som skal flyttes.

⟨adresse 2⟩ Slutadresse for data som skal flyttes

⟨adresse 3⟩ Startadresse på det område, hvortil data skal flyttes.

Data kan flyttes fra lav hukommelse til høj hukommelse og vice-versa. Supplerende hukommelses-segmenter af enhver længde kan flyttes frem eller tilbage. For hver flytning der foretages, sker en automatisk sammenligning af de enkelte bytes og såfremt uoverensstemmelser viser sig, listes disse med adresseangivelse.

EKSEMPEL: T 1400 1600 1401
Placerer data fra $1400 til og med $1600 een byte højere i hukommelsen.

KOMMANDO: **V**
FORMÅL: Sammenligner en fil på bånd eller diskette med indholdet i hukommelsen.
SYNTAKS: **V** ⟨"filnavn"⟩[,⟨device⟩] [,alt-start-adresse]

⟨"filnavn"⟩ Ethvert gyldigt C-128 filnavn.

⟨device⟩ Et tal værdi, som angiver på hvilken enhed filen befinder sig;

Kassette er 1 eller 01, diskette er 8 eller 08, 09, etc.

[alt-start-adresse] giver mulighed for at begynde sammenligning fra denne adresse.

VERIFY kommandoen sammenligner en fil med hukommelsens indhold. C-128 svarer med 'VERIFYING'. Hvis der findes en fejl, tilføjes ordet ERROR; hvis filsammenligningen er korrekt, kommer markøren atter på skærmen.

EKSEMPEL: V "ARBEJDSFIL",8

KOMMANDO: **X**
FORMÅL: Tilbage til BASIC
SYNTAKS: **X**

KOMMANDO: ⟩ (større end)
FORMÅL: Kan bruges til indsætning af op til 8 eller 16 hukommelses-adresser.
SYNTAKS: ⟩⟨adresse⟩[⟨data byte 1.....8/16⟩]

⟨adresse⟩ Første hukommelses-adresse, som som skal sættes.

⟨data byte 1⟩ Data som skal sættes i adressen.

⟨data byte 1.....8/16⟩ Data som skal placeres i den adresse, som følger den første adresse.

Det højeste antal bytes, som kan indtastes er otte (i 40-lolonners mode) eller seksten (i 80-kolonners mode). Hvis RETURN tasten nedtrykkes, vises indholdet af disse 8/16 adresser efter adressen.

EKSEMPLER:

⟩2000 Viser en linie med de bytes, der følger $2000.
⟩2000 31 32 38
indsætter værdier i $2000 og viser en linie med de bytes, der følger $2000.

KOMMANDO: @ (alfategn)
FORMÅL: Kan bruges til visning af diskstatus.
SYNTAKS: @ [enhed #],⟨disk cmd streng⟩

⟨enhed #⟩ apparatets enheds nummer.

⟨disk cmd streng⟩ strengkommando til disk

Obs! @ alene giver status på diskettedrevet.

EKSEMPEL: @ Undersøger disk
00, OK, 00, 00
@,I Initialiserer drev 8.

APPENDIKS K
BASIC 7.0
FORKORTELSER

BEMÆRK.: De forkortelser der nævnes herunder, arbejder i store bogstaver/grafik - mode. Tryk på den viste tast (de viste taster), og hold derefter SHIFT - tasten nedtrykket, og indtast de bogstaver der står efter ordet SHIFT.

NØGLEORD	FORKORTELSE
ABS	A SHIFT B
APPEND	A SHIFT P
ASC	A SHIFT S
ATN	A SHIFT T
AUTO	A SHIFT U
BACKUP	BA SHIFT C
BANK	B SHIFT A
BEGIN	B SHIFT E
BEND	BE SHIFT N
BLOAD	B SHIFT L
BOOT	B SHIFT O
BOX	ingen
BSAVE	B SHIFT S
BUMP	B SHIFT U
CATALOG	C SHIFT A
CHAR	CH SHIFT A
CHR$	C SHIFT H
CIRCLE	C SHIFT I
CLOSE	CL SHIFT O
CLR	C SHIFT L
CMD	C SHIFT M
COLLECT	COLL SHIFT E
COLINT	ingen
COLLISION	COL SHIFT L
COLOR	COL SHIFT O
CONCAT	C SHIFT O
CONT	ingen

NØGLEORD	FORKORTELSE
COPY	CO SHIFT P
COS	ingen
DATA	D SHIFT A
DEC	ingen
DCLEAR	DCL SHIFT E
DCLOSE	D SHIFT C
DEF FN	ingen
DELETE	DE SHIFT L
DIM	D SHIFT I
DIRECTORY	DI SHIFT R
DLOAD	D SHIFT L
DO	ingen
DOPEN	D SHIFT O
DRAW	D SHIFT R
DSAVE	D SHIFT S
DVERIFY	D SHIFT V
EL	ingen
END	ingen
ENVELOPE	E SHIFT N
ER	ingen
ERR$	E SHIFT R
EXIT	EX SHIFT I
EXP	E SHIFT X
FAST	ingen
FETCH	F SHIFT E
FILTER	F SHIFT I
FOR	F SHIFT O
FRE	F SHIFT R
FNXX	ingen
GET	G SHIFT E
GETKEY	GETK SHIFT E
GET#	ingen
GOSUB	GO SHIFT S
GO64	ingen
GOTO	G SHIFT O
GRAPHIC	G SHIFT R
GSHAPE	G SHIFT S
HEADER	HE SHIFT A
HELP	HE SHIFT L
HEX$	H SHIFT E
IF...GOTO	ingen

NØGLEORD	FORKORTELSE
IF..THEN..ELSE	ingen
INPUT	ingen
INPUT#	I SHIFT N
INSTR	IN SHIFT S
INT	ingen
JOY	J SHIFT O
KEY	K SHIFT E
LEFT$	LE SHIFT F
LEN	ingen
LET	L SHIFT E
LIST	L SHIFT I
LOAD	L SHIFT O
LOCATE	LO SHIFT C
LOG	ingen
LOOP	LO SHIFT O
MID$	M SHIFT I
MONITOR	MO SHIFT N
MOVESHAPE	ingen
MOVSPR	M SHIFT O
NEW	ingen
NEXT	N SHIFT E
ON...GOSUB	ON...GO SHIFT S
ON...GOTO	ON...G SHIFT O
OPEN	O SHIFT P
PAINT	P SHIFT A
PEEK	PE SHIFT E
PEN	P SHIFT E
PI	ingen
PLAY	P SHIFT L
POKE	PO SHIFT K
POS	ingen
POT	P SHIFT O
PRINT	?
PRINT#	P SHIFT R
PRINT USING	?US SHIFT I
PUDEF	P SHIFT U
RBUMP	RB SHIFT U
RCLR	R SHIFT C
RDOT	R SHIFT D
READ	RE SHIFT A
RECORD	R SHIFT E

NØGLEORD	FORKORTELSE
REM	ingen
RENAME	RE SHIFT N
RENUMBER	REN SHIFT U
RESTORE	RE SHIFT S
RESUME	RES SHIFT U
RETURN	RE SHIFT T
RGR	R SHIFT G
RIGHT$	R SHIFT I
RLUM	ingen
RND	R SHIFT N
RSPCOLOR	RSP SHIFT C
RSPPOS	R SHIFT S
RSPR	ingen
RSPRITE	RSP SHIFT R
RUN	R SHIFT U
RWINDOW	R SHIFT W
SAVE	S SHIFT A
SCALE	SC SHIFT A
SCNCLR	S SHIFT C
SCRATCH	SC SHIFT R
SGN	S SHIFT G
SIN	S SHIFT I
SLEEP	S SHIFT L
SLOW	ingen
SOUND	S SHIFT O
SPC(	SP SHIFT C
SPRCOLOR	SPR SHIFT C
SPRDEF	SPR SHIFT D
SPRITE	S SHIFT P
SPRSAV	SPR SHIFT S
SQR	S SHIFT Q
SSHAPE	S SHIFT S
STASH	S SHIFT T
STatus	ingen
STEP	ST SHIFT E
STOP	ST SHIFT O
STR$	ST SHIFT R
SWAP	S SHIFT W
SYS	ingen
TAB(	T SHIFT A
TAN	ingen

NØGLEORD	FORKORTELSE
TEMPO	T SHIFT E
TI	ingen
TI$	ingen
TO	ingen
TRAP	T SHIFT R
TROFF	TRO SHIFT F
TRON	TR SHIFT O
UNTIL	U SHIFT N
USR	U SHIFT S
VAL	V SHIFT A
VERIFY	V SHIFT E
VOL	V SHIFT O
WAIT	W SHIFT A
WHILE	W SHIFT H
WIDTH	WI SHIFT D
WINDOW	W SHIFT I
XOR	X SHIFT O

APPENDIKS L
DISK KOMMANDOER SAMMENDRAG

Dette tillæg indeholder et sammendrag over kommandoer til diskoperationer i C128 og C64 modes på Commodore 128. Hvis De ønsker yderligere oplysninger om disse kommandoer, henvises til kapitel 5, BASIC 7.0 Leksikon. I vejledningen til diskettestationen findes også information om disse kommandoer.

De NYE BASIC 7,0 kommandoer kan kun anvendes i C128 mode.
Alle BASIC 2,0 kommandoer kan bruges både i C128 og i C64 modes.

KOMMANDO	ANVENDELSE	BASIC 2.0	BASIC 7.0
APPEND	Tilføj data til en fil. @		*
BLOAD	Indlæs en binær fil, startende fra et angivet sted i hukommelsen.		*
BOOT	Indlæs og udfør program.		*
BSAVE	Gem en binær fil fra det angivne sted i hukommelsen.		*
CATALOG	Vis disketteindholdet på skærmen. @		*
CLOSE	Luk logisk diskfil.	*	*
CMD	Omdiriger skærmudskrift til diskfil.	*	*
COLLECT	Frigiv utilgængelig diskette plads. @		*

@ Selv om der ikke en enkelt tilsvarende kommando til denne funktion i BASIC 2.0, findes der en tilsvarende instruktion, bestående af flere kommandoer. Se disse BASIC 2.0 sammenstillinger i vejledningen til diskettestationen.

KOMMANDO	ANVENDELSE	BASIC 2.0	BASIC 7.0
CONCAT	Sammenkæd to data filer. @		*
COPY	Kopier filer fra drev til drev. @		*
DCLEAR	Klargør alle åbne kanaler på diskettedrevene. @ fra disk. @		*
DCLOSE	Lukker logiske diskfiler.		*
DIRECTORY	Vis diskettens indhold på skærmen. @		*
DLOAD	Indlæs et BASIC program		*
DOPEN	Åben en diskfil til læsning og/eller skrivning. @		*
DSAVE	Gem et BASIC program på disk. @		*
DVERIFY	Sammenlign program i hukommelsen med program på disk. @		*
GET#	Modtag indput fra åben diskfil.	*	*
HEADER	Formatter en disk. @		*
LOAD	Indlæs fil fra disk.	*	*
OPEN	Åben en fil til indput eller udskrift.	*	*
PRINT#	Udskriv data til fil.	*	*
RECORD	Sæt relative filpointere i position		*
RENAME	Udskift navnet på en diskfil.@		*
RUN filnavn	Udfør BASIC program fra disk.		*
SAVE	Gem program i hukommelsen på disk.	*	
VERIFY	Sammenlign program i hukommelsen med program på disk.	*	

@ Selv om der ikke en enkelt tilsvarende kommando til denne funktion i BASIC 2.0, findes der en tilsvarende instruktion, bestående af flere kommandoer. Se disse BASIC 2.0 sammenstillinger i vejledningen til diskettestationen.

ORDFORTEGNELSE

Som følge af, at jargonen indenfor dataområdet kan forekomme overvældende, har vi, for at hjælpe med at forstå dette komplekse emne, samlet en fortegnelse omfattende de mest brugte udtryk.

Akustisk kobler eller akustisk modem: En ydre enhed, som oversætter digitale signaler til hørlige toner, som kan transmitteres over telefonlinier. Hastigheden er begrænset til ca. 1200 bits i sekundet (bps). Se også 'direct-connect' modem.

Adresse: En etiket eller et nummer som identificerer det register eller hukommelseslokation, hvor en informationsmængde er lagret.

Alfanumerisk: Bogstaver, tal og specialsymboler fra tastaturet, men ikke grafiske tegn.

ALU: Aritmethic Logic Unit. En del af centralenheden (CPU), hvor binære data behandles.

Animering: Simulering af objektbevægelser på skærmen ved brug af graduerede, progressive bevægelser og brug af computer instrukser.

Array: (tabelområde) En datalager-struktur, i hvilken en serie af relaterede konstanter eller variabler lagres i fortløbende lagerlokationer. Hver konstant eller variabel i et tabelområde kaldes et element. Tilgang til et element er mulig ved anvendelse af symboler (subscript). Se symbol.

ASCII: (udtales aski) forkortelse for American Standard Code for Information Interchange. En standard syv-bit kode til repræsentation af alfanumeriske karakterer. Er nyttig i forbindelse med afsendelse af oplysninger fra tastatur til computer, samt mellem computere. Se Karakterstreng koder.

Assembler: Et program som oversætter assembler-sprog instruktioner til maskinsprogs instruktioner.

Assembler sprog: Et maskinorienteret sprog i hvilket mnemotekniske udtryk bruges til repræsentation af den enkelte maskinsprogs instruktion. Assembler sprog er udviklet specielt til den computer, hvortil det skal benyttes. Se CPU og maskinsprog.

Assignment Statement: En BASIC instruktion som sætter en variabel, konstant eller et tabelelement med en bestemt numerisk eller strengværdi.

Asynkron transmission: Et skema efter hvilket datakarakterer sendes i uregelmæssige intervaller. Transmissionshastighed på almindelige telefonlinier er begrænset til ca. 2400 bps. Se også Synkron transmission.

Attack: Den tid indenffor hvilken en musikalsk nodes styrke forøges fra nul til maksimum.

Baggrundsfarve: Farven på den del af skærmen, hvorpå karaktererne er placeret.

BASIC: Står for Beginner's All-purpose Symbolic Instruction Code.

Baud: Seriel datatransmissions-hastighed. Oprindeligt en telegrafisk betegnelse. 300 baud er tilnærmelsesvis det samme som 30 bytes i sekundet.

Binær: Et totals-system, hvor alle tal repræsenteres af værdierne 0 eller 1. Grundtallet er 2

Bit: Forkortelse for Binary digit. En bit er den mindste enhed i en computer. Hver bit kan have een af to mulige værdier, nul eller eet. Hvis værdien på en bit er 1, siges den at være on. Er værdien 0 siges den at være off.

Bit kontrol: En måde, hvormed serielle data transmitteres, idet den enkelte bit har en bestemt betydning og hvor en enkelt karakter er tilføjet en start- og en stopbit.

Bit Map Mode: En avanceret grafisk mode (tilstand) i Commodore 128, hvormed hvert eneste punkt på skærmen kan kontrolleres. Et punkt på skærmen svarer til en bit i hukommelsen.

Border Color: Kantfarve. Det farvede felt rundt om selve skærmbilledet.

Branch: At springe til en programsektion og udføre denne. GO-TO og GOSUB er eksempler på BASIC branch instruktioner.

Bubble Memory: Boblehukommelse. En relativ ny type computerhukommelse, som bruger små magnetiske 'lommer' eller 'bobler' til datalagring.

Burst Mode: En speciel hurtig kommunikation mellem diskstation og computer, hvor datatransmissionen sker med en hastighed, der er mange gange større end den normale.

Bus: Parallelle linier som bruges til at overføre signaler mellem forskellige enheder. Computere er ofte beskrevet ud fra deres busstruktur (S-100-bus computere o.lign.).

Bus Netværk: Et system hvori alle stationer, eller computerenheder, kommunikerer gennem en fælles forsyningskanal, eller bus.

Byte: En gruppe bestående af otte bits (en oktet), som udgør den mindste adresserbare enhed i en computer. Hver lokation i Commodore 128's hukommelse indeholder een byte information. En byte en den lagerenhedsstørrelse, som kræves for at lagre en karakter i hukommelsen. Se bit.

Carrier Frekvens: Bærefrekvens. En konstant frekvens mellem to kommunikerende enheder. Moduleres for at overføre binær information.

Character: Karakter. Ethvert symbol på computerens tastatur, som kan udskrives på skærmen. Karakterer omfatter tal, bogstaver, skilletegn og grafiske symboler.

Character Memory: Karakterhukommelse. Det område i Commodore 128's hukommelse som indeholder de karaktermønstre, som vises på skærmen.

Character Set: Karaktersæt. En gruppe af relaterede tegn. Karaktersættet i Commodore 128 består af: uppercase-bogstaver, lower-case bogstaver og grafiske tegn.

Character String Code: Karakterstreng kode. Den tildelte numeriske værdi som repræsenterer en Commodore 128 karakter i hukommelsen.

Chip: Et meget lille elektronisk kredsløb som udfører computer operationer som grafik, lyd og input/output.

Clock: En mikroprocessors tidstakt.

Clocking: En teknik som anvendes for at synkronisere en sende/modtage kommunikationsenhed, som er moduleret til indkodning af binær information.

Collision Detection: En opgave som udføres i flerbrugersystemer for at und gå, at to maskiner transmitterer samtidig. Også betegnelse for undersøgelse af to eller flere sprites indbyrdes placering.

Color Memory: Farvehukommelse. Det område i Commodore 128's hukommelse som kontrollerer farven på den enkelte lokation i skærmhukommelsen.

Command: Kommando. En BASIC instruktion som bruges i direkte mode for at udføre en operation. Se Direct Mode.

Compiler: Et program som oversætter et højniveau sprog, som f. eks. BASIC, til maskinsprog.

Composite Monitor: En enhed asom anvendes til at vise et 40 kolonners skærmbillede.

Computer: En elektronisk, digital enhed som lagrer og behandler information.

Condition: Vilkår. Udtryk mellem ordene IF og THEN, som afgøres at være sande eller falske i en IF...THEN instruktion. Vilkåret i en IF...THEN instruktion sætter computeren i stand til at træffe valg.

Coordinate: Koordinat. Et enkelt punkt i et skema, som indeholder vertikale (Y) og horisontale (X) værdier.

Counter: Tæller. En variabel som bruges til at holde styr på det antal gange en begivenhed er indtruffet i et program.

CPU: Central Processing Unit. Centralenhed. Den del af en computer som indeholder de kredse som kontrollerer og udfører computerinstruktionerne.

Cursor: Markør. Den blinkende firkant som markerer den aktuelle position på skærmen.

Data: Tal, bogstaver eller symboler som er input til behandling i en computer.

Data Base: En stor mængde data som er lagret på en velorganiseret måde. Et database management system er et program som tillader adgang til informationen.

Data Link Layer: En logisk del af datakommunikations kontrol som stort set sikrer, at kommunikation mellem sammenkoblede enheder sker uden fejl.

Data Packet: Datapakke. En måde hvorpå serielle transmissionsdata pakkes effektivt. Includerer en fejlkontrol sekvens.

Data Rate eller Data Transfer Rate: Datatransmissions hastighed. Den hastighed hvormed data sendes til en modtagende computer. Opgives i bits pr. sekund (bps).

Datassette: Kassettebåndstation. En enhed som bruges til sekventiel lagring af programmer og datafiler på bånd.

Debug: At rette (afluse) et program for fejl.

Decay: Den tid indenfor hvilken en tone styrke nedsættes fra maxværdien til en vedligeholdelsesstyrke kaldet 'sustain niveau' Se sustain.

Decrement: At formindske en indexvariabel eller tæller med en bestemt værdi.

Dedicated eller Leased Line: Fast telefonkredsløb. Et specielt telefonlinie arrangement som leveres af telefonselskaberne. Kræves af bestemte computere eller terminaler, hvor systemforbindelser konstant skal være etablerede.

Delay Loop: Forsinkelsesløkke. En tom FOR...NEXT løkke som nedsætter hastigheden på programudførelsen.

Dial-Up Line: Opkaldslinie. Normal telefonforbindelse, der kan bruges til transmission af data.

Digital: I forbindelse med computer teknologi og datakommunikation indkodes al information som værende bits af værdierne 1 eller 0, eller 'on' og 'off'.

Dimension: Den definerede størrelse af et ARRAY. Specificerer den retning, i hvilken tabelelementer er lagret langs en akse. For eksempel har et todimensionalt array en X-akse for rækker og en Y-akse for kolonner. Se Array.

Direct mode: Direkte mode (tilstand). Den tilstand som, umiddelbart efter at RETURN er nedtrykket, udfører Basic kommandoer. Kaldes også tidstro tilstand. Se Command.

Direct Connect Modem: En enhed som konverterer en computers digitale signaler til elektroniske impulser, som kan transmitteres via telefonledninger. Det modsatte af Akustisk Kobling.

Disable: Slukke/undertrykke. At slukke for en bit, byte eller specifik operation i en computer.

Disk Drive: Diskdrev. En ydre lagerenhed, som lagrer og indlæser filer på og fra disketter.

Disk Operating System: Disk Operativ System. Program som overfører infor mation til og fra en diskette. Kaldes ofte DOS.

Duration: Varighed. Den tid hvori en musikalsk node spilles.

Electronic Mail eller E-Mail: En kommunikationsservice for computerbrugere hvorved tekstmeddelelser sendes til en central computer, eller 'elektronisk postboks', og senere modtages af adressaten.

Enable: Tænde/muliggøre. At tænde en bit, byte eller specifik funktion i en computer.

Envelope Generator: En del af Commodore 128 som producerer specifikke bølgeformer (savtak, trekant, pulsbredde og støj) for musikalske noder. Se Waveform.

EPROM: En PROM hvis indhold kan slettes af brugeren, normalt ved at belyse den med ultraviolet lys. Se PROM.

Error Checking eller Error Detection: Fejlkontrol. Softwarerutiner som identificerer, og ofte retter, fejlramte data.

Execute: At udføre de specificerede rutiner i en kommando eller programinstruktion.

Expression: Udtryk. En kombination af konstanter, variabler eller tabelelementer som bruges i forbindelse med logiske, matematiske eller sammenlignende operatorer som returnerer en numerisk værdi.

Fil(e): Et program eller en samling af data, der behandles som en enhed og lagres på diskette eller bånd.

Firmware: Computer instruktioner, som er lagret i ROM, som f. eks. et spil på programkapsel.

Frequency: Frekvens. Antallet af en tones lydbølger pr. sekund. Frekvens korresponderer med tone højden. (bas, diskant).

Full-Duplex: Transmissionsform. Sætter to computere i stand til at sende og modtage data på samme tid.

Function: Funktion. En foruddefineret operation, som returnerer en enkelt værdi.

Function Keys: Funktionstaster. De fire taster længst mod højre på Commodore 128's tastatur. Den enkelte tast kan programmeres til at udføre en serie instruktioner. Da tasterne kan 'skiftes', kan ialt otte forskellige instruktionssæt indlægges.

Graphics: Grafik. Visuelle skærmbilleder, der repræsenterer data i computerens hukommelse (d.v.s. karakterer, symboler og billeder).

Graphic Characters: Grafiske tegn. Tegn på computerens tastatur, som ikke er alfanumeriske.

Grid: Gitter. En todimensional matrix, opdelt i rækker og kolonner. En sådan matrix bruges til at tegne sprites og programmerbare karakterer.

Half-Duplex: Halv duplex - transmissionsform. Transmission kan kun foretages i een retning ad gangen; hvis den ene enhed sender, kan den anden blot modtage data, til det er dens tur til at sende Hardware: Fysiske komponenter i et computersystem. F. eks. tastatur, diskettestation og printer.

Hexadecimal: Refererer til 16-tal systemet. Maskinsprogsprogrammer skrives ofte i hexadecimal notation.

Home: Skærmbilledets øverste venstre hjørne.

IC: Integrerede kredse. En silic chip der indeholder et elektrisk kredsløb dannet af komponenter som transistore, dioder, modstande og kondensatorer. Integrerede kredse er mindre, hurtigere og mere stabile end individuelle kredse, der indeholder komponenter af den ældre type.

Increment: Forøgelse. At forøge en index variabel eller tæller med en specificeret værdi.

Index: Variabeltælleren i en FOR...NEXT løkke.

Input: Inddata. Data som indlæses i en computer for behandling. Data kan være input fra tastatur, diskette, kassettebånd eller via et modem (transmission).

Integer: Heltal. Et tal, som ikke indeholder decimaler.

Interface: Grænseflade. Samlepunktet mellem en computer og en ekstern enhed, enten en operatør, en ydre enhed eller en kommunikationsenhed. Et interface kan være et fysisk stik eller logisk, indeholdende software.

I/O: Input/Output. Henviser til processen ved indtastning af data til computer eller overføring af data fra computer til disk, printer eller andet lagringesmedie.

Keyboard: Tastatur. Input komponent i et computersystem.

Kilobyte: Kb(yte) = 1024 bytes.

Local Network: Lokalt netværk. Et af mange kortdistance datakommunikations skemaer kendetegnet ved almindelig anvendelse af et transmissionsmedium, som omfatter mange enheder og som arbejder med høje hastigheder. Kaldes også Local Area Network, LAN.

Loop: Løkke. Et programafsnit som som udføres et specificeret antal gange.

Machine Language: Maskinsprog. Et sprog på laveste niveau, som en computer kan forstå. Computeren oversætter alle højniveausprog, som for eksempel BASIC, til maskinsprog, før nogen instruktioner overhovedet udføres. Maskinsprog skrives i binær form og instruktionerne kan derfor direkte udføres af computeren. Kaldes også maskinkode eller objektkode.

Matrix: Et todimensionalt skema med række og kolonneværdier.

Memory: Hukommelse. Lageradresser i en computer. RAM og ROM er to forskellige hukommelsestyper.

Memory Location: Hukommelsesadresse. En specifik lageradresse i computerens interne lager. I Commodore 128 findes 131072 lokationer, (0-131072).

Microprocessor: En CPU (centralenhed) som indeholdes i et enkelt integreret kredsløb. De microprocessorer, som anvendes i Commodore 128 er 6510, 8502 og Z80.

Mode: Den tilstand, hvori der arbejdes.

Modem: Står for MOdulation/DEModulation. En enhed, som omsætter elektriske signaler til hørlig lyd, som kan transmitteres via telefonkredsløb og ved modtagelsen igen omsættes til elektriske impulser.

Monitor: En skærmenhed svarende til en fjernsynsskærm men med en bedre (skarpere) gengivelse.

Motherboard: En enhed som, i et busorienteret system, indeholder de buslinier og tilslutningsstik, som systemet skal bruge.

Multicolor Character Mode: Flerfarve Karakter Mode. En grafisk mode, som tillader brugeren at arbejde med 4 forskellige farver indenfor en 8 * 8 karakters ramme.

Multicolor Bit Map Mode: En grafisk mode, som tillader brugeren at bruge een af fire farver til hver enkelt pixel indenfor en 8 * 8 karakters ramma. Se Pixel.

Multiple-Access Network: Flerbruger Netværk. Et fleksibelt system, hvori alle arbejdspladser samtidig har adgang til netværket; der tages forholdsregler mod, at to computere vil transmittere på samme tidspunkt.

Null String: Nulstreng. Et tomt tegn (""). En karakter, som endnu ikke er tildelt en karakterstreng kode. Giver fejlmeddelelse ILLEGAL QUANTITY ERROR, hvis den bruges i en GET instruktion.

Octave: Oktav. Een komplet serie på 13 noder i den musikalske skala.

Operating System: Operativsystem. Et indbygget program, som styrer alt, hvad computeren foretager sig.

Operator: Et symbol, som fortæller computeren, at den skal udføre en matematisk, logisk eller sammenlignende operation på de specificerede variabler, konstanter eller tabelelementer i udtrykket. De matematiske operatorer er +, -, *, / og ↑. De sammenlignende er 〈, =, 〉, 〈=, 〉= og 〈〉. De logiske operatorer er AND, OR , NOT og XOR.

Order of Operations: Operationsrækkefølge. Den sekvens, hvori udregninger foretages i et matematisk udtryk. Kaldes også Operations Hierarki.

Parallel Port: En port der anvendes til transmision af data, en byte ad gangen over flere tråde.

Parity Bit: Paritetsbit. Et 1 eller 0 som føjes til en gruppe bits og derved afgør, om summen af bittene er lige eller ulige.

Peripheral: Ydre enhed. Enhver ekstra enhed, som tilkobles computeren, som for eksempel en diskettestation, en printer, et modem eller et joystick.

Pitch: Det tonehøjde på en node, som fastsættes af lydbølgens frekvens. Se Frekvens.

Pixel: Computerudtryk for et billedelement. Hvert punkt som indgår i dannelsen af et skærmbillede kaldes en pixel. Hvert tegn på skærmen dannes indenfor et 8 * 8 pixel område. Et helt skærmbillede dannes af en matrix bestående af 320 * 200 pixel. I bit-map mode svarer hver pixel til en bit i computerens hukommelse.

Polling: En kommunikations kontrolmetode, som bruges af visse computer/terminal systemer, hvorved en "hoved"-station i et netværk i rækkefølge spørger enhederne, om de har information til afsendelse.

Pointer: Et register som bruges for at indikere adressen på en position i hukommelsen.

Port: En kanal hvorigennem data overføres til og fra CPU'en. En 8bit CPU kan adressere 256 porte.

Printer: Ydre enhed, som udskriver på papir. En sådan udskrift kaldes ofte en HARD COPY.

Program: En serie instruktioner, som får en computer til at udføre en bestemt opgave. Programmer kan gemmes på disketter eller kassettebånd, ligge i en computers hukommelse eller udskrives på en printer.

Programmable: Programmerbar. En problemkreds kan løses ved anvendelse af computerinstruktioner.

Program Line: Programlinie. En eller flere instruktioner i et program, som har et foranstillet linienummer. På Commodore 128 må een programlinie højst have en længde på 160 karakterer.

PROM: Programmable Read Only Memory. En hukommelsesenhed, hvorfra der kun kan læses. Dennes indhold kan ikke ændres.

Protocol: De regler hvorunder computere udveksler information og som endvidere bestemmer, hvorledes dataenheder til overførsel skal organiseres.

Random Access Memory: (RAM) Det programmerbare område i computerens hukommelse, hvorfra der kan læses og hvortil der kan skrives. Alle RAM adresser er umiddelbart tilgængelige til enhver tid og i enhver rækkefølge. Indholdet af RAM slettes, når der slukkes for computeren.

Random Number: Et nicifret decimalt tal fra 0.000000001 til 0.999999999 som genereres af RaNDom (RND) funktionen.

Read Only Memory: (ROM). Den permanente del af computerens hukommelse. Ind holdet af ROM lokationer kan læses, men ikke ændres. Commodore 128's ROM indeholder BASIC fortolkeren, karaktersæt samt dele af operativsystemet.

Register: Enhver hukommelsesadresse i RAM. Hver register indeholder een byte. Et register kan indeholde enhver værdi mellem 0 og 255 i binær form.

Release: Den tid indenfor hvilken en musikalsk nodes styrke falder fra sustain niveau til nul.

Remark: Kommentarer til forklaring af et program. Remarks udføres ikke af computeren, men vises i en programudskrift.

Resolution: Opløsning. Tætheden mellem skærmens pixels. Er bestemmende for et skærmbilledes finhed og detaljemængde.

RGBI Monitor: Rød/Grøn/Blå/Intensitet. En højopløsnings skærmenhed, nødvendig for 80 kolonners skærmformat.

Ribbon Cable: En gruppe af sammenknyttede parallelle ledninger.

Ring Network: Et system hvori alle stationer er sammenkædede til at udføre en uendelig løkke eller cirkel.

RS-232: En anbefalet standard for elektroniske og mekaniske specifikationer af en seriel transmissionsport. Commodore 128 parallel bruger port kan anvendes som seriel port, hvis tilgangen sker via software. I nogle tilfælde kræves også en interface enhed.

Screen: Skærm. Videoskærm som enten kan være en fjernsynsskærm eller en monitor.

Screen Code: Et tal som repræsenterer en karakter i skærmhukommelsen. Når der nedtrykkes en tast på tastaturet, indsættes skærmkoden for det valgte tegn automatisk i skærmhukommelsen. En karakter kan også vises på skærmen ved at indlægge dens skærmkode direkte i skærmhukommelsen med POKE kommandoen.

Screen Memory: Skærmhukommelse. Det areal i Commodore 128's hukommelse som indeholder den information, som vises på videoskærmen.

Seriel Port: En port der anvendes til seriel transmission af data, bits transmitteres enkeltvis over en enkelt tråd.

Serial Transmission: Seriel transmission. Afsendelse af databits i sekventiel rækkefølge.

Software: Computer programmer (instruktionssæt) lagret på diskette, bånd eller kapsel og som kan indlæses i RAM. Software fortæller computeren, hvad den skal udføre.

Sound Interface Device: (SID). Den MOS 6581 lyd synthesizer chip, som udfø rer alle lydeffekter i Commodore 128.

Source Code: Kildetekst. Et ikke-udførbart program skrevet i et højniveau sprog. Sourcekoden skal af en compiler eller assembler oversættes til objektkode (maskinsprog), som kan forstås af computeren.

Sprite: En programmerbar, flytbar højopløsnings grafisk figur. Også kaldet en MOB (Movable Object Block).

Standard Character Mode: Den tilstand, hvori Commodore 128 virker, når den tændes og når der skrives programmer.

Start Bit: En bit eller en gruppe bits, som identificerer begyndelsen af et dataord.

Statement: Instruktion. En BASIC instruktion i en programlinie.

Stop Bit: En bit eller en gruppe bits, som identificerer slutningen af et dataord og definerer mellemrummet mellem disse.

String: Streng. En alfanumerisk karakter eller serier af karakterer omgivet af anførelsestegn.

Subroutine: Subrutine. Et uafhængigt programafsnit, som er adskilt fra hovedprogrammet, og som udfører en speciel opgave. Subrutiner kaldes fra hovedprogrammet med GOSUB instruktioner og skal afsluttes med en RETURN instruktion.

Subscript: En variabel eller konstant, som modsvarer et bestemt element indenfor et tabelområde ved sin position i området.

Sustain: Det midterste styrkeområde af en musikalsk node.

Synchronious Transmission: Datakommunikation som anvender en synkronisering, eller et signal fra en tidsenhed (cloking), mellem sendende og modtagende enheder.

Syntax: Et programmeringssprogs grammatiske regler.

Tone: Hørlig lyd med specifik tonehøjde og bølgeform.

Transparent: Beskriver en computer operation, som ikke kræver indgreb af brugeren.

Variable: En del af et lager, som indeholder et skiftende alfanumerisk værdi. Variabelnavne kan have enhver længde, men kun de første to karakterer lagres af Commodore 128. Det første tegn skal være et bogstav.

Video Interface Controller (VIC): Den MOS 6566 chip som styrer 40 kolonners grafiske funktioner i Commodore 128.

Voice: Stemme. En lydproducerende komponent i SID chippen. SID chippen indeholder tre stemmer, så Commodore 128 kan producere tre forskellige lyde samtidig. Hver stemme består af en tone oscillar/bølgeform generator, en 'envelope' generator og en amplitude modulator.

Waveform: Bølgeform. En grafisk repræsentation af en tones bølgeform. Bølgeformen er bestemmende for tonens fysiske karakteristika.

Word: Et antal bits, som af CPU'en behandles som en enkelt enhed. I en 8-bits maskine er ordlængden otte bits; i en 16-bits maskine er ordlængden 16 bits.

INDEX

S