EN DATA BECKER BOG

BRÜCKMANN · ENGLISCH
GERITS · STEIGERS

AMSTRAD

464/664 & 6128

INTERN

Simpelthen bibelen til 464/664 & 6128

DANSK / NORSK

udgave

DEN STORE FANTASTISKE

PROFI-PAINTER SENSATION!

Nu kan du tegne og male af hjertens lyst, og du kan gemme mesterværkerne.

'MAC AMSTRAD'

I hovedmenuen støder man på begreber som INFO, FILE, EDIT, FONT og OPTION. I den venstre side af rammen er der 16 såkaldte ikoner (små billeder, der forestiller den funktion, de udfører). De forskellige typer af pensler og penne vælges ud fra disse. I højre side af skærmen findes alle de mønstre, der kan anvendes til at fylde figurer ud med. Men i bunden af billedet har vi lækkerierne! Der kan vælges mellem blyant, spraydåse, tusch, viskelæder og geometriske figurer lige fra en enkelt linie til hele elipser, rektankler og rhomber, som frit kan placeres på skærmen. Og det i fantastiske variationer. Alt i alt er der 7 farvemenuer til disposition: en aktuel stregfarve fra en palette med 4 toner, en af 27 forskellige baggrundsfarver, samt alle tænkelige kombinationer af 4 til de to-farvede mønstre.

Der er endnu et stykke konfekt, vi ikke har omtalt. Det er ZOOM-funktionen. Man kan udvælge en bestemt del af et skærmbillede og forstørre det så meget at de enkelte pixels kan ændres. Dertil anvendes blyant-symbolet.

En specialitet er lasso-funktionen. Den svarer til den rektangulære ramme, men lassoen »fanger« ikke faste udsnit, men kun tegnede genstande af enhver form. Man kan altså tegne en cirkel, fange den med lassoen og flytte den rundt på skærmen, som man vil. Baggrunden bag genstanden forbliver uændret.

Kun kr. 498,-

INTERN

EN DATA BECKER BOG

BRÜCKMANN · ENGLISCH
GERITS · STEIGERS

AMSTRAD

464/664 & 6128

INTERN

Simpelthen bibelen til 464/664 & 6128

DANSK / NORSK

udgave

NORDIC COMPUTER SOFTWARE
SMEDEGADE · POSTBOX 105 · DK-6950 RINGKØBING
1986

Postbox 105
Smedegade 7
DK-6950 Ringkøbing

Sats & tryk: Tarm Bogtryk & Offset A/S

Dansk oversættelse & bearbejdning: Peter Friis & N.C.S.

ISBN 87-7283-007-7

VIGTIGT

Programmer, programeksempler mv. er omfattet af lov om copyright. Disse må kun anvendes til personlige- eller undervisningsformål og må ikke anvendes kommercielt.

Alle programeksempler, tekniske anvisninger osv. i denne bog er omhyggeligt gennemgået for fejl. Trods dette kan der optræde fejl i reproduktionsfasen.
Skulle nogle læsere opdage fejl, beder vi Dem rette henvendelse til os, så vi har mulighed for at foretage rettelse.

INDHOLDSFORTEGNELSE

1. INDLEDNING 9

1.1. HVAD DE BØR VIDE OM DERES CPC 10

1.1.1. Lagring 10
1.1.2. Kommandoudvidelser via RST 12

1.2. PROCESSOR Z80 13

1.2.1. Tilslutning af Z80 15
1.2.2. Opbygning af registre i Z80 17
1.2.3. Z80s specielle egenskaber i CPC 20

1.3. GATE-ARRAY, SYSTEM-KOORDINATOREN 22

1.3.1. Gate-array's tilslutninger 22
1.3.2. Gate-array's registeropbygning 26

1.4. VIDEO-CONTROLLEREN HD 6845 29

1.4.1. Tilslutning af CRTC 31
1.4.2. De interne registre i video-controller'en 32

1.5. RAM I CPC 34

1.5.1. De ekstra 64 K RAM i 6128 37

1.6. VIDEO-RAM MELLEM Z80 OG 6845 39

1.7. PARALLEL-INTERFACE KREDS 8255 42

1.7.1. 8255's tilslutninger 42
1.7.2. 8255's forskellige modes 44
1.7.3. Styring af 8255, registerbeskrivelse 45
1.7.4. 8255's opgave i CPC 47

1.8. TONEGENERATOREN AY-3-8912 49

1.8.1. Sound-chip'ens tilslutninger 51
1.8.2. De enkelte registres funktioner i 8912 53
1.8.3. Sådan fungerer AY-3-8912 i CPC 55

1.9. DISKETTESTATIONEN I CPC 664 OG 6128 57

1.9.1. FDC 765 58
1.9.2. Pin-belægningen på FDC 59
1.9.3. Brugen af FDC 765 i CPC 63

1.10. INTERFACES I CPC 64

1.10.1. Tastaturet 64
1.10.2. Videotilslutning 66

1.10.3. Floppytilslutning 66
1.10.4. Recorderen 67
1.10.5. Centronic-interface 70
1.10.6. Tilslutning af joystick 72
1.10.7. Expansion-connector 73

2. OPERATIVSYSTEMET 75
2.1. OPERATIVSYSTEMETS VEKTORER 75
2.1.1. Operativsystemets vektorer i CPC 664 76
2.1.2. Operativsystemets vektorer i CPC 6128 83
2.2. OPERATIVSYSTEMETS RAM 90
2.2.1. Operativsystemets RAM i CPC 664 90
2.2.2. Operativsystemets RAM i CPC 6128 92
2.3. UDNYTTELSE AF OPERATIVSYSTEMETS RUTINER . 94
2.4. INTERRUPTS I OPERATIVSYSTEMET 101
2.5. OPERATIVSYSTEMETS ROM 104
2.5.1. Kernel (KL) 104
2.5.2. Machine pack (MC) 116
2.5.3. Jump restore (JRE) 123
2.5.4. Screen pack (SCR) 129
2.5.5. Text screen (TXT) 136
2.5.6. Graphics screen (GRA) 148
2.5.7. Keyboard manager (KM) 154
2.5.8. Sound manager (SOUND) 162
2.5.9. Cassette manager (CAS) 165
2.5.10. Screen editor (EDIT) 171
2.6. KARAKTERGENERATOREN 176

3. BASIC 198
3.1. BASIC-FORTOLKEREN I CPC 664 & CPC 6128 198
3.2. BASIC STACK 200
3.3. BASIC OG MASKINSPROG 202
3.3.1. CALL-kommandoen 202
3.3.2. BASIC-udvidelser med RSX 203
3.3.3. Variabelpointeren "@" 205
3.4. BASIC ROM 207
3.4.1. Flydende komma aritmetik 207

4. TILLÆG 251
4.1. OPERATIVSYSTEMETS RUTINER 251
4.2. REFERENCER TIL SYSTEM-RAM 256
BASIC-TOKENS 261
MONITOR 264

1. INDLEDNING

Firmaerne AMSTRAD og CPCs forhold er højst usædvanligt. Næppe havde CPC 464 erobret markedet takket være fremragende kvalitet og lav pris, før firmaet udsendte CPC 664 som den næste computer på markedet. Og knap tre måneder senere så den tredie i CPC-rækken, CPC 6128, dagens lys. Begge CPC 464s efterfølgere faldt heldigt ud som følge af fremragende fabrikat og prisbillighed.

Systemets fuldkommenhed fremtræder endnu klarere ved de to sidste computeres fremkomst. Kiv og strid om Dallas eller sports-TV lægges på hylden takket være anskaffelsen af en farve- eller grønmonitor. De lumske fodfælder, forbindelseskablerne, hører fortiden til, ligesom den indbyggede floppy, der reducerer "kabelsalat" og den bekostelige lagerudvidelse eller interface-card, gør livet lettere.

Og hvad kan man ikke udrette...! LOCOMOTIVE-BASIC er slet og ret det bedste, der kan købes for penge. Især kan fremhæves, de meget fleksible og lige til at indsætte, programmerede interrupts, som denne BASIC indeholder.

Den fremragende grafik og muligheden for at fremkalde 80 tegn på billedskærmen, uden modul, er uovertruffen. Andre computere i denne prisklasse har ofte problemer med at fremstille læselig tekst uden flimren ved hjælp af 40 tegn.

Grafik-opløsningen på 640*200 punkter virker perfekt. Også sammenlignet med IBM-PC, som koster fra 5 til 8 gange mere end en CPC.

Også CPC'ens tonemuligheder er bemærkelsesværdige. Ganske vist er gengivelsen af stradivarius-klange, selv med den mest perfekte programmering, ikke mulig, men De har jo valgt en computer og ikke en violin.

Hvad hastigheden angår, behøver CPC'en ikke at skamme sig. Den indbyggede Z80-processor arbejder med en frekvens på 4 mhz og indeholder et overordentlig stort kommandoregister. Dette forråd blev der lagt stor vægt på af konstruktørerne, hvilket resulterede i en virkelig fornem BASIC-interpreter, som søger sin lige.

Men efter kortere eller længere tid (sandsynligvis kortere!) opstår ønsket hos næsten alle computerejere om mere information, mere viden om den computer, vedkommende ejer. Selv den bedste brugervejledning til CPC vil da ikke længere være nok, og man ønsker mulighed for at kunne overskride BASIC-grænserne med maskinsprog.

Den fra en-række-INTERN bøger kendte ROM-listing, bringer vi her i en ny, sammentrængt form. I stedet for at bringe den oprindelige listing, har vi valgt at bringe udførlige kommentarer. Ved hjælp af de, i denne bog bragte disassemblers kan De, når det ønskes, fremstille Deres egen listing. Fremtidig vil man ved hjælp af dette brugssystem kunne opnå en indsigt, som ellers ville sprænge en normal bogs dimensioner.

Deres forfattere.

1.1. HVAD DE BØR VIDE OM DERES CPC

Deres CPC indeholder ialt 6 højintegrerede ICer. Det vigtigste element i enhver computer, processoren, er en Z80 i CPC. Ydermere er indbygget en videokontrol, HD 6845, en parallelindgang, 8255, et tonechip AY-3-8912, floppykontrol 765 og et for CPC specielt konstrueret gate-array.

Videokontrollen har til opgave, at stille alle nødvendige signaler til rådighed for monitorens anvendelse. Den adresserer også lagerbeholdningen af de der værende tegn og grafik til skærmbilledets RAM. Samtidig fremstiller den det nødvendige refresh, uden hvilken disse informationer hurtigt ville gå tabt.

Tonechippens opgave fremgår af dens navn. Konstruktørerne har truffet et storartet valg. AY-3-8912 er indsat i mange computere, fordi den frembyder mangesidig og vidtrækkende indflydelsesmuligheder for lyden.

8255 kan man kalde "trækdyret" i CPC. Den løser mange opgaver. Rækkende fra kontrol af tastaturet, over styring af tonechippen, til styring af recorderen og fastlæggelse af diverse muligheder i CPC.

I forbindelse med flere TTL-ICer og en såkaldt data-separator danner floppykontrollen 765 interface for de maximalt to floppy-drivværker og letter ved hjælp af sin høje "intelligens" derved programmeringen.

Gate-array er af særlig interesse. Denne chip styrer så mange funktioner i CPC, at man kunne tillægge den status som hjælpeprocessor. Den er simpelthen i stand til at overtage mange af billedskærmens opgaver. Herunder de forskellige farver og linieformater, ligesom de nødvendige intervaller fremstilles i gate-array. Interrupteren, som afbryder det normale programforløb 300 gange i sekundet, bliver, som også signalet for styringen af RAM i CPC, frembragt af gate-array.

1.1.1. LAGRING

For bare fem år siden, anså man computere med 16 K RAM for ganske veludstyrede. Men efter fremkomsten af C64 er lagerkapaciteten øget væsentligt. En computerfabrikant kan kun påregne sig markedsandele, hvis hans produkt viser "det magiske 64". Da prisen på RAM imidlertid er faldet drastisk på det sidste, spiller det ingen rolle, om man udstyrer computeren med 64 Kbytes (som i 664) eller med 128 Kbytes (som i 6128).

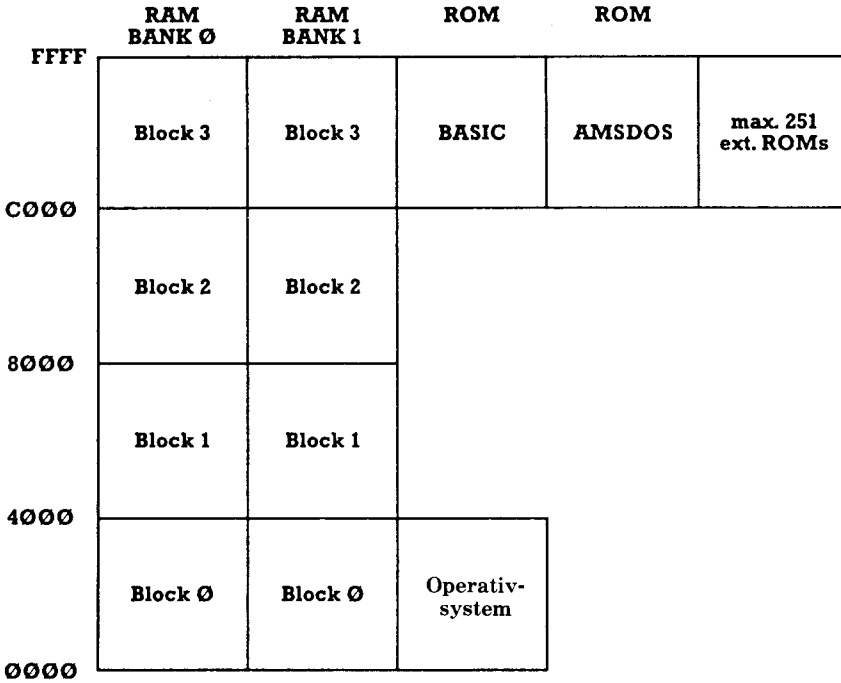
Det er ikke særlig svært at indbygge 64 K-lager i en computer, da alle 8-bits-processorer umiddelbart kan adressere dette. Også CPCs Z80 klarer det uden tricks. Da ikke blot RAM, men også ROM bør adresseres, opstår imidlertid visse vanskeligheder, der dog er undgået med stor elegance i CPC. Et mindstemål af hardware og yderst udsøgt software yder den i CPC indbyggede bank-switching, med hvilken man kan veksle mellem RAM og ROM efter behag. Også styringen af 64 Kbyte RAM i 6128 er løst ved hjælp af bank-switching.

Med hensyn til CPC viser sig følgende forhold: Gennemgående vil 64K RAM adresseres. Dertil "parallelt", dog ligger 64 Kbyte-banken ligeledes i de nederste l6 K, som udgør halvdelen af 32 K-ROM. Den anden halvdel af 32 K-ROM ligger i de øverste 16 K.

De nederste 16K ROM indeholder hovedsagelig computerens arbejdssystem inclusive rutinen for styringen af recorderen og lagerudvekslingen. I operativsystemet finder man alle nødvendige rutiner for CPC, f.eks. et tegn til læsning af tastaturet, et tegn eller et grafikpunkt, som bringes på skærmen, men også recorderen og printerporten samt lyd betjenes af operativsystemet.

I de øverste 16 K finder vi Basic-interpreteren. Parallelt med dette område ligger ROM for AMSDOS, som indeholder alle rutiner, der styrer floppyen. Det er imidlertid ikke nok - man kan yderligere i området tilslutte andre ROMs til 251. I denne ROM kan man f.eks. indføre andre programmeringssprog, BASIC-udvidelser og diverse spil.

Man kan vise lageret grafisk som vist på skemaet 1.1.1.1.



1.1.1.1 Hukommelsesopdeling i CPC.

1.1.2. KOMMANDOUDVIDELSER VIA RST

I den hensigt at få ROMmens forskellige muligheder udnyttet bedst muligt, har programmørerne benyttet sig af et elegant lille trick. Ved hjælp af særlige programmer og den elegante udnyttelse af RESTART-ordrerne i Z80 finder vi en udvidelse af mulighederne for restarts fra RST1 og til RST5. Disse RSts lader sig anvende på lige fod med de øvrige JMPs eller CALLs. Ved visse RSTs kræves imidlertid en 3-byte-adresse. I den tilføjede 3. byte bestemmes, i hvilken ROM, JMP eller CALL skal indgå.

LOW JUMP RST 1.

Kommandoen kan kalde en rutine i operativsystemet eller den underliggende RAM. Adressen på den ønskede rutine må stå direkte bag RST-ordren. Det er nødvendigt med 14 adressebits, for at området mellem 0 og &3FFF kan nåes, hvorfor man benytter begge øverste bits i udvælgelsen af ROM eller RAM:

Bit 14 = 0 Arbejdssystemet vælges
Bit 14 = 1 RAM vælges
Bit 15 = 0 BASIC-ROM vælges
Bit 15 = 1 RAM vælges

En ordre til systemrutinen &1410 kan se ud som følger:

RST 1

DW &1410 + &8000

Via den indsatte bit 15 udvælges området mellem &C000 og &FFFF RAM, mens operativsystemet kaldes ved den slettede bit 14.

Koden i adresse 8000 består ene og alene i sin egenskab af et spring til &B98A.

SIDE CALL RST 2

Denne restart-ordre bruges til at kalde en rutine i ekspansions-ROM. C-ordren bruges så, når et program, der foreligger som ROM-modul, behøver mere end 16 Kbyte og hvor der ikke er yderligere plads i ekspansions-ROM. Man kan således kalde en rutine i 2., 3. eller 4. tilhørende ROM, uden at man behøver at kende det absolutte nummer i den aktuelle ROM. Efter RST 2-ordren må rutinen -&C000, altså den omtrentlige adresse m.h.t. start af ROM, benyttes.

I adressen &0010 findes et spring til &BA1D.

FAR CALL RST 3

Ved hjælp af RST-ordren kan man kalde en rutine, hvor som helst i ROM eller RAM. Ydermere må der stå en parameter-blok, hvis 2-byteadresse som består af 3 bytes, bagved RST 3-ordren. Disse 2 første bytes inderholder rutinens adresse, som skal kaldes, og den tredie byte skal indeholde den ønskede ROM/RAM-status. Derved bliver værdierne fra 0 til 251 i den omtalte ekstra ROM kaldet. De resterende 4 værdier har følgende funktion:

Værdi	&0000-&3FFFF	&C000-&FFFF
252	Operativsystem	BASIC
253	RAM	BASIC
254	Operativsystem	RAM
255	RAM	RAM

RAM LAM RST 4

Ved hjælp af denne RST-ordre, kan De læse indholdet af RAM fra maskinprogrammet, uafhængig af den valgte ROM-status. RST 4-ordren overtager derved ordren.

LD A,(HL)

Dertil må HL indeholde adressen på lagercellen. I adressen &0020 findes et spring til &BAD6.

FIRM JUMPRST 5

Med RST-ordren kan man springe til en rutine i operativsystemet. Adressen til RST 5-ordren må følge umiddelbart efter. Operativsystemets ROM bliver "enabled" før rutinen forsøges kaldt igen, og bliver ved returnering atter "disabled". I adressen &0028 findes et spring til &BA35.

1.2. PROCESSOR Z80

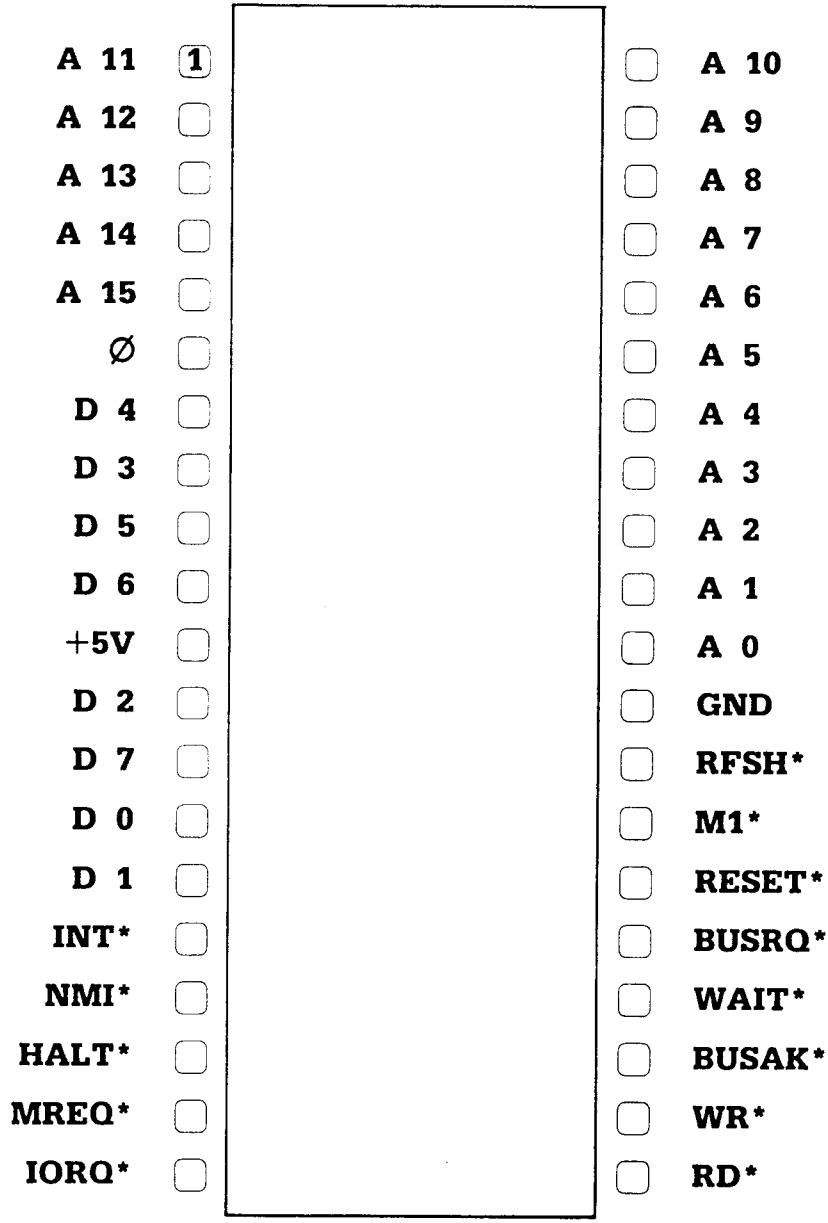
I begyndelsen af 70erne indledtes mikro-processorernes sejrsgang.Firmaet INTEL fik med processoren 8080 en betydelig markedsandel, da der på den tid ingen konkurrence var i denne klasse. Det gør sig i høj grad gældende, når processorens kapacitet underkastes en nøjere vurdering. Således kræver 8080 yderligere 3 forskellige driftsspændinger og dertil 2 ICer til brug for styringssignalet og interval-frembringelse.

I årene 74/75 blev Z80 udviklet af firmaet ZILOG. I stedet for at udvikle en ny processor fra grunden, holdt man sig til det så godt indarbejdede koncept fra 8080. Derfor kan Z80 og 8080 supplere hinanden, d.v.s. alle 8080s programmer kan også anvendes i en Z80 processor. Imidlertid blev alle de mindre heldige egenskaber, som 8080 havde, rettet, og kommandoregisteret blev stærkt udvidet. Dertil kommer, at Z80 kun behøver 1 driftsspænding på + 5 volt og eksterne ICer er ikke nødvendige til styring af signalet.

Vi kan i telegramstil læse processorens dataydelser, før vi beskæftiger os nærmere med dens egenskaber.

Enkelt strømforsyning 5 volt
Simpelt interval
TTL-kompatibel
Henholdsvis 2.5., 4, 6 eller endog 8 mhz frekvens
Softwarekompatibel med 8080
Dobbelt registersæt, dertil 2 indeksregistre

Ikke skjult interruptindgang med 3 anvendelsesmuligheder
Selvopretholdende refresh af dynamisk RAM
8080-periferi-ICs kan tilsluttes direkte



1.2.1.1 Pinbelægning Z80.

Disse dataydelser og den store mængde af færdig software har gjort Z80 til en af de mest succesrige 8-bits-procesorer.

Indenfor hjemme- og personlige computere har kun en videreudviklet processor - 6502 fundet en lignende udbredelse.

1.2.1. TILSLUTNING AF Z80

Efter disse korte oplysninger om særlige kendetegn for driften, skal vi se på funktionerne for de 40 pins i Z80.

Tilslutningen til Z80 lader sig sammenfatte i 4 grupper, databus, adressebus, styrebus og forsyningsveje.

ADRESSEBUS

A0-A15: ADRESSELINIER
Ved hjælp af disse tilslutninger bliver en lagercelle udvalgt i adresseområdet. Adresseområdet omfatter 65536 lagerpladser. Ved brug af I/O-ordrerne benyttes de nederste 8 adresse-bits til at angive den omtalte I/O-adresse. Derved er 256 forskellige adresserings muligheder. Med visse indskrænkninger m.h.t. ordresætningen kan man endog adressere 65536 indgange. På den måde bliver alle 16 adresselinier bragt frem til dannelse af indgangsadressen. Vi vil senere vende tilbage til disse specielle forhold.

DATABUS

D0-D7: DATALINIER
Ved hjælp af 2-vejsledninger overføres data fra og til processoren. De fremstiller forbindelse mellem processoren og den gennem adressebussen udvalgte lagercelle eller indgangsadresse.

STYREBUS

MI*: MACHINE CYCLE ONE
Styresignalet viser, at processoren læser operationskoden fra databussen. Iøvrigt betyder stjernen, at det ved disse og de følgende signaler drejer sig om low-aktive signaler.

MREQ*: MEMORY REQuest*
Udgangssignalet viser via low, at processoren foretager en læse- eller skriveoperation i lageradressen, og om adresseringen i adressebussen er valid.

IORQ*: INPUT/OUTPUT ReQuest*
Denne udgangs low viser, at processoren foretager en skrive- eller læseoperation på en indgangsadresse, og at indgangsadressen til adressebussen er valid.

RD*: ReaD*

Udgangssignalet er low, hvis processoren skal læse data fra en lagercelle eller en indgangsadresse. Ved kombination med MREQ* og IORQ* kan der skelnes mellem læsning fra lager eller indgang.

WR*: WRite*

Z80-signalet bliver til low, når Z80 under testning, når data lades fra Z80 i lageret eller indgangsadressen fremtræder i valid form. Også her kan der skelnes mellem, om data skal indføres i lager eller i port-adresse gennem sammenføjning af WR*, MREQ* og IORQ*.

RESET*:

Lægges denne indgang på low, bliver Program-Counter ladet med værdien &0000. Interrupt bliver spærret, og interruptmodus 0 startes op. Så snart indgangen atter bliver high, starter processoren programmet fra adresse &0000.

NMI*: NON MASKABLE INTERRUPTS*

Ved hjælp af en high-low-side i denne indgang vil processoren stadig afbrydes i det kørende program. Program-Counter vil blive ladet med de i adressen &0066 og &0067 indeholdte værdier og herfra vil programmet fortsætte.

IRQ*: INTERRUPT ReQuest*

Gennem en low i indgangen kan processoren afbrydes i det kørende program, når den form for interrupt frigives efter ordre. Resultaterne adskiller sig efter den valgte interruptmodus og vil senere blive gennemgået. IRQ* fremstiller i modsætning til NMI* et statisk signal, som skal være i overensstemmelse med interruptkravet.

WAIT*:

Ved hjælp af signalet kan læse- eller skriveordrer fra Z80 til et langsommere lager, eller under særlige betingelser, tilpasses systemet.

BUSRQ*: BUSReQuest*

Bliver indgangen low, så vil ved oparbejdning af de løbende ordrer adresse- og datastyring samt alle udgangsordrer blive high og BUSAK*-signalet bliver low. Nu kan en anden processor overtage styringen af lageret og perifere enheder. Signalet vil hovedsagelig stå til rådighed for DMA (DMA=direkt memory accesses, betydeligt hurtigere datatransmission ved at omgå processoren).

BUSAK*: BUSAKnowledge*

BUSAK* fremstiller sammen med BUSRQ* korresponderende udgangssignaler.En low meddeler DMA-kontrollen, eller den anden processor, at alle styre- og bussignaler er high, og at der nu følger et indgreb.

HALT*:

Udgangen bliver low efter at processoren har udført maskinsprogskommandoen HALT.Efter denne ordre foretager processoren sig intet videre. Den gennemfører NOP, hvorved refresh vedligeholdes. Kun en interrupt kan "vække" den igen.

RFSH*: ReFreSH*

Udgangssignalet viser, at der på de 7 nederste adresser ligger en gyldig refreshadresse. Da processoren kun har brug for adresse- og databussen til bestemte tider, kan adressebussen, i den mellemliggende tid, bruges til at opfriske de dynamiske RAMs, uden at man behøver at anvende eletronik eller særlige opfriskningsrutiner.

INTERVAL OG STRØMFORSYNING

0: Phi

Indgangen til phi leverer processorens frekvens. Da Z80 er en statisk IC, kan frekvenserne fra 0 hertz til det maksimale anvendes. Imidlertid stilles der bestemte krav til frekvenssignalets form. I overensstemelse med dataside, må den maksimale low-tid højst andrage 2 mikrosekunder. Denne værdi er imidlertid mest af akademisk interesse, da man vil bestræbe sig på at forsyne processoren med den højst mulige frekvens, for hurtigst muligt at få programmet oparbejdet.

GND:

ALMINDELIG TILSLUTNING AF PROCESSOREN.

Vcc:

Gennem tilslutningen får Z80 sin energi, nemlig +5 volt jævnstrøm og ca. 150-200 milliampere.

1.2.2. OPBYGNING AF REGISTRE I Z80

Som allerede omtalt er Z80 konstrueret sådan, at 8080-programmer uden videre kan overtages. Under alle omstændigheder er antallet af registre i Z80 betydelig højere.

Men hvad er egentlig et register?

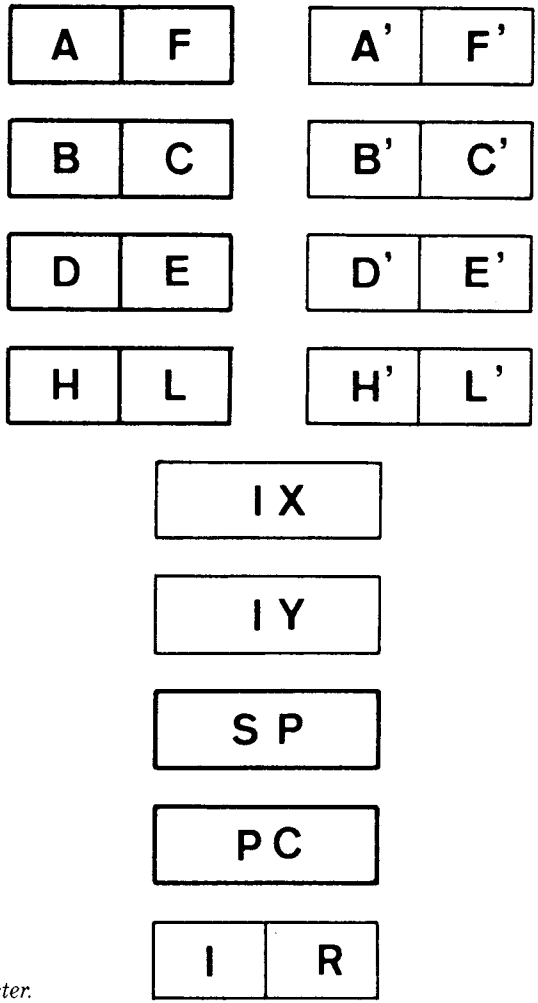
Et register er blot et skrive/læselager i processor-chippen. Enhver processor må indeholde et mindstemål af registre. I denne lagercelle bliver data lagret, samt resultaterne af de aritmetiske og logiske ordrer. Andre registre har særlige opgaver, såsom forvaltningen af stack og anvendelsen som Program-Counter.

Da operationerne som f.eks. overføring af data mellem 2 registre eller addition af 2 registerindhold, ikke bliver afviklet over databussen, kan en sådan operation gennemføres meget hurtigere, end hvis de nødvendige værdier måtte hentes fra en ydre lagerplads.

Som tommelfingerregel kan det siges, at processoren med mange registre, er det interne lager overlegent ved kørsel af samme program, da dataoverføring indenfor processoren altid er hurtigere, end en overføring fra og til ydre lagerpladser.

Alt ialt råder Z80 over 22 registre. Atten registre med 8 bits og 4 med 16 bits. Opdelingen fremgår af grafikken 1.2.2.1.

I grafikken fremhæves nogle registre med en kraftigere indramning. Disse registre rummes også i 8080. Det er påfaldende, at de fleste 8-bits-registre forekommer dobbelt. Det er registrene A, F, B, C, D, E, H og L. Z80 giver mulighed for dobbeltudførelse, og programmøren kan gennem ordrer vælge mellem begge sets.



1.2.2.1 Z80 register.

I det følgende vil vi kun tale om eet registersæt. Det er i CPC for så vidt rigtigt, da der kun står et enkelt registersæt til rådighed for programmøren. Det alternative registersæt benyttes af operationssystemet til interruptstyring. Læg imidlertid mærke til, at alle et registersæts opgaver også kan overtages af det alternative registersæt, når det ikke bruges til specielle formål.

Registrene B til L stiller i almindelighed 8-bits-registre til rådighed, mens registrene A og F pålægges særlige opgaver.

A-registret betegnes i almindelighed som akku eller akkumulator. I akku får man resultatet af alle aritmetiske og logiske operationer i 8-bit-format. I akku må også lagres en operand under denne operation. Skal man f. eks. addere 2 bytes, er det nødvendigt at lagre en operand i akku, den anden operand kan hentes i et andet register eller i lageret udenfor processoren. Efter additionen findes resultatet i akku.

Da det korrekte resultat i sådanne opgaver kan blive så stort, at det ikke mere kan udtrykkes med kun 8 bits, er det nødvendigt at tilføje endnu en bit. Denne opgave overtages af F-registret. F-registret, kaldet flag-registret, er opdelt i enkelte bits. En af disse bits skal opbevare en sådan addition. Andre bits viser, om resultatet af regneoperationerne er lig nul osv.

Registrene B til L kan ikke blot "tiltales" hver for sig. Ligesom B og C, D og E samt H og I også kan sammenfattes i 16-bits-registre. Disse dobbeltregistre har selvfølgelig derfor fået navne efter de to enkeltregistre, såsom BC, DE og HL. Dobbeltregistrene egner sig fortrinligt til adressering af tabeller og til transport og gennemsøgning af datablokke. HL-dobbeltregistret får yderligere en særlig betydning. Da Z80 gennem ordrerne addition eller subtraktion bliver forsynet med 16-bits-værdier fungerer HL ved sådanne ordrer som 16-bits-akku.

PC, SP, IX og IY arbejder kun med 16-bits-værdier.

PC er Program-Counter (PC). Indholdet af PC bliver indført som adresse fra adressebussen i det ydre lager. Men med enhver ordre bliver PC automatisk inkrementeret (forhøjet med een). Ved ordrer med mere end en byte bliver PC automatisk forhøjet med det nødvendige antal. Er det nødvendigt med spring i programmet, bliver den nye programadresse automatisk indført i PC, og processoren arbejder videre fra den adresse.

SP er en såkaldt stackpointer. Stacken er nødvendig, når man vil kalde et underprogram fra programmet. I så fald bliver returadressen automatisk lagt på stack, og ført tilbage efter at underprogrammet er afsluttet. De to 16-bits-registre IX og IY muliggør gennem særlige ordrer et specielt virkningsfuldt arbejde med tabeller.

Og nu skal registrene I og R omtales. I-registret eller interrupt-registret anvendes i forbindelse med den særlige interrupt-operationsmåde IM3. I den modus må den, for det af interrupt fremstillede element, efter ordre fra processoren yde en 8-bits-værdi. Værdier som low-byte og I-registerindhold danner adressen interrupt-rutinen. R- eller refreshregistret er nødvendig i forbindelse med den af Z80 automatisk gennemførte refresh. Efter ethvert fremkald af en befaling bliver de nederste 7 bits i dette register automatisk inkrementeret. Den 8. bit vedbliver efter enhver programmering at være 0 eller 1. Hverken I- eller R-registrene anvendes i CPC. Da der ikke kan fremkaldes noget udsagn om tilstanden i R-registret, og disses værdier stadig ændres, kan registrene bruges som tilfældige generatorer.

1.2.3. Z80s SPECIELLE EGENSKABER I CPC

De mangfoldige muligheder i Z80 giver hard- og softwaredesigneren frie hænder til konstruktion af en computer. Denne CPU (central-processing-unit) kan ligeledes indsættes effektivt i minimalsystemer og sådanne ydelsesdygtige instrumenter som CPC-computeren.

Konstruktørerne af CPC har grebet dybt i trickposen i den hensigt, at opnå et maksimum af ydelse, med et minimum af elementer. Derved er der frembragt nogle konsekvente detaljer, som det er vigtigt at vide besked med, når effektiv programmering skal opnås, ligesom udbyttet af dataanlægget er særlig vigtigt, når det drejer sig om maskinsprog. Disse specialiteter vil vi tage under luppen.

Først interruptstyringen af CPC.

Gate-array er en enestående interrupt-kilde i CPC, hvis fantastiske element styrer en meget vigtig del af computerens ydeevne. Indenfor 3,3 millisekunder, altså 300 gange i sekundet, frembringer gate-array en kort impuls og anbringer den i Z80s IRQ*-indgang. NMI*-indgangen i processoren er ikke anvendt og står til rådighed for eventuelle udvidelser i ekspansionsconnector.

Interruptsignalets frekvens frembringes ved H-sync-signalet fra CRTC 6845 gennem en frekvensdeler. Modulet deler alle gennem H-sync-impuls fremkomne ca. 65 microsekunder via 52.

Da Z80 drives af interrupt-modus IMI i CPC, fremkalder ethvert interrupt IRQ et RST7 eller et CALL &0038. Processoren afbryder straks det løbende program, lægger den aktuelle tilstand i PCen på stablen og reducerer til adressen 10038. Her står der nu i CPC et spring til adressen vedr. den egentlige interruptrutine. Det sted, hvor afbrydelsen har fundet sted, bliver mærket istack. Sådan kan man efter at have afslutte interruptrutinen, fortsætte det afbrudte program.

Da IRQ*-indgangen i processoren også ligger i ekspansionsconnectoren, stiller man naturligvis det spørgsmål, hvordan kan man skelne mellem et interrupt fra gate-array og et eksternt interrupt. Her har konstruktøren anvendt et specielt trick. I et kort øjeblik bliver interrupt igen tilladt indenfor interrupt-rutinen. Da impulsen fra gate-array kun er max. 5 mikrosekunder, har tilladelsen ingen virkning, impulsens virkning er forlængst borte. Ydre interruptkilder genoptager først deres signaler efter udtrykkelig anvisning fra processoren. Foreligger der en sådan ydre interrupt , bliver interruptrutinen altså selv afbrudt. Men dette tilfælde kan ses og behandles specielt. Dermed er også ydre IRQ*-kilder mulige. Det eneste krav til den er en tilstrækkelig lang impuls.

Det andet specialtilfælde, som man bør være opmærksom på, er den nedsatte mulighed for anvendelse af Port-kommandoer.

I forbindelse med signalet IORQ* kan Z80 højst adressere 256 forskellige primæradresser i overensstemmelse med lagerkapaciteten. Således bliver den ønskede Port lagt på de nederste 8 adresse-bits A0 til A7. Disse Port-indgange bliver hovedsagelig benyttet til tilslutningen af perifere-kredse.

M.h.t. andre processorer, hvor mulighed for Port-adressering ikke kendes, er konstruktøren altid henvist til adressering af perifere kredse som lagerpladser. Denne fremgangsmåde kalder man memory-mapped, og den har den ulempe, at det adresseområde, der står til rådighed for RAM, bliver mindre.

Med hensyn til brug af Port-adresseringer, stiller Z80 den meget dynamiske gruppe af IN- og OUT- ordrer til rådighed. Ser man nærmere på ordrerne fra denne gruppe, så finder man gennem ordren IM (C), r og OUT (C), r en elegant mulighed for at adressere mere end de 265 oprindelige indgange. Gennem ordrenbestemmes tilstanden af de 8 nederste adressebits i C-registrets indhold, samtidig med at indholdet af B lægges på adresse-bits A8 til A15. Herved står samtlige 65536 Port-adresser til disposition.

Konstruktørerne af CPC har forstået at udnytte netop denne egenskab. Alle periferi-ICer udvælges ved hjælp af adresse-bits A8 til A15.

Ved sådanne tricks er der desværre ofte en ulempe. I dette tilfælde består ulempen af en begrænsning af kommandosættet i Z80. Alle øvrige I/O-ordrer i Z80 kan ikke mere indsættes. Dette gælder især for I/O-ordrerne med sløjfeautomatik. Disse benyttes B-registret som tæller og står derfor ikke til rådighed som leverandør af Port-adressens high-bytes. Dette gælder især ordrerne INI, INIR, IND og INDR samt OUTI, OTIR, OUTD og OTDR.

Som det tredie, der kendetegner CPC, er anvendelsen af wait-cyklen.

Nødvendigheden af denne tilslutning stammer helt fra dengang, da de til rådighed stående lager-ICs var nogle lunefulde fyre. Især de første EPROMs lod vente et mikrosekund fra markering af adressen og indtil data var parat.

For at bruge Z80 med disse "efternølere" var det nødvendigt at vente en tid. Denne ventetid kan frembringes med signalet WAIT*. Efter hver negativ flanke i interval-indgangen efterprøver processoren tilstanden i WAIT*-tilslutningen. Har denne tilslutning spændingen 0 volt, så tilføjer Z80 en såkaldt wait-cyklus fra intervallets længde. Efter intervallets udløb, altså med negativ flanke, bliver tilstanden i wait-ledningen kontrolleret osv.

Det i CPC anvendte signal ligger imidlertid ikke i den anvendte lager-IC. Det er imidlertid hurtigt nok for en Z80 med 4 mhz. Årsagen findes i den nødvendige synkronisering af processor og video-controller. Da begge ICs kan kaldes fra lageret, må der være en kontrol, som til enhver tid findes i rækken. Herved har video-controlleren sit ubestridelige fortrin, da billedet på monitoren ellers vil blive stærkt forstyrret. For at opnå denne synkronisering bliver der frembragt et wait*-signal for hvert fjerde intervalsignal. Skønt processoren styres med 4 mhz, frembringes en effektiv arbejdsfrekvens på ca. 3,3 mhz ved hjælp af wait-cyklen.

Signalerne BUSRQ* og BUSAK*, styresignalerne for DMA-funktionen, er ikke benyttet i CPC. Alligevel er de indført i ekspansions-connectoren og står til rådighed for ekstern udvidelse.

Signalet HALT* skal heller ikke anvendes i CPC, men er dog alligevel til rådighed i ekspansions-connectoren.

1.3. GATE-ARRAY, SYSTEM-KOORDINATOREN

Så godt som alle CPCs chips er i handelen. De sælges af enhver velassorteret elektronikforhandler. Undtaget er ROM og gate-array. I dette afsnit skal vi beskæftige os med den sidst omtalte IC.

Denne 40-bens IC er udviklet specielt til CPC og har flere vigtige opgaver. Ønsker man at efterligne alle integrerede funktioner ved hjælp af TTl-gitteret, ville antallet af IC's i CPC mere end fordobles.

Gate-array's opgaver er bl.a.:

Fremstilling af alle nødvendige interval-frekvenser
Fremstilling af signal til RAMs dynamiske funktion
Styring af indgreb i RAM
Ud- og indkobling af ROM i lageret
Fremstilling af videosignalet
Fremstilling af RGB-informationer til farvemonitor
Styring af skærm-modes
Lagring af farvevalg
Fremstilling af interrupt-impulser

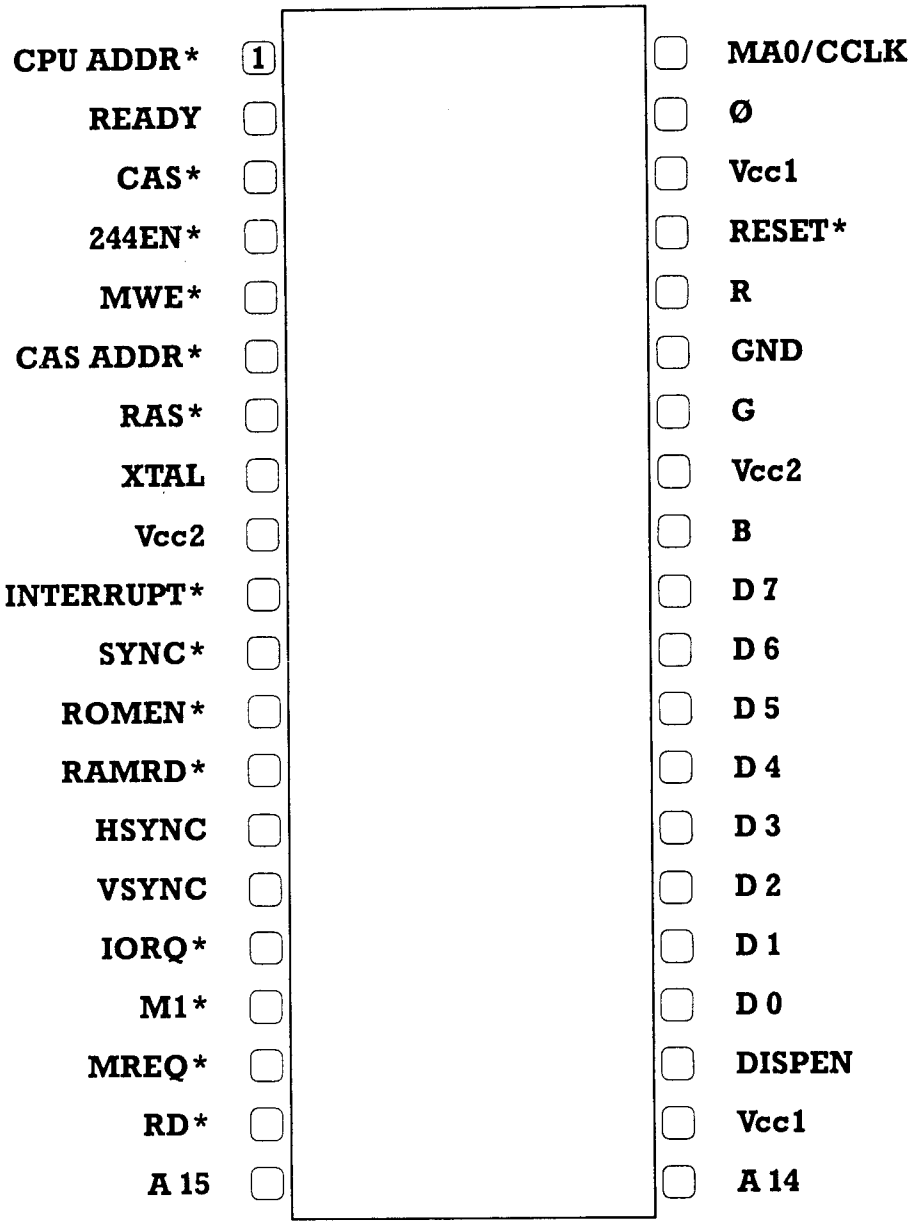
Desværre står der ikke megen information til rådighed m.h.t. disse interessante ICs. Et data-ark eller lignende beskrivelser er så godt som ikke til at få, idet udgiveren sikkert betragter sådanne oplysninger som fabrikationshemmeligheder.

Vore anstrengelser og forsøg på så grundigt som muligt at forske i ICs funktioner resulterede imidlertid i succes, og vi vil ikke skjule vor viden for læseren.

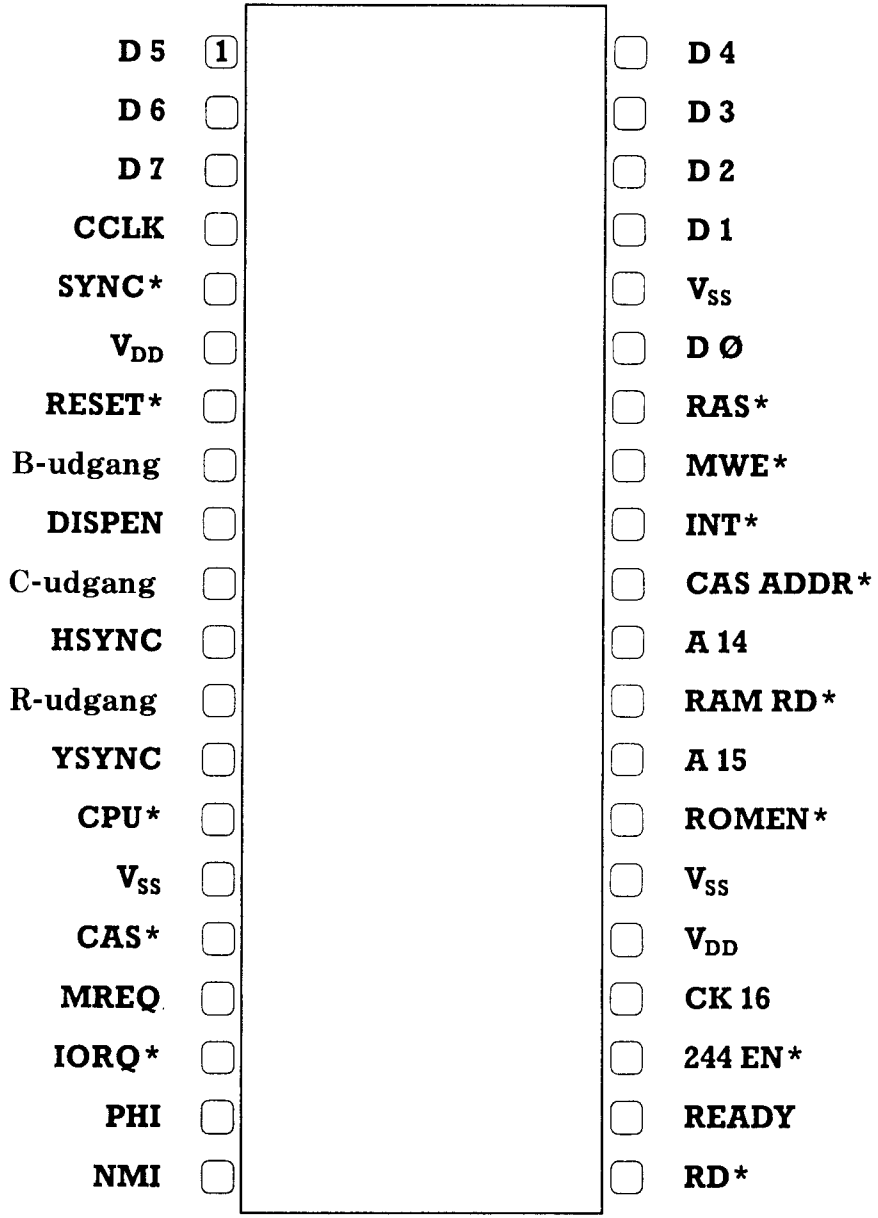
1.3.1. GATE-ARRAY'S TILSLUTNINGER

Før vi kan beskæftige os med de enkelte tilslutningsfunktioner af gate-array, må vi komme med en oplysning. Der eksisterer mindst 3 versioner af gate-array. I CPC 464, den første CPC, havde IC betegnelsen 40007. Denne ICs blev imidlertid under brugen så varm, at det blev nødvendigt at afkøle den ved hjælp af en aluminiumsskærm. Dette forhold var ikke praktisk i det lange løb, da IC'en selv med denne afkøling blev overophedet. I CPC 664 indsattes derfor IC 40008. Gennem ændring i den indre opbygning kunne energitabet reduceres. Den udgave blev kun i ringe grad overophedet. Sluttelig indsattes IC 40010 i CPC 6128. Ved hjælp af den blev rækkefølgen af tilslutningerne ændret. Efter valg kan også den ældre udgave af gate-array indsættes i 6168. Den nødvendige plads på printet er til rådighed.

I beskrivelserne af tilslutningerne, har vi holdt os til versionen, der findes i CPC 6128. De i parenteserne anførte tilslutningsnumre angiver de oprindeligt indsatte versioner af GA i 664 og 464. Det signal, som bestemmer det hele i CPC, er det i pin 24 (pin 8) (XTAL) liggende signal med en frekvens på 16 mhz. Denne frekvens frembringes af en med to gitre opbygget oscilator, og udgør herved CPCs hjerteslag.



1.3.1.1 Pinbelægning for Gate Array 40007 og 40008.



1.3.1.2 Pinbelægning for Gate Array 40010.

Den med 4 delte frekvens udgør intervalsignalet på 4 mhz til processoren, og findes som takt phi i pin 19 (pin 39).

Efter deling med 4 endnu engang fås en frekvens på 1 mhz. Signalet findes via pin 14 (pin 1) fra gate-array.

1-mhz-signalet har to funktioner. For det første er det periodesignalet for lyd-chippen, dernæst bestemmer det, om processoren eller CRTC adresseres i RAM. Ved hjælp af low bliver processorens adressefremføring sluttet til RAM gennem multiplexer-ICs 74 LS153.

Da styringen af RAM i CPC ikke er uden faldgruber, bringer vi en udførlig beskrivelse af RAM-styresignalet i et senere kapitel.

Da RAM-chipsene kun råder over 8 adressetilslutninger, må samtlige 16-bitsadresser multiplexes, altså lægges tidsmæsigt efter hinanden (lægges på indgangene). Denne tidsstyring opnås med signalerne CAS ADDR* pin 31 (pin 6), CAS* pin 16 (pin 3) og RAS* pin 34 (pin 7). Signalerne RAS* og CAS* lægges direkte i RAM, signalet CAS ADDR* føres til den allerede nævnte multiplex.

Signalet MAO/CCLK fra gate-array pin 4 (pin 40) har en frekvens på 1 mhz. Det signal er ganske vist faseforskudt i forhold til CPU ADDR*-signalet, hvilket vil sige, at de to frekvenser ikke er high samtidig. MAO/CCLK har ligeledes en dobbelt funktion. Den udgør først og fremmest taktsignalet for CRTC, som afleder alle andre signaler, og desuden lægges den som hjælpeadresse-bit på en af de fire adressemultiplex'er. Denne hjælpeadressebits funktion skal ligeledes omtales senere sammen med styringen af RAM.

Endvidere bliver signalet RAMRD* i pin 29 (13) fremstillet af gate-array. Tilslutningen bliver low, når processoren efter input af adresse, skal læse data fra RAM, og gennem RD*-signalet meddeler sig til array ved pin 21 (19). Da ROM og RAM stort set overlapper hinanden, kan RD*-signalet fra processoren ikke anvendes direkte. Vil man læse data fra ROM, så bliver signalet RAMRD* high, og udgangene fra DATA og LATCH/BUFFERs 74LS3373 bliver høj-ohmsk. Derved kan, indtil da, ikke opnås nogen information fra RAM til databussen, skønt lageradressen ligger i RAM, og at den stiller en byte til rådighed for udgangen.

Foruden RAMRD* bliver READY-signalet fra pin 22 (pin 2) i GA lagt i IC 74LS373. Dette signal fremstiller signalet til indføjelse af wait-cyklen i processoren. Ved hjælp af den etablerede kontakt mellem READY og latch/buffer opnås, at informationen i processor-databussen ikke ændres under wait-cyklen. 74LS373 lagrer efter tilslutning af en high i pin 11 de aktuelle udgangsinformationer, hvorved low atter fremkommer. Derefter forholder IC sig som en enkelt buffer, d.v.s., at udgangene efterfølger indgangsændringerne umiddelbart.

Signalet ROMEN* i pin 27 (12) i GA bliver low, når processoren skal udlæse data fra ROM. Det i CPC indbyggede 32k-BASIC- og funktionssystem- ROM optager adresseområdet fra &0000 til &3FFF og fra &C000 til &FFFF. Man kan altså henvende sig til det i to uafhængige halvdele. Om det øvre lagerområde skal læses fra ROM eller RAM, må meddeles GA gennem en OUT-ordre. Derved er det muligt kun at aktivere een halvdel i ROM.

I overensstemmelse med den ønskede lager-konfiguration dekoder GA adresseledningerne A14 og A15's tilstand. Gennem det ønskede lager bliver RAMRD* eller ROMEN* aktiveret.

En skriveordre i processoren går altid til RAM uafhængig af den valgte lager-konfiguration. Dertil bruges signalet MWE* fra GA.

Udover den beskrevne funktion bliver adresseoverførelserne A14 og A15 anvendt til et andet formål i pin 18 (10)og 30 (21). Også GA har en Port-adresse, som benyttes til at programmere de forskellige muligheder i GA. Port-adressen er &7F00 og dekodes over adresseoverførelserne (A14 high, A15 low) og signalet IORQ* i pin 17 (pin 18).

Da databussen i Z80 ikke er direkte forbundet med GAs dataoverførelser D0 til D7, ligger array på tilslutningen 244EN* til low, når indgangsadressen &7F00 erkendes på den tidligere beskrevne måde. Derved frigives udgangene 74LS244 (en databus-buffer), og det fra Z80 leverede byte kan skrives i array.

Også signalet I0RQ* har en dobbelt betydning for GA. Specielt for Z80 gælder det, at en identificeret interrupt lægger signalet IORQ* og MI* samtidig i low. Denne tilstand noteres af GA og interruptimpulsen frigøres straks. Hvis kommandoen DI, disable-interrupt, derimod har afbrudt behandlingen af IRQ, så bliver tilslutning 32 (10) i GA, som efterlader IRQ som low. Så snart IRQ atter er tilsluttet EI, bliver den tilhørende interrupt identificeret, og interruptudgangen bliver atter high.

Interruptsignalet udgøres af en programmerbar delekæde i GA og fremkommer ved pin 32 (pin 10). Delekæden forsynes med CRTC-signalet HSYNC og deler frekvensen med 52. Da HSYNC-impulsen optræder hvert 65. mikrosekund, går der 3,3 millisekunder mellem to interruptimpulser. Her kobles impulserne sammen med CRTC's VSYNC-signal. VSYNC's bredde er i CRTC programmeret til at være på ca 500 mikrosekunder. Efter ca. 125 mikrosekunder optræder der interrupt, hvorved interrupt-rutinen har ca. 375 mikrosekunder til at undersøge, om der ved port-bit 0 i 8255's Port B findes en VSYNC. Signalet anvendes som tidtager for visse operationer. Tilfældet optræder dog kun ved hvert 15. interrupt, hvor de øvrige 14 tests erfarer en High-tilstand. Således vil den interne tæller ikke blive påvirket.

Signalerne HSYNC og VSYNC samt DISPEN skal selvfølgelig bruges til opbygning af video-signalet. En logisk kombination af signalerne udgør SYNC*-signalet på Pin 5 (Pin 11) i GA.

1.3.2. GATE-ARRAY's REGISTEROPBYGNING

For at kunne udføre alle de beskrevne opgaver, skal data være lagret i Gate-array. Det nøjagtige antal interne registre er ikke kendt, men vi kan beskrive de formodentligt vigtigste registres funktioner.
Som det er tilfældet for de øvrige kredse i CPC, så accesseres Gate-array via Port-adresseringen.

Den belagte adresse er &7Fxx. Heraf fremgår det, at adressebit A15 er Low, og A14 må være High. De øvrige adressebits (A12 til A8) skal være satte, da de øvrige kredse decodes på en lignende ufuldstændig måde. Kredsenes selektions-porte er ligeledes kun forbundet via enkelte adressebits.

Den nederste adressebyte's tilstand har ikke betydning for dekodningen. Her kan være anført en vilkårlig værdi.

Der kan ialt skelnes imellem 3 forskellige registre.

De to første registre hører sammen med farvevalg, d.v.s. de farvetilordninger, som er gældende på baggrund af PEN og INK.

Det første register lades med den adresse, hvori farveværdien skal skrives. I det følgende betegnes register med FN-reg. Den egentlige farveværdi kan derefter skrives i det andet register (under samme Port-adresse) og vil i det følgende være betegnet som FW-reg.

Det 3. register er et multifunktionsregister (MF-reg) og bestemmer skærmmode og hukommelseskonfiguration. Her bestemmes udvalget af muligheder via de enkelte bits i registeret.

I Gate-array's registre kan der KUN SKRIVES. Det er IKKE muligt at udlæse de enkelte værdier.

Da Gate-array kun kan kommunikeres via en enkelt Port-adresse, må der være en metode til at kunne skelne mellem de forskellige grupper på. Forskellen fremkommer i de 2 øverste bits i byten. De mulige kombinationer lyder:

Bit 7	Bit 6	Funktion
0	0	Skriv værdien i FN-reg
0	1	Skriv farveværdien i det valgte FW-reg
1	0	Skriv værdien i MF-reg
0	0	Bruges til Bank-switching i 6128

Men, hvad har det nu at gøre med farvenummer- og farveværdi-registre? De to registre svarer egentlig til Basic-kommandoerne PEN og INK. Pen-kommandoen bruges som bekendt til at ændre skriftfarven på skærmen. Tilordningen af et PEN-nummer til en farve kan bestemmes med INK-kommandoen. Nummeret, der skal ændres og den ønskede farveværdi angives. Det er netop den funktion, der udføres af de to registre. I FN-registeret lagres nummeret på farven, der skal ændres og herefter kan den ønskede farveværdi skrives i Gate-array. Ønsker man f.eks. at ændre den til PEN 1 hørende farve, er følgende linie nødvendig:

OUT &7F00,&X00000001:0UT &7F00,&X010xxxxx

I den første OUT-ordre er bits 6 og 7 lig nul. I bits nul til 3 angives nummeret på den farve, der skal ændres. I vort eksempel er det nr. 1. Bit 5 har ingen funktioner, bit 4 har en speciel betydning, som vi straks vender tilbage til.

I OUT-ordre nr. 2 er bits 6 og 7 valgt således, at FW-registret udvælges. Den bit, som kaldes "X", bestemmer nu farveværdien. Med 5 bits er 32 forskellige farver mulige, men kun 27 forskellige farver kan kaldes. De resterende 5 farver er alle dubletter.

Efterprøver De dette eksempel i Basic, vil De se, at den ønskede effekt ikke umiddelbart opnås. Et kort glimt af den nye farve, er alt, hvad der viser sig.

Det skyldes CPC-firmwaren's egenart. I princippet fremstår alle farver "blinkende". Det bemærkes ikke, da der ikke afbrydes mellem forskellige, men kun mellem ens farver. Ved hver farveafbrydelse omlades alle parametre i GA. Indtaster De komma. doen "SPEED INK 255,255", så kan De efter nogle forsøg, opnå en effekt af betydelig længere varighed.

Men nu forklaringen vedr. den, indtil nu, "glemte" bit 4 i FN-reg. Er den bit blevet sat ved brug af registeret, så overses informationen i bits 0 til 3, og den i næste OUT-ordre overførte farveværdi, fortolkes som ny rammefarve.

MF-registret adresseres, når bit 7 sættes, og bit 6 er low i OUT-ordren. De øvrige bits i registeret har følgende betydning:

Bit 5 : Denne bit har ingen funktion?
Bit 4 : 1 = V-Sync- tælleren slettes
Bit 3 : 1 = ROM &C000 til &FFFF afbrydes
Bit 2 : 1 = ROM &0000 til &3FFF afbrydes
Bit 1 : Skærmmode
Bit 0 : Skærmmode

M.h.t.funktionen af bit 5 i dette register, har der indtil nu ikke kunnet gives oplysninger.

Er bit 4 sat, så slettes interrupt-impulsens delearray, og V-Sync-impulsens tælleværk begynder forfra. På den måde kan tidsafstanden mellem 2 interrupt-impulser forlænges. I Basic kan funktionen vises ved hjælp af den følgende lille programsløjfe:

```
10 OUT &7F00 , &X10010110 : GOTO 10
```

Ved start er computeren helt blokeret. Selv en reset via SHIFT/CTRL/ESC kan ikke gennemføres. I dette et-linie-program slettes tælle-registret så hurtigt, at ingen interrupt-impulser kan forekomme. Da tastaturet imidlertid kun spørges i interrupt, må man bide i det sure æble, slukke for computeren og starte forfra.

Bits 2 og 3 bestemmer den øjeblikkelige ROM-konfiguration. Er en bit sat, så befinder RAM sig i det for processoren angivne adresseområde. Er bits slettet,så læser processoren data fra ROM. At manipulere disse to bits planløst, fører mindst til fejlmeldinger, i værste fald til systemsammenbrud og efterfølgende reset.

De resterende bits 0 og 1 fastsætter den aktuelle skærm-mode. De mulige kombinationer er:

Bit 1	Bit 0	
0	0	mode 0, 20 tegn/linie, 16 farver
0	1	mode 1, 40 tegn/linie, 4 farver
1	0	mode 2, 80 tegn/linie, 2 farver
1	1	som mode 0, men ingen, blink

Når De har prøvet et-linie-programmet, til hindring af interrupt i mode 1, vil De have observeret en ejendommelig ændring af tegnene på skærmen. Vi har valgt 80-tegnsmode som skærm-mode, som eksempel for dette, uden at slette skærmen. De fremkomne tegn ser ud, som om de mangler punkter i midten. Forklaringen på det fænomen finder De i forbindelse med næste kapitel, hvor skærmbilledets opbygning og fremstillingen af tegnene vil blive gennemgået.

1.4. VIDEO-CONTROLLEREN HD 6845

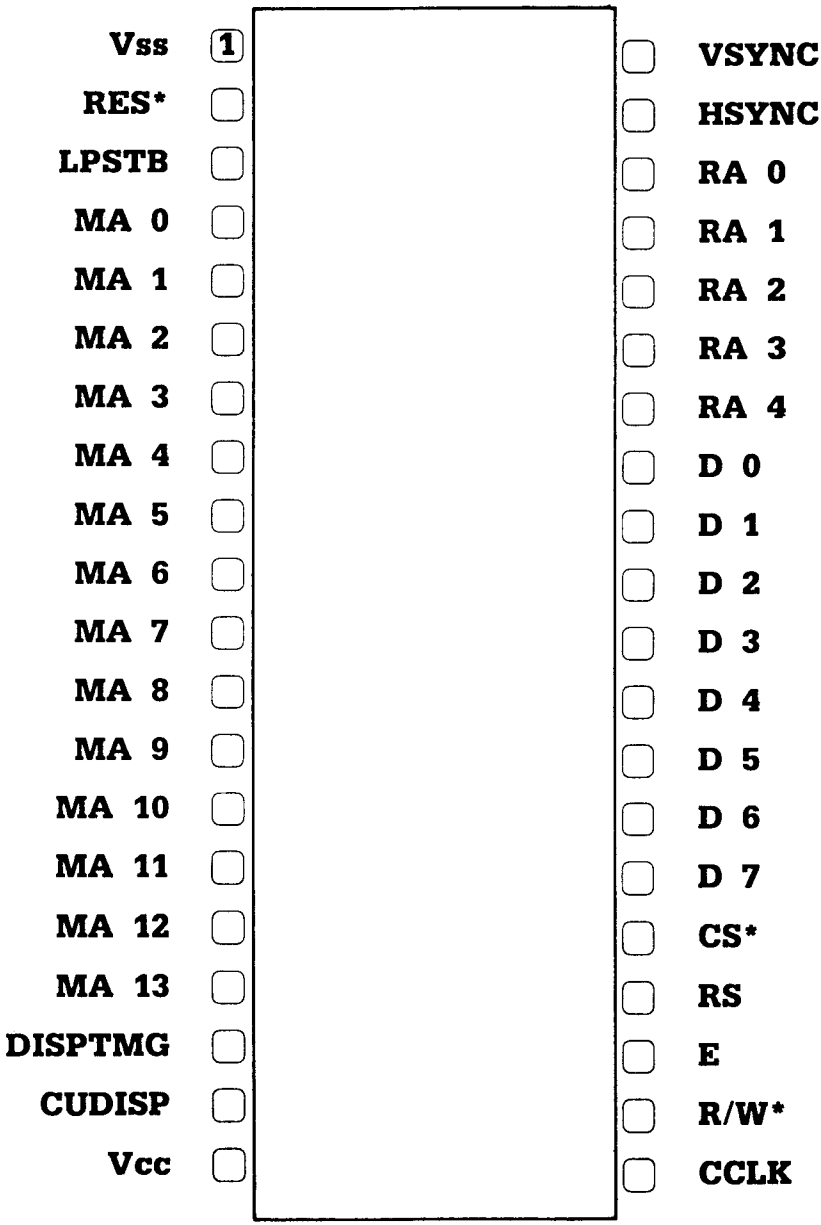
Hovedansvaret for opbygning af skærmbilledet er overgivet til Video-Controller HD 6845, der også kaldes Cathode Ray Tube Controller, forkortet til CRTC. Denne kreds er skabt specielt til at virke som interface mellem microprocessorer og rasterskærme, såsom standard monitors.

Den frembringer gennem et enkelt taktsignal de for monitoren nødvendige syncronsignaler, hvorved alle nødvendige parametre lader sig programmere.

Inden vi beskriver pin-belægningen og de interne registeropbygninger, vil vi give et kort overblik over mulighederne i denne interessante kreds:

Programmerbart antal tegn pr. linie
Programmerbart antal linier på skærmen
Programmerbar vertikal punktmatrix i tegn
Brug af et adresseområde på 16 K
Automatisk refresh ved brug af dynamisk RAM
Cursor-kontrolfunktioner
Programmerbar cursor (højde og blink)
Light-Pen-Port
Simpel 5 volt driftspænding
TTL-kompatible ind- og udgange

Oprindelig blev 6845 udviklet af MOTOROLA m. henblik på indsættelse i computersystemer udviklet med processor-familien 68xx. Som følge af den usædvanlige fleksibilitet og enkle brug, finder man controlleren i mange systemer. Selv i så effektive systemer som f.eks. Sirius, finder vi controlleren.



1.4.1.1 Pinbelægning for CRTC HD 6845.

1.4.1. TILSLUTNING AF CRTC

De 40 Pins's betydning er følgende:

MA0-13 :Memory adress lines
Gennem 14 tilslutninger adresseres memory i skærmbilledets RAM.

RA0-4 :Raster adress lines
Fra karaktergeneratoren vælger 5 forbindelser, den til tegnet hørende rasterlinie.

D0 - 7 :Bidirectional databus
Gennem disse pins læses og skrives informationer i controlleren.

R/W* :Read/write*
Signalet bestemmer retningen af dataoverførelsen. Ved et low signal kan data overføres til CRTC fra processoren, og ved high læses de fra CRTC.

CS* :Chip select*
For at muliggøre dataoverførelse ved hjælp af 6845 må denne adresseres. Det foretages via et low på CS*-indgangen.

RS :Register select
Signalet er nødvendigt for at vælge mellem adresseregistret og 18 kontrolregistre. Ved hjælp af low i RS kan man gribe ind i adresseregistret, ved et high gribes ind i kontrolregisteret.

EN :Enable
Ved dette signals fremkomst bliver de hos IC'en liggende processorsignaler overtaget af controlleren.

RES* :RESET*
Et low i indgangen nulstiller alle tællere i CRTC, og alle udgange sættes til low.
Funktionen udføres kun, hvis LPSTB-indgangen samtidig er low. Denne reset sletter ikke kontrolregistret!

CLK :Character Clock
Er taktsignalet, der ved deling styrer alle de for monitoren nødvendige signaler.

HSYNC :Horizontal sync
Leverer signalet til den horisontale synkronisering i monitoren. Forkert indstilling eller manglende HSYNC ytres ved at billedet vælter.

VSYNC :Vertical sync
Leverer det nødvendige signal for vertikal synkronisering.

DISPTMG :Display timing

Signalet er high, når det til monitoren givne signal skal vises på skærmen. Ved hjælp af signalet undertrykker man tilbageløb af strålen.

CUREN :Cursor enable

(Ofte kaldet cursor-display, CURDISP), anvendes, når cursoren ikke styres gennem software men af CRCC selv. Også cursor-blink kan styres gennem den tilslutning.

LPSTB :Light pen strobe

Hvis der lægges en low-high-side på denne indgang, bliver den aktuelle tilstand af MA-portene overdraget til, og lagret i, Light-Pen-Register. Registeret kan udlæses og anvendes i et tilsvarende program.

1.4.2. DE INTERNE REGISTRE I VIDEO-CONTROLLER'EN

Som allerede nævnt, rummer 6845 et adresseregister og 18 kontrolregistre. Da signalet RS (register-select) kun kan vælge mellem 2 adresser, opstår spørgsmålet om, hvordan alle 18 kontrolregistre kan kommunikeres gennem blot een adresse.
Svaret finder vi i adresseregisteret. I adresseregisteret indføres det kontrolregisters nummer, som man skal bruge som næste gang. Den fremgangsmåde synes meget omstændelig, men indebærer en ubestridelig fordel. Ved metoden optager CRTC nemlig kun 2 og ikke 18 eller sågar 32 adresser. Da CRTC desuden normalt kun programmeres en enkelt gang ved opstart af computeren, får man altså mere plads med i købet!.

Men lad os betragte de 18 registre nøjere. Den følgende beskrivelse kan, som følge af enkelte registres komplicerede struktur, forekomme noget tør og vanskelig tilgængelig. Et grundlæggende kendskab til teknikken er imidlertid nødvendig. Skulle De ikke forstå det hele, så trøst Dem med, at Video-Controlleren i Deres computer ikke skal programmeres "ved håndkraft".

I den følgende fremstilling betyder et R efter registerbetegnelsen, at registret kan læses, et W betyder, at det er muligt at skrive i registeret. Vær opmærksom på, at nogle registre enten kun kan læses, eller skrives (kendetegnet med et -).

AR -/W ADRESS REGISTER

5-bit-registeret lades med nummeret på det ønskede kontrolregister. Registerværdier 18 til 31 ignoreres, de gyldige værdier er 0 til 17. Registeret kommunikeres når både CS og RS er low.

R0 -/W HORIZONTAL TOTAL

I dette 8-bit-register indføres antallet af tegn pr. hel linie. Imidlertid er en komplet linie væsentlig længere end de synlige tegn på skærmen, da tiden for kant og strålens returløb skal medregnes. Tilsvarende vælges værdien ca. 1,5 gange så stor som antallet af egentlige tegn.

R1 -/W HORIZONTAL DISPLAYED

Registeret indeholder antallet af de tegn, der skal være synlige på skærmen. Den her indførte værdi skal være mindre end værdien af R0.

R2 -/W HORIZONTAL SYNC POSITION

8-bit værdien i registeret bestemmer tidspunktet for HSync-impulsen. Formindskes værdien af R2, forskydes monitorbilledet mod højre, en forhøjelse forskyder billedet mod venstre.

R3 -/W SYNC WIDTH

Bredden af HSync og WSync-impulserne fastlægges af de fire nederste bits i dette register. De øverste 4 bits i registret benyttes ikke.

R4 -/W VERTICAL TOTAL

De nederste 7 bits i registeret bestemmer antallet af rasterlinier i billedet. Værdien bestemmer også ved, om et skærm-refresh skal være på 50 eller 60 Hz.

R5 -/W VERTICAL TOTAL ADJUST

Ved hjælp af de nederste 6 bits i registeret, kan man foretage en finjustering af skærm-refreshfrekvensen.

R6 -/W VERTICAL DISPLAYED

De nederste 7 bits i registeret bestemmer det egentlige antal af rasterlinier, som vises på skærmen. Her kan der teoretisk indsættes enhver værdi mindre end (R4).

R7 -/W VERTICAL SYNC POSITION

7-bit-værdien i registeret bestemmer tidspunktet for VSync-impulsen. Formindskes værdien af R7, forskydes monitorbilledet nedefter, en forhøjelse forskyder billedet opad.

R8 -/W INTERLACE

Ved hjælp af de to nederste bits i registeret bestemmes, om udlæsningen skal efterfølges med eller uden line-feed (interlace).

R9 -/W MAXIMUM RASTER ADRESS

5-bit-registeret bestemmer antallet af rasterlinier i det tegn, der skal udlæses.

R10 -/W CURSOR START RASTER

Registerets bits 0 til 4 bestemmer hvilken rasterlinie cursoren skal begynde i. Bits 5 og 6 bestemmer cursor-mode. Cursor-mode bestemmes med følgende bits:

Bits 6	5	
0	0	cursor blinker ikke
0	1	cursor vises ikke
1	0	cursor blinker (ca. 3 gange pr. sek./50 hz.)
1	1	cursor blinker (ca. 1,5 gange pr. sek./50 hz)

R11 -/W CURSOR END RASTER

Som ved (R10) fastlægger de nederste 5 bits i registeret, hvilken rasterlinie cursoren standser i.

R12 R/W START ADRESS HIGH

Bits 0 til 5 fastlægger fra hvilken adresse i hele 16k-adresse-området i CRTC skærmhukommelsen starter. Læses dette register, er bits 6 og 7 altid low.

R13 R/W START ADRESS LOW

Registeret fastlægger ligesom (R12) den mindstbetydende adressebyte i den adresserede skærmhukommelse.

R14 R/W CURSOR HIGH

Bits 0 til 5 i registeret udgør den øjeblikkelige cursorpositions high-byte.

R15 R/W CURSOR LOW

Som ved (R14) bliver cursor-adressens low-byte lagt i dette register.
Da såvel R14 som R15 kan skrives og læses, kan man frit bestemme cursorpositionen via registeret.

R16 R/-

Efter en positiv strobeimpuls indeholder registeret den high-byte, som svarer til den aktive skærmhukommelsesadresse for tidspunktet for impulsen. Bits 6 og 7 i dette register er altid low.

R17 R/-

Som ved R16 indeholder registeret low-byte på tidspunktet for Light-Pen-stroben. Såvel R16 som R17 kan kun læses.

1.5. RAM I CPC

Den i CPC indbyggede skriv- og læsbare hukommelse (RAM) bruges ikke kun som data- og programlager. Også skærmbilledets data lægges her.

Efter at vi i de tidligere kapitler har omtalt de tre vigtigste IC-kredse i CPC, processoren, Gate-array og Videocontroller i detaljer, kaster vi i dette afsnit et blik på sammenspillet mellem disse tre komponenter under brug af hukommelsen. Her forklares, hvordan Videocontrolleren aktiverer RAM for at få udlæst tegn på skærmen. Vi vil også beskrive adresseringen af de i CPC 6128 indbyggede ekstra 64 Kbyte udførligt.

Men inden, da vil vi gøre en lille afstikker for at se på, hvordan de anvendte RAM-kredse med deres 8 Adress-Pins overhovedet fungerer.

Først må vi have klarlagt, hvordan det er muligt at adressere 65536 hukommelses-celler med de forhåndenværende 8 adressetilslutninger.

Den grundlæggende metode er at dele 16-bits-adressen i to halvdele og lægge de to adresse-bytes efter hinanden på RAM's adresse-Pins. Fremgangsmåden hedder, at multiplexe. Imidlertid kræver metoden, at der er et styresignal til at fortælle RAM, hvilke informationer, der ligger på tilslutningerne.

På dette tidspunkt kommer de fra Gate-array leverede signaler RAS* og CAS* ind i billedet. Når en adressebyte findes i RAM, får RAM gennem et High-Low-skift af RAS* at vide, at der ligger en adressehalvdel parat. På den måde lagres de tilstedeværende adresseinformationer i RAM.

Nu kan den anden halvdel af adressen flyttes til RAM. Straks, når denne adresse-byte foreligger, bliver CAS*-signalet low. Dermed har RAM modtaget hele 16-bit-adressen og den ønskede lagercelle kan findes og udlæses, eller få et nyt indhold.

Skift mellem adressehalvdelene skal selvfølgelig styres af et passende signal, og i CPC hedder signalet CAS-ADDR*.

Multiplex (SWITCH) udgøres af 4 IC's af typen 74LS153. Disse IC-funktioner kan man bedst beskrive, som to elektroniske drejeomskiftere. Begge omskiftere har 4 indgange og 1 udgang. Gennem 2 styreindgange kan man bestemme, hvilken af de 4 indgange, der er forbundet med udgangen.

Begge styreindgange styres af signalerne CPU-ADDR* og CAS-ADDR*. Gennem signalet CPU-ADDR* bestemmes det, om processoren eller CRTC kan sende en adresse til RAM. CAS-ADDR* skifter mellem de aktuelle adressehalvdele.

Følgende tabel viser den specifikke tilordning af adressetilslutninger fra processoren og Video-Controller:

Z80	6845	Z80	6845
A0	CCLK	A8	MA7
A1	MA0	A9	MA8
A2	MA1	A10	MA9
A3	MA2	A11	RA0
A4	MA3	A12	RA1
A5	MA4	A13	RA2
A6	MA5	NA14	MA12
A7	MA6	NA15	MA13

Som det ses, ligger alle processorens adressebits i RAM via multiplex. Imidlertid ligger adressesignalerne A14 og A15 ikke direkte på multiplex i CPC 6128. Her er kredsen til Bank-Switching tilsluttet. Men også Video-Controlleren adresserer ved hjælp af CCLK det samlede 64K-lager. Det står i modsætning til forrige kapitel, hvor vi jo, sagde at CRTC kun kan adressere et 16K område.

Det udsagn er for så vidt rigtigt, hvis kun de 14 med MA (memory adress line) betegnede tilslutninger tælles med.

Disse 14 tilslutninger gør det muligt at adressere et område på 16 K.

Den i CPC anvendte brug af 6845 til adressering af Video-RAM ses ikke oftest.Med tilslutningerne RA0 til RA4 styres normalt en fastprogrammeret tegn- eller karakter-ROM, som indeholder bit-mønstret, for de viste tegn på skærmen.

Normalt har computeren et lagerområde, der betegnes Video-RAM, i hvilket de tegn, der vises på skærmen, lagres. I lageret optager enhver tegnposition 1 byte. Det kræver ved udlæsning af 80 * 25 tegn et lager på 2000 bytes.

Man kan imidlertid give den nødvendige information ved hjælp af 1 byte. Ethvert tegn består jo af punkt-rækker, der ligger under hinanden.

Også på CPC's monitor kan man se disse rækker. Således består f.eks. cursoren af 8 rækker under hinanden, i hvilke alle billedpunkter er "tændte". Ved udlæsning af bogstaver eller tal er kun de punkter tændte, som kræves for at man kan genkende tegnet. Dette punktmønster kan lagres ved hjælp af et bitmønster, hvorved alle satte bits er synlige på skærmen.

RA-tilslutningerne er nødvendige for at få de enkelte rækker, altså bitmønstre, udlæst fra karakter-ROM. RA-tilslutningerne anvendes til adressering af ROM.

Som man kan forestille sig, er det ved anvendelse af fastprogrammerede Karakter-ROM ikke muligt at vise højopløselig grafik på skærmen. Computere konstrueret efter dette princip, er afhængige af det faste tegnsæt.

I CPC er den oprindelige karakter-ROM bortfaldet. Man er gået andre veje.

Da RA-tilslutningerne adresserer direkte i hukommelsen, skal bit-informationerne ligeledes være at finde i RAM. Kun med den metode er det muligt at skabe alle tænkelige bitmønstre på skærmen.

Men før vi beskæftiger os med Video-RAM's konkrete opbygning, må signalet CCLK forklares. Dertil er det imidlertid nødvendigt med lidt matematik.

CRTC styres med en taktfrekvens på 1 Mhz. Ved hver taktimpuls adresseres en lagercelle. I den celle findes bit-vis de informationer, som skal danne tegnene på skærmen, hvor de satte bits ses som punkter i den aktuelle Pen-farve.Da en frekvens på 1 Mhz svarer til en periode på et microsekund, står der nøjagtig en ottendedel af taktfrekvensen til rådighed for fremstillingen af ethvert punkt. For at fremstille alle 640 punkter i en linie, kræves der en tid på 80 microsekunder.

Da varigheden af V-sync til en linie er 52 microsekunder, går regnskabet ikke op. Med disse værdier kan der maksimalt udlæses 40 tegn.

Problemet løses ved hjælp af en speciel funktion i RAM, kaldet Page-Adress-Mode. Hvis en RAM efter udførte RAS- og CAS-signaler har anbragt indholdet af en ønsket lagercelle på dataudgangene, så er det tilstrækkeligt at lægge en ny adressehalvdel i RAM ved hjælp af en ny CAS-impuls, for at få den næste byte. Det forudsætter naturligvis, at kun halvdelen af en adresseinformation er ændret.

Konstruktørerne af CPC har udnyttet netop denne egenskab. Adresseinformationen til de to CAS-impulser skal naturligvis være forskellige, da man ellers læser den samme

adresses indhold to gange. CCLK-signalet skelner nøjagtigt mellem de 2 CAS-impulser. Multiplex lægger signalet på adressebit 0 (set fra processoren), når signalet CAS-ADDR er low, og signalet CPU-ADDR er high. Det udgør dermed nederste adressebit i Video-RAM.

De hurtigt efter hinanden afleverede bytes, fra Video-RAM, bliver mellemlagret i Gate-array, og konverteret til den for monitoren nødvendige serielle form, og afleveres ved RGB-udgangen sammen med farveindformationerne.

Tilbage bliver de to signaler MA12 og MA13. Ved hjælp af disse bits bestemmes begyndelsen af Video-RAM i 16K-blokke. Sædvanligvis er disse bits alle satte, og Video-RAM begynder derfor ved &C000. Men et videoområde på &4000 til &7FFF er også muligt gennem tilsvarende programmering.

1.5.1. DE EKSTRA 64 K RAM I 6128

Forklaringen på sammenhængen af lagerombytningerne i 6128 var noget indviklet. Med enkle PEEK's og POKE's kunne problemet imidlertid ikke have været løst. Nu mangler kun det med computeren leverede program "BANKMAN". Da programmet af ubegribelige årsager er beskyttet, besværliggøres tingene noget. Men med en hel del gå-på-mod og eksperimentering, opnår man dog at kende de mest grundlæggende funktioner i Bank-proceduren.

Før vi går over til at beskrive lagerombytning, må 2 begreber forklares. En Bank betegner et stykke hukommelse på 64 Kbyte, hvorimod en blok er på 16 Kbyte. I det følgende afsnit vil de to betegnelser blive anvendt hyppigt.

Hukommelsens opbygning styres af en PAL-kreds af typen HAL16L8. På den kreds benyttes dataledningerne D0 til D2 samt D6 og D7 og adressesignalerne A14 og A14, signalet IOWR*, signalet CAS samt RESET og CPU* . Som udgange står signalerne NA14, NA15, CAS0 og CAS1 til rådighed. PAL-kredsen selv belægger indgangsadressen &H7Fxx, nøjagtig som Gate-array. Gennem beskrivelsen af Gate-array vil De vide, at registerudvælgelsen i GA foretages ved hjælp af tilstanden af databits D6 og D7. Kombinationen, hvor begge bits er 1 (high), vælger ikke noget register i GA. I stedet behandles informationerne i databits D0 til D2 i PAL. Denne information ændrer RAM-konfigurationen.

Efter en RESET-impuls opfører computeren sig, som om den kun har en 64 Kbyte-Bank indbygget. Adresseringssignalerne A14 og A15 fra Z80 overføres uden modifikation til Adress-Multiplex via PAL. CAS-signalet fra GA anbringes via PAL til pin CAS0. CAS1-signalet er indtil videre inaktivt. Dermed aktiveres første bank i CPC ved Memory-Acces. Refresh er iøvrigt også indstillet den anden Bank, da kun RAS-signalet er nødvendigt hertil. Signalet ligger parallelt på begge Banks.

Når en passende værdi meddeles PAL, så ændres lagerforholdene i CPC klart. Men lad os engang overveje, hvilke værdier, der kan komme på tale. Port-adressen var jo allerede klar. Endvidere ved vi, at databits D6 og D7 skal være satte for ikke at kommunikere med forkerte registre i GA. Databits D3 til D5 testes ikke, da de ikke er forbundet med PAL. Dermed har vi indkredset de mulige værdier. Kun værdierne &C0 til &C7 er mulige. Men hvilken funktion har de så?

Det er desværre vanskeligt at finde en forklaring til alle kombinationer p.g.a. CPC opbygning. Næsten hele hukommelsen skiftes samtidigt ud. Efter ombytningen er også det program, der stod for omskiftningen, forsvundet. Følgen bliver en klassisk systemophængning. Vi kan dog forklare de væsentligste kombinationer. Ved disse værdier bliver lageret i adresseområdet fra &4000 til &7FFF i Bank 0 ombyttet med en blok i Bank 1. De nødvendige værdier finder De i følgende tabel:

&C0	BANK 0, BLOK 1	DEFAULT-STATUS
&C4	BANK 1, BLOK 0	
&C5	BANK 1, BLOK 1	
&C6	BANK 1, BLOK 2	
&C7	BANK 1, BLOK 3	

Bliver en af værdierne mellem &C4 og &C7 udlæst på Port-adressen &7Fxx, bliver CAS0-signalet inaktivt i området &4000 til &7FFF. I stedet for aktiveres CAS1-signalet. Ligeledes modificeres informationerne i Adress-pins A14 og A15 i Z80 via PAL. Undtaget er værdien &C5, som påvirker det samme adresseområde i den anden Bank. Til &C4 f.eks. vil adresseområdet fra &0000 til &3FFF i den anden Bank blive adresseret, uden at processoren "opdager noget". Såvidt processoren ved, befinder RAM sig fortsat i det ønskede adresseområde. Tilsvarende gælder for værdien &C6 adresseområdet fra &8000 til &BFFF. For &C7 gælder området fra &C000 til &FFFF i den anden Bank.

Desværre kunne den eksakte betydning af værdierne &C1 til &C3 ikke opklares. Disse værdier spiller sikkert en stor rolle for CP/M 3.0, da ingen TPA med 61 Kbyte kan realiseres af de andre værdier. Imidlertid kan vi røbe hukommelsesbelægningen under CP/M 3.0 for interesserede læsere.

I dette operativsystem skal man håndtere 3 parallelt liggende RAM-områder. Selvfølgelig skal De ikke selv foretage RAM-skift. Det klarer CP/M jo for Dem.

I alle tre Banks er adresseområdet fra &C000 til &FFFF identiske. Dette område vil aldrig selv blive udskiftet, da skift mellem de andre områder foregår via dette. I adresseområdet ligger den øverste del af TPA og de altid residente dele af BIOS og BDOS faste.

I Bank 0 befinder der sig yderligere tre blokke. Den første blok (&0000 til &3FFF) indeholder den nederste Jump-blok, der straks bliver kopieret fra ROM efter opstart af CPC og flyttet til nederste RAM. I Bank 0's blok 1 (&4000 til &7FFF) finder vi skærm-RAM. I blok 2 befinder størstedelen af BIOS og BDOS sig, såvel som de nødvendige Jump-Blokcs, som stod i vejen for en udvidelse af TPA under CP/M 2.2.

Bank 2 består af blokkene 0 til 2 og indeholder størstedelen af TPA. Den endnu manglende del af TPA finder vi i denne Bank's blok 3. Denne blok er fælles for alle tre Banks. Endelig er området fra &4000 til &7FFF i Bank 2 belagt. I det område er CCP og de for CP/M nødvendige Hash-tabeller anbragt.

Ønsker De, at eksperimentere med de, i dette kapitel, ikke oplyste værdier for lagerombytning, så læg Dem følgende tips på sinde. Først må De flytte skærmlageret fra &C000 til &4000. Derefter skal De lægge Deres testprogram i det frigjorte område fra &C000, og starte det op. Bedst vil det være med et lille Monitor-program, som ligger i

området. Det beror på, at området sandsynligvis ikke bliver afbrudt, hvilket kan forekomme i de andre områder.Monitor-programmet skal før kald af systemrutiner flyttes via Jump-Blocks i konfigurationen til default (oprindelig konfiguration). Altså må værdien &C0 skrives til den tilsvarende Port-adresse. Glemmer De denne procedure, kan det ske, at Jump-Blocks ikke er til stede. Så hænger computeren op!

1.6. VIDEO-RAM MELLEM Z80 OG 6845

Prøv dette korte program på Deres CPC:

```
10 MODE 2
20 FOR I=&C000 TO &FFFF
30 POKE I,255
40 NEXT I
```

På skærmen vil der fremkomme en smal linie, som tegnes fra øverste venstre hjørne mod højre. Ved slutningen af den første linie vil den fortsætte nøjagtigt 8 linier længere nede.

Når skærmen er fyldt med linier, så begynder det hele igen øverst til venstre, denne gang en punktrække lavere.

Prøv engang programet i MODE 1 og MODE 0.

Derefter ændrer De forsøgsvis linie 30 til:

```
30 POKE i,1
```

Nu har vi en punktrække, som fylder skærmen med lodrette rækker.

Når programmet kører i MODE 2, ser man, at de lodrette rækker befinder sig i højre side af tegnet. I MODE 1 får vi to lodrette rækker pr. karakter, i MODE 0 får vi ovenikøbet 4.

Vi vil foretage en sidste ændring i programmet. Slet nu linie 10 i programmet og indtast "MODE 2" i direkte mode. Skærmbilledet slettes og "READY" viser sig i øverste venstre hjørne. Benyt cursor-ned-tasten (nedadrettet pil) indtil READY-meddelelsen forsvinder ud af skærmen. Cursoren står nu i den sidste linie. Lad programmet køre endnu engang.

Resultatet er temmeligt irriterende.

Dette lille program har fortalt flere vigtige ting.

For det første har det bevist, at skærmlageret begynder ved &C000 og ender ved &FFFF. Overraskende nok er beliggenhed og størrelse i skærmlageret ens i alle 3 modes. Der skelnes altså ikke mellem MODE 0 og MODE 2. Kun farverne er forskellige.

Under alle omstændigheder giver et 16K-byte stort skærmlager i mode 0 ingen mening, da der kun vises 20 tegn pr. linie. 20*25 tegn giver kun 500 tegn på skærmen. Hvorfor behøver CPC tilsyneladende 16384 bytes for at frembringe 500 tegn?

Svaret er enkelt. Som allerede set indeholder CPC ingen Video-RAM, i hvilken et tegn lagres i en enkelt byte.

I 80-tegnsmode optager et tegn på skærmen 8 bytes, ved 40 tegn er det 16 bytes og ved 20 tegn hele 32 bytes. Det ses også af programmet, som fremkaldte de lodrette linier.

80 tegn mode er lettest at forstå, fordi en sat bit svarer til et punkt i den aktuelle tegn- (pen-) farve. Er en bit derimod ikke sat, viser baggrundsfarven sig på dette sted. Da kun 1 tegnfarve er mulig i mode 2, gives der ikke yderligere muligheder.

Men hvorfor er 32 bytes pr. tegn nødvendigt i mode 0?

Det forhold er, i MODE 0 og MODE 1, ikke så enkelt at beskrive. Prøv at indtaste nedenstående lille program og sammenholde resultatet med teksten her. De vil da bedre forstå beskrivelsen, end hvis De holder Dem til det teoretiske.

```
 10 MODE 2
 20 REM
 30 PRINT "A"
 40 FOR ADRESSE = &C000 TO &F800 STEP &800
 50 P$=BIN$(PEEK(ADRESSE),8)
 60 FOR L=1 TO 8
 70 IF MID$(P$,L,1)="1"THEN PRINT "X"; ELSE PRINT ".";
 80 NEXT L
 90 PRINT
100 NEXT ADRESSE
```

Lader De programmet køre som beskrevet, så vil De få et billede, som er magen til den trykte matrix "A".

Prøv at ændre MODE-kommandoen i linie 10 til "MODE1" og lad programmet køre. Resultatet er temmelig forbløffende.

At kun den halve matrix befinder sig i de udskrevne bytes, var at forudse. At den matrix også kun bruger en halv byte, altså bits 4 til 7, virker helt forvirrende.

Vi nærmer os gådens løsning, når vi erstatter linie 20 med:

```
20 PEN 2
```

Foruden ændringen af skrive- (PEN) farve, er der også sket en ændring i det frembragte bit-mønster. Det er det, der er løsningen på vort problem..!

Er De på dette tidspunkt blevet noget fortrolig med CPC, vil De vide, at 4 farver er mulige i 40 tegn-modus. Disse 4 farver lader sig slet og ret lagre sammen med tegnet, idet kun 4 bits er nødvendige for den placerede pixel, og low- og high-nibble (1 nibble =

en halvbyte, 4 bits) skelner mellem farverne. Anvendes det princip, er det kun nødvendigt for Gate-array at fordoble antallet af pixels for udlæsning i horisontal retning, for også virkelig at udlæse 8 punkter, hvor kun 4 punkter er lagret.

I MODE 0, ved udlæsning af 20 tegn pr. linie, udvides denne metode, endnu engang. Her indeholdes pixelinformationen kun i 2 bits. Placeringen af de to pixel i byten bestemmer den farve, punktet skal vises i. Dermed er ialt 16 kombinationer mulige, netop antallet af de til rådighed stående farver. Da kun to pixel er lagret i en byte, er det nødvendigt med 4 bytes for en linie, altså ialt 8 * 4 = 32 bytes for hvert tegn i 16 mulige farver.

Prøv programmet i MODE 0 med forskellige værdier for PEN-kommandoen. De vil hurtigt komme efter funktionsprincippet.

Dermed er der gjort rede for de 2 første punkter fra starten af kapitlet. Derimod er spørgsmålet om "forskydning" af skærm-RAM endnu uafklaret. Problemet skal søges i CPCs hardware.

Også en Z80 med en frekvens på 4 mhz, bruger en vis tid til forskydning af en 16K-datablok. For ikke at forskyde det samlede Video-RAM-område på 640 adresser ved hver linie under udlistning af et længere Basic-program, har man udnyttet en særlig egenskab ved CRTC. Gennem programmering af registrene 12 og 13 i 6845, kan skærmen begynde på enhver lige lagercelle i Video-RAM. Derved kan scrollning foregå meget hurtigt, da kun de aktuelle registre skal forsynes med de nødvendige værdier. Den nye linie ved nederste skærmkant slettes hurtigt og forsynes med nye tegn.

Start af Video-RAM på en ulige adresse, f.eks. &C001, er ikke muligt p.g.a. den beskrevne anvendelse af CCLK-signalet som adressebit.

Det følgende program viser, at manipulation af det nævnte register også er muligt fra Basic.

```
10 ADRREG=&BC00          :REM 6845'S ADRESSEREGISTER
20 DATREG=&BD00          :REM DATAREGISTERETS PORT
30 OUT ADRREG,13         :REM VALG AF REGISTER
40 FOR OFFSET=1 TO 40
50 OUT DATREG,OFFSET     :REM 40 GANGE AENDRING
60 FOR VENT=1 TO 40      :REM EN KORT PAUSE
70 NEXT VENT,OFFSET
```

I dette program scrolles skærmindholdet horisontalt. Udelades venteløkken, vil scrollningen foregå så hurtigt, at det ikke er muligt at følge den.

Også vertikal scrollning er muligt fra Basic. Dog skal her begge registre, Low- og Highbyte, manipuleres. Da der går en hel del tid mellem de to OUT-kommandoers udførelse, vil det resultere i en ubehagelig flimren på skærmen.

Vedrørende Video-RAM, er der endnu en særhed at bemærke.

Lad os lægge de kendte værdier sammen. I MODE 2 består et tegn af 8 bytes. Der er plads til 80 tegn i en linie, hvoraf der findes 25. Det giver et samlet lagerbehov på 80 * 25 * 8 = 16000 bytes. Men en 16K-blok indeholder jo 2 i 14. potens = 16384 bytes. Hvor er de manglende 384 bytes?

De benyttes faktisk ikke; i hvert fald ikke så længe skærmbilledet ikke scrolles. Her kan man kortvarigt mellemlagre værdier, der senest forsvinder ved næste CLS.

Man fristes til at spørge sig selv, hvordan man kan programmere fornuftig grafik med denne forrykte organisering af skærmbilledets hukommelse. Det synes også helt umuligt at aflæse et tegn fra skærmen. Hos andre computere er det ingen sag at POKE et tegn ind på skærmen, eller at udlæse et tegn fra Video-RAM med PEEK. Det er normalt også en gylden regel, at man altid kan finde Video-RAM fra en bestemt adresse.

Det er nu ikke så slemt, som det ser ud ved første blik. Operativsystemet er i stand til at finde ud af de skiftende startadresser, eller f.eks. at bestemme et tegn fra skærmmatrix, hvilket iøvrigt sker hver gang man benytter COPY-tasten. De dertil anvendte rutiner, kan også bruges i hjemmelavede maskinsprogsprogrammer.

Mange af rutinerne i operativsystemet, findes i et senere kapitel. Vi vil vise grafikmulighederne i et eksempel til tegning af firkanter, og et program til opbygning af en grafik-Hardcopy

1.7. PARALLEL-INTERFACE KREDS 8255

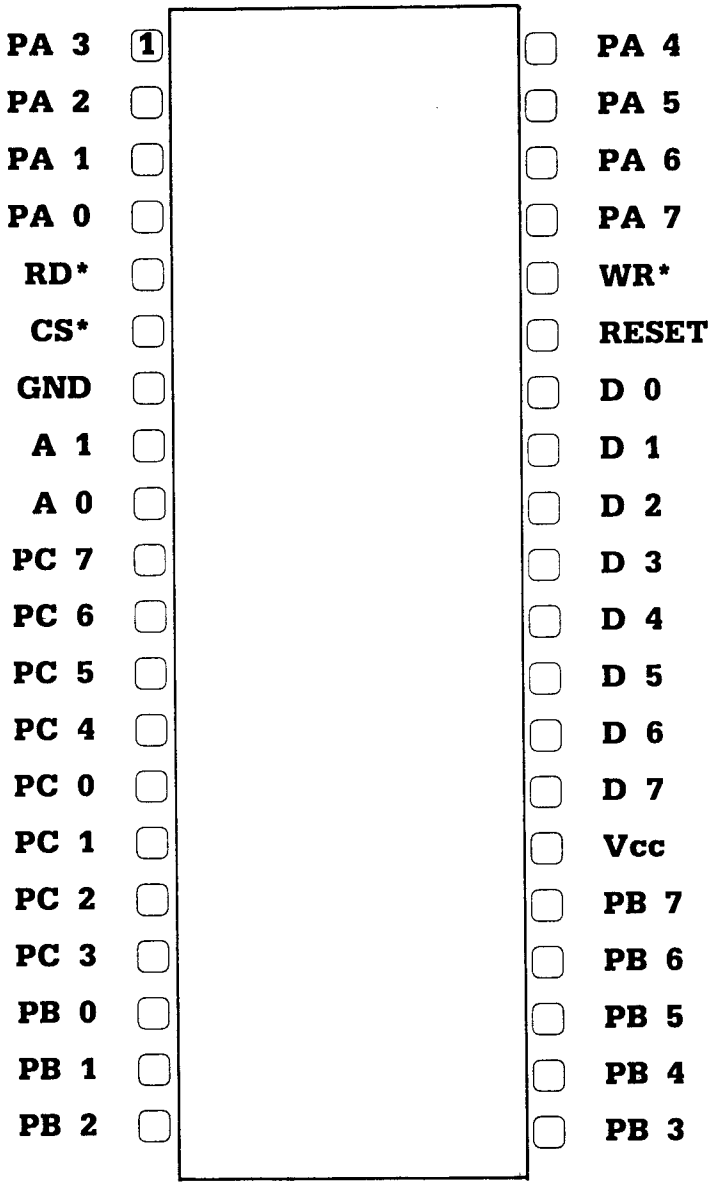
Den oprindeligt fra INTEL udviklede 8255, (brugt sammen med 8080), egner sig også som multifunktions-I/O-kreds til andre processorer. 8255 råder over ialt 24 porte gennem hvilke, signaler kan udlæses eller sendes. 8 ledninger danner en 8-bits port, hvor den 3. port kan deles op i to adskilte programmerbare halvdele.

De vigtigste 8255-data lyder:

24 programmerbare I/O-porte.
Enkel driftspænding på 5 Volt.
Fuldstændig TTL-kompatibel.
3 programmerbare driftmodes.
Hver port kan programmeres selvstændigt.
Høj udgangsstrøm 1 mA ved 1.5 Volt spænding.
Funktion sæt/reset bit muligt

1.7.1. 8255'S TILSLUTNINGER

Pin-belægningen på 8255 er vist på nedenstående tegning.



1.7.1.1 Pinbelægning for parallelport 8255.

De betyder:

D0-D7 : DATA LINES
Tilslutningen forbindes med processorens databus. De bruges til overførelse af data fra og til processoren.

CS : CHIP SELECT
Ved et Low på tilslutningen vælges kreds. 8255 accepterer de på RD-, WR- og dataledningerne liggende signaler.

RD : READ
Et low i tilslutningen tillader 8255 at sende data eller statusinformationer, over databussen, til processoren.

WR : WRITE
Bliver low, når processoren vil sende data eller styrekommandoer til 8255.

A0, A1 : ADRESS LINES 0, 1
Gennem tilslutningerne vælges mellem de tre datakanaler og styreregistret. Hyppigt forbindes tilslutningerne med de to nederste adresseledninger.

RESET :
Et high i indgangen reset'er alle registre inclusive styreregisteret. Port-ledningerne sættes til Input.

PA0 - PA7 : PORT A
Disse 8 ledninger udgør I/O-Port A og kan anvendes som ind eller udgang efter behag.

PB0 - PB7 : PORT B
Samme funktion som A.

PC0 - PC7 : PORT C
Samme funktion som A.

1.7.2. 8255's FORSKELLIGE MODES

Før vi beskæftiger os med de fire interne registre, må vi se nærmere på kredsens muligheder. Som sagt i begyndelsen, råder 8255 over 3 forskellige modes:

Mode 0: Enkel input/output
Mode 1: Tastet input/output
Mode 2: To-vejs BUS.

Funktionen 0 er den enkleste og hyppigste. I denne mode kan det bestemmes om portene skal virke som output- eller inputledninger. Programmeres ledningerne som udgange, og lægges der information fra processoren her, bliver værdien lagret, og udgangene opretholdes indtil nyprogrammering eller reset.

Programmeres porte som indgange, afleveres den øjeblikkelige status til disse ledninger ved læsning.

Både Port A og B lader sig kun programere som komplet Port for den ønskede data-retning. F.eks. er det ikke muligt at anvende Port-bits PA0, PA3 og PA7 som udgang og de resterende tilslutninger som indgang.

Port C kan imidlertid deles i 2 halvdele. Dataretningen kan programmeres selvstændigt i hver af halvdelene.

Mode 1 adskiller sig grundlæggende fra Mode 0. I denne Mode er det muligt at overføre i een retning gennem Hand-Shake-signaler. Nu tales der ikke mere om tre forhåndenværende Ports; Port C's to halvdele er til rådighed for de to andre Ports, som styre- og Acknowledgesignaler. Hermed menes de to grupper A og B.

Gruppe A består af Port A og bits 4 til 7 i Port C, gruppe B tilsvarende af Port B og bits 0 til 3 i Port C.

For at opnå en bekvem programmering af Mode 1, kan man på samme måde anvende en speciel bit, i den tilsvarende halvdel af Port B, som interrupt-signal.

En sådan 8 bits-dataoverførsel anvendes f.eks. i Printer-interfaces. Her indikerer et signal, at der ligger data parat på dataledningerne. Et retur-signal meddeler om modtageren, i dette tilfælde printeren, er klar til at modtage nye data, eller om data er korrekt modtaget.

Tredie funktion (mode 2) er tovejs. Funktionen er kun mulig ved hjælp af indgang A. Bits PC 3-7 anvendes som styre- og acknowledge-signaler.

Funktionen kan f.eks. anvendes til styring af et diskdrev, da det her skal være muligt at flytte data, både fra og til processoren, gennem den samme tilslutning.

I alle 3 modes er det desuden muligt at sætte eller slette, de som output programmerede, bits i porten.

Alle de her beskrevne funktioner lader sig kombinere. Således er det muligt, at programmere indgang A i mode 0 som udgang, port B i mode 1 som indgang, og de resterende bits i port C som indgang.

1.7.3. STYRING AF 8255, REGISTERBESKRIVELSE

Når man ser det forvirrende antal muligheder, så spørger man om, hvordan det er muligt at programmere alle disse kombinationer ved hjælp af kun eet styreregister.

Metoden er ganske enkel. Øverste bit i styrebyten bruges som markeringsbit. Er denne bit sat i styrebyten, så har bits 0 til 6 følgende betydning:

Bit 0 : Styrer Port C's funktion, bits 0 til 3.
1 = Indgang
0 = Udgang

Bit 1 : Styrer Port B's funktioner.
1 = Indgang
0 = Udgang

Bit 2 : Udvælger Mode for Gruppe B
1 = Mode 0
0 = Mode 1

Bit 3 : Styrer Port C's funktioner, bits 4 til 7
1 = Indgang
0 = Udgang

Bit 4 : Styrer Port A's funktioner.
1 = Indgang
0 = Udgang

Bit 6,5 : Bestemmer gruppe A's mode.
00 = Mode 0
01 = Mode 1
1x = Mode 2, Bit 5 er uden betydning

Hvis øverste bit derimod er slettet, så defineres Port C's funktion - sæt bit/slet bit - via bits 0 til 3. Betydningen af disse bits er som følger:

Bit 0 : Styrer sæt bit/slet bit
1 = Sæt bit
0 = Slet bit

Bits 3-1 : Bitudvalg
000 = PC0
001 = PC1
010 = PC2
011 = PC3
100 = PC4
101 = PC5
110 = PC6
111 = PC7

Bits 4 til 6 i styreordet er, når bit 7 slettes, uden betydning.

I styreregisteret kan der kun skrives. Det er ikke muligt at udlæse værdierne. Dog kan registrene, der angår portene læses. Det gælder også, når portene virker som udgang. I det tilfælde svarer den læste værdi til Port-ledningernes status.

Indgreb i de fire registre sker over tilslutnings-pins A0 og A1. Nedenstående tabel viser tilordningen til registrene:

A1	A0	
0	0	Port A Register
0	1	Port B Register
1	0	Port C Register
1	1	Styreregister

1.7.4. 8255'S OPGAVE I CPC

Efter at have fået overblik over de utallige muligheder for at anvende 8255, vil vi beskæftige os med den praktiske anvendelse af den universelle interface-kreds i CPC. Som næsten alle ICs i CPC, bliver 8255 også anvendt fuldt ud.

8255 betjener tastaturet, Sound-Chip, motoren i kassetterecorderen, skaberrecorderens skrivesignal, aflæser bit-strømmen fra recorderen, tester på V-sync-signalet i CRTC, afgør, om printeren er klar til at modtage. Undersøger med en bit tilstanden af EXP-signalet i ekspansions-connector. Afgør via en bro om skærmdisplay'et skal ske efter PAL- eller SECAM-normen med 50 eller 60 hz, og at der tilmed er 3 bits tilovers, der ved opstart af computeren kan erfare, hvilken computer de sidder i. Kontrollen finder nemlig ud af, om der skal skrives Schneider, Awa, Triumpf, Amstrad eller et andet af de ialt 8 mulige firmanavne i opstartsbilledet.

Alle disse funktioner, med de 24 disponible I/O ledninger, viser hard-warefabrikantens dygtighed og sans for at økonomisere.

Databussen er forbundet direkte med processorens databus. CS-signalet (chip-select) skabes med processorens adressebit A11. Pin A0 og A1 i 8255, som bruges ved registervalget, er forbundet med processor-adressepins A8 og A9.

Som omtalt, kommunikeres alle perifere kredse i CPC via Port-adresser. Derfor er ledningen RD* i 8255 forbundet med signalet IORD*.

Signalet frembringes ved sammenknytning af signalerne RD* og IORQ* i Z80. Kun, når IORQ* og RD* er low, fremstår et low gennem RD*.

På lignende måde styres WR*-tilslutningen i 8255. Her fremkommer et low, når såvel WR* som IORQ* i Z80 er low.

Med nærværende data kan 8255's Port-adresser bestemmes. For at skrive en værdi i register 0 i dataregistret i Port A, skal tilslutningerne A11, A9 og A8 være low. Binært set, vil adressebussens high-byte få værdien:

A15	A14	A13	A12	A11	A10	A09	A08
1	1	1	1	0	1	0	0

Det svarer til den hexadecimale værdi &F4.

De nederste 8 adressebits indgår ikke i 8255's udvalg. Her er enhver værdi mellem &00 og &FF mulig.

Heller ikke de satte bits i high-byte er nødvendige for korrekt adressering af 8255. Kom man på den ide at indsætte værdien 00H som high-byte, ville det såmænd også fungere. Men da dekodningen af de enkelte perifere kredse sker på en lignende ufuldstændig måde, skal disse bits være satte. Ellers vil andre kredse, såsom CRTC eller Gate-array tro, at man henvendte sig til dem også.

Men tilbage til vort eksempel. For at lade register A med en værdi, skal adressebussen pålægges værdien &F400, hvilket opnås med kommandoerne:

```
LD A,værdi
LD BC,&F400
OUT(C),A
```

Ligeledes, kan Port-register C udlæses med kommandoerne:

```
LD BC,&F600
IN A,(C)
```

Principielt anvendes alle tre Ports i mode 0. Derved står alle 24 tilslutninger til rådighed som I/O-ledninger.

Port A (&F400) er forbundet med tonegeneratoren AY-3-8912's otte dataledninger. Alt efter ønsket funktion, programmeres Port A som ind- eller udgang.

Progammeret som udgang sendes styrekommandoerne til Sound-kredsen via de 8 Port-ledninger. Kommandoerne bliver uddybet i kapitlet om programmeringen af AY-3-8912. Her skal kun fortælles, at Sound-chippen også råder over en bidirektional 8-bits-Port. En del af tastatur-matrix er tilsluttet den port. Via Port A i 8255 kan det gennem AY-3-8912 fastlægges, om en tast er trykket ned. Til det formål skal Port A være programmeret som indgang.

Port B (&F500) programmeres fast som indgangsport. Gennem porten udføres alle tests bortset fra tastaturscanning. De enkelte bits har følgende funktioner:

Bit 0 : Denne bit tester CRTC's W-sync-status. Da testen skal ske i en fart, kan den med INP læste værdi af bit 0 ved rotering forskydes til carry-flag. Så hurtigt kan tilstanden i V-sync bestemmes.

Bits 1-3 : Udvælger firmanavnene i opstartsteksten.

Bit 4 : Bit'en er forbundet med en ledningsbro. Er broen åben, bliver Video-Controlleren programmeret til 50 hz for PAL-drift. Er broen lukket, vil CRTC blive programmeret til en skærmopdatering på 60 hz svarende til SECAM-normen. Denne programmering er vigtig når CPC skal køre sammen med et TV-apparat via MP1-modulet.

Bit 5 : Tester EXP-signalet fra expansions-connectoren.

Bit 6 : Viser en tilsluttet printers status. Da printeren ikke kan modtage tegn til hver en tid, gives der mulighed for at "holde pause" i en dataoverførsel til printeren ved at lægge ledningen high.

Bit 7 : Via bit'en læses de med TTL-niveau leverede data fra recorderen. Også her gælder det, der er sagt om bit 0. Idet ledningen skal kunne testes i en fart, kan man ved en enkel rotation af bit 7 til Carry-flag, omgående bestemme ledningens status.

Den resterende Port C (&F600) er programmeret som fast udgangsport i CPC. Med 4 af sine 8 ledninger styrer den en del af tastatur-scanningen, og yderligere 2 bits anvendes til recorderen. De sidste 2 bits er nødvendige for styring af Sound-chippen. Da ledningerne i Port C direkte kan sættes og slettes, egner den sig især til at løse disse opgaver.

Hver for sig har de følgende funktioner:

Bits 0-3 : Styrer tastaturmatrix. De fire ledninger, som er programmeret som udgang, er forbundet med en BCD-decimal-decoder, der konverterer de binære informationer til decimale værdier. Decoderen lægger, svarende til de binære indgangsinformationer, en af sine 10 udgange til Ground. Værdierne mellem 0 og 9 er valide som indgangskombinationer.

Bit 4 : Bit 4 styrer kassetterecorderens motor. Motoren styres ikke direkte, men gennem en transistor der driver et relæ. Er bit 4 low, afbrydes motoren.

Bit 5 : Gennem denne Pin i 8255 leveres de lydfrekvenser fra computeren, som indspilles af recorderen.

Bits 6-7 : Disse Port-bits er forbundet med Sound-chippen gennem tilslutningerne BC1 og BDIR, og virker som Chip-Select- og Strobe-signal for AY-3-8912. En nærmere beskrivelse af tilslutningerne bringes i det næste kapitel, som omhandler tonegeneratoren.

1.8. TONEGENERATOREN AY-3-8912

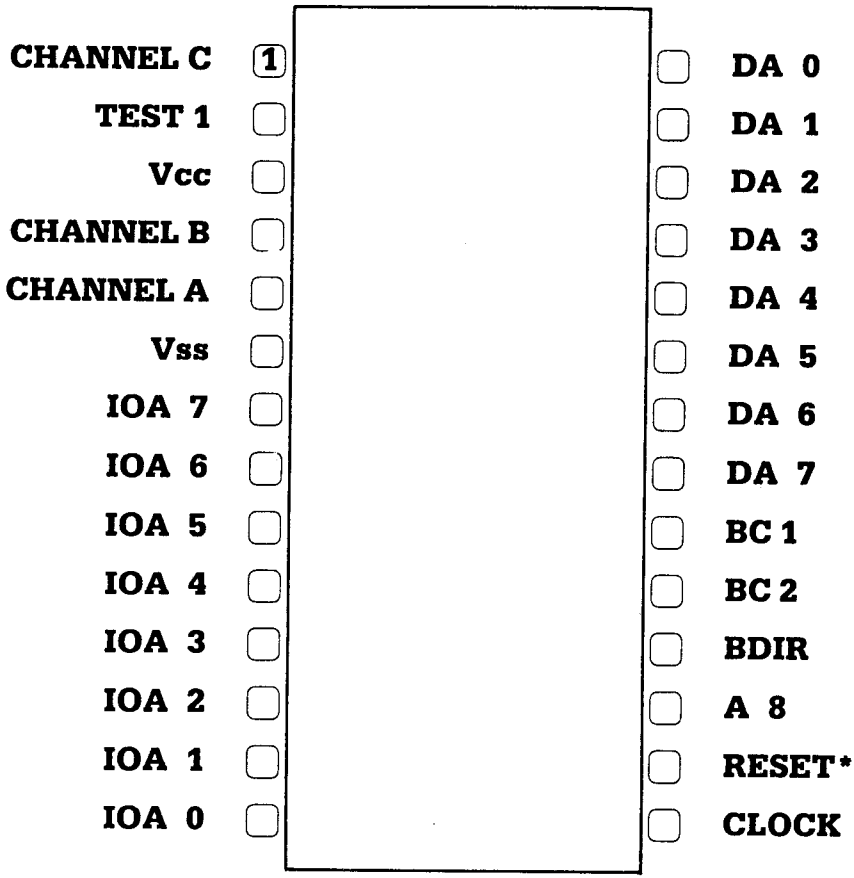
AY-3-8912 fra General Instruments er en programmerbar tonegenerator (PSG) i topklasse. Den blev udviklet til TV-spil, for at forsyne disse med realistiske lyde, hvor de første TV-spil kun frembragte unuanceret støj. For at gøre kredsen fleksibel, er den udstyret med talrige klangmuligheder. Under udviklingen af kredsen har man erkendt, at de fleste steder, hvor den kunne tænkes anvendt, ville der også være brug for en eller anden form for taster, joysticks eller kontakter. Derfor blev Sound-chippen udstyret med en 8-bit-parallel-Port.

I nedenstående oversigt ses kredsens grundlæggende egenskaber:

Tre uafhængige programmerbare tone-oscillatorer
En programmerbar støjgenerator
Komplet software-styrede analog-udgange
Programmerbar mixer for tone- og støj
15 logaritmisk adskilte lydstyrketrin
Programmerbar indhyldningskurve
Bidirektional 8-bit-dataport
TTL-kompatibel
Simpel 5-volt driftspænding

AY-3-8912 har 16 registre, af hvilke 15 kan benyttes. Alle kredsens klangmuligheder kan programmeres via registrene.

Kredsen kan internt opdeles i enkelte funktionsblokke.



1.8.1.1 Sound-chip AY-3-8912.

En af blokkene indeholder tonegeneratorerne. De forsynes med et taktsignal, som svarer til Clock-frekvensen delt med 16. Tonegeneratorerne sørger for den grundlæggende opbygning af de tre firkant-toner.

Støjgeneratoren frembringer et frekvensmoduleret firkant-signal, hvis pulsbredde styres af en pseudo-støjgenerator.

Mixeren blander de tre generatorers udgangssignaler med støj-signalet. Blandingen kan programmeres adskilt for hver af kanalerne.

Amplitudekontrollens funktionsblok giver brugeren to muligheder. Via programmering af et volumen-register, kan man styre de 3 kanalers styrke.

Alternativt kan man øve forskellig indflydelse på dem gennem PSG. I så fald benytter man udgangen fra indhyldningskurvens register til ændringen af lydstyrken. Da indhyldningskurven (envelope) er programmerbar med 4 adskilte parametre, findes der mange muligheder for indflydelse på lyden.

D/A-konverterens blok er ansvarlig for udgangssignalernes volumen.Eftersom lydstyrke- og envelope-informationerne foreligger som digitalværdier, bliver de konverteret i D/A-konvertereren. Den sidste funktionsblok har intet med lyden at gøre. Den indeholder faktisk to komplette I/O-porte, men kun den ene er ført ud til kredsens benforbindelser.

1.8.1. SOUND-CHIP'ENS TILSLUTNINGER

Da PSG's tilslutninger ikke umiddelbart er selvforklarende, vil vi bringe en mere detailleret beskrivelse:

DAO-7 :

Tilslutningerne forbindes med processorens databus. Betegnelsen DA betyder, at såvel data som registeradresser ledes til samme tilslutning.

A8 :

Tilslutningen kan opfattes som et CHIP-SELECT-signal. Ved henvendelse til PSG's register må denne tilslutning være high.

BDIR & BC1,2 :

Tilslutningen BDIR-signalet (Bus DIRection) og tilslutningerne BC1 og BC2 (Bus Control) styrer registeraccess i PSG. Ved første blik forekommer den, i tabellen viste tilordning, noget ulogisk. Da denne IC oprindelig blev udviklet som en kreds til processoren 1610, en speciel 16-bit-processor fra General Instruments, tog man hensyn til denne processors særlige egenskaber og styretilslutninger.

BDIR	BC2	BC1	PSG's funktion
0	0	0	inaktiv
0	0	1	latch adress
0	1	0	inaktiv
0	1	1	læs fra PSG
1	0	0	latch adress
1	0	1	inaktiv
1	1	0	skriv til PSG
1	1	1	latch adress

Af disse tabeller er kun 4 kombinationer af betydning. Derfor lægges tilslutning BC2 hyppigt på +5 volt. Resten af tabellen bestemmes kun af signalerne BDIR og BC1, og ser ud som følger:

BDIR	BC1	funktion
0	0	PSG databus er høj-ohm'sk
0	1	læs data fra PSG
1	0	skriv data i PSG
1	1	skriv registernummer i PSG

ANALOG A :

Det er udgangen fra kanal A. Her kan udtages de fra kanal A frembragte toner. Den maximale udgangsspænding er 1 Vss.

ANALOG B :

Funktion som pin 1 til kanal B.

ANALOG C :

Funktion som pin 1 til kanal C.

IOA7-0 :

IOA-tilslutningerne udgør 8-bit-Porten i PSG. Tilslutningerne arbejder som ind- eller udgange, alt efter programmering. Derved kan man kun indstille retningen for hele porten. En blandet mode (nogle bits som indgang, andre som udgang samtidig) er ikke mulig.

CLOCK :

En deling af denne frekvens styrer alle lydfrekvenser. Signalets frekvens skal ligge mellem 1 og 2 MHz.

RESET :

Ved hjælp af et low-niveau bliver alle interne registre nulstillet. Udelades denne reset, vil registrene indeholde tilfældige værdier ved opstart. Resultatet vil være en (sandsynligvis) ikke særlig musikalsk støj.

TEST1 :

Anvendes kun af fremstillingsfiramaet og skal forblive utilsluttet.

Vcc :

En driftsspænding på +5 volt lægges på tilslutningen.

Vss :

GROUND.

1.8.2. DE ENKELTE REGISTRES FUNKTIONER I 8912

De registernumre, som anvendes i den følgende opstilling, er de samme som de tal, som skal indføres i adresseregistret, for at kommunikere det enkelte register.

Det er en interessant kendsgerning, at adresseregistret beholder sit indhold indtil næste programmering. Man kan altså benytte et dataregister flere gange, uden hver gang at skulle lade adresseregistret pånyt.

Og nu til beskrivelsen:

Reg 0,1 : Registeret bestemmer perioden og dermed frekvensen for lyden på ANALOG A. Imidlertid benyttes ikke alle 16 bits. De 8 bits i register 0 og de 4 nederste i register 1 anvendes. På den måde kan frekvensen med register 0 fint påvirkes med register 1. Jo mindre 12-bit-værdien i disse registre bliver, desto højere bliver tonen.

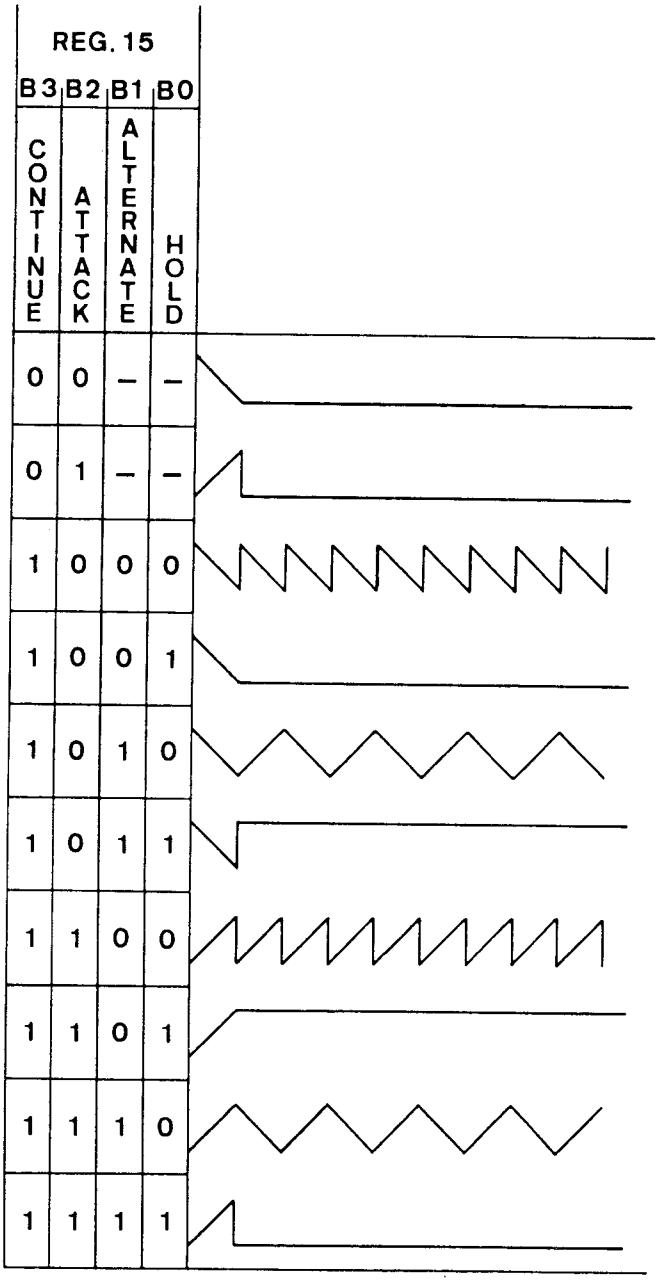
Reg 2,3 : Funktion som reg 0,1, men gennem kanal B.

Reg 4,5 : Funktion som reg 0,1, men gennem kanal C.

Reg 6 : Registeret påvirker støjgeneratoren gennem sine 5 nederste bits. Også her gælder det: Jo mindre registerværdien er, desto højere bliver støjfrekvensen.

Reg 7 : I dette multifunktions-register kontrollerer de enkelte bits forskellige opgaver. I den følgende tabel vises påvirkningerne:

Bit 0:	Lyd fra kanal A On/Off 0=On/1=Off
Bit 1:	Lyd fra kanal B On/Off 0=On/1=Off
Bit 2:	Lyd fra kanal C On/Off 0=On/1=Off
Bit 3:	Støj til kanal A On/Off 0=On/1=Off
Bit 4:	Støj til kanal B On/Off 0=On/1=Off
Bit 5:	Støj til kanal C On/Off 0=On/1=Off
Bit 6:	Port A som ind- og udgang 0=ind/1=ud
Bit 7:	Port B som ind- og udgang 0=ind/1=ud



1.8.2.1 PSG-lydkurver.

Reg 8 : Registeret bestemmer lydstyrken af kanal A's signal. De 4 nederste bits anvendes som volumenkontrol.
Bit 4 har en særlig funktion. Er den sat, så bestemmes lydstyrken ved hjælp af lydkurvens register. Man ignorerer således indholdet af bits 0 til 3.

Reg 9 : Som reg 8 for kanal B.

Reg 10 : Som reg 8 for kanal C

Reg 11,12 : Alle 16 bits i de to registre påvirker lydkurvens varighed. Indholdet i register 11 betragtes som low-byte, d.v.s. finindstilling af perioden, mens reg 12 er lyd-generatorens high-byte.

Reg 13 : Bits 0 til 3 i registeret bestemmer lyd-generatorens kurveformen. Det er så godt som umuligt at beskrive tilordningen i en forståelig tekst. Den frembragte lydkurve er derfor vist grafisk i 1.8.2.1.

1.8.3. SÅDAN FUNGERER AY-3-8912 I CPC

Her skal vi se på den konkrete tilslutning, og nogle praktiske forhold vedrørende driften af sound-chip i CPC. Da den forrige registerbeskrivelse fremstod noget abstrakt, og måske ikke særlig overskuelig, så vil De, efter at have læst dette kapitel, bedre forstå nogle af specialiteterne i PSG.

Pins 3, 17 og 19 i Sound-chip'en er tilsluttet +5 volt. AY-3-8912 får sin driftsspænding over pin 3. Da BC2 (pin 19) og A8 (pin 17) er tilsluttet +5 volt, indgår de ikke i registerudvalget.

De resterende register-styretilslutninger BC1 (pin 20) og BDIR (pin 18) er forbundet med Port-bits PC6 og PC7 i 8255. Alt efter disse tilslutningers status kan man meddele registeradresser til PSG, så vel som skrive eller udlæse data i PSG.

De egentlige adresse- og dataoverførsel sker gennem PSG-tilslutningerne D0 til D7, som er forbundet med port A i 8255. Alt efter den ønskede funktion, skal Port A i 8255 programmeres som ind- eller udgang.

Clock-signalet i pin 15 er et firkantsignal med en frekvens på 1 mhz. Signalet leveres af gate array gennem deling af kvartsfrekvensen. Alle lyd- og indhyldningskurvefrekvenser er et resultat af en deling af signalet.

I/O-Porten i PSG er forbundet med tastaturet og tilslutningen for joy-stick. En detailleret beskrivelse af tastaturet og joy-stick bringes i et senere kapitel. Her vil vi kun interessere os for de lydmæssige muligheder i Sound-chip'en.

De vigtigste tilslutninger til denne IC er de tre analog-udgange A, B og C på pin 1, 4 og 5. Disse udgange er udført som open-emitter-udgange. For at frembringe en lyd-vekselspænding er de tre 1K-modstande, som er tilsluttet mellemudgang og GND, nødvendige.

Tonesignalet mixes over 3 modstande, og bruges som monosignal i tilslutning 1 i ekspansionskonnektoren. Dette monosignal bliver også ført til den interne forstærker og videre til højttaleren.

De 3 udgange bliver også ført til stereo-udtaget på bagsiden af computeren. Dertil lægges signalet fra kanal B, samtidig over 2 modstande, på de to stereokanaler. Udgangene A og C bliver lagt direkte over hver sin bipolær-kondensator på en af stereokanalerne. Ved hjælp af denne tilslutningsmetode, kan man, gennem passende programmering, opnå ægte stereo-effekt. Lad os tænke os, at en tone først fremkommer gennem kanal A. Efter en tid kan samme tone yderligere fremkomme gennem kanal B. Her kunne signalets lydstyrke i kanal B langsomt stige, tilsvarende kan signalets lydstyrke reduceres. Resultatet bliver, at tonen synes at vandre fra et hjørne af rummet til et sted midt mellem de to højttalere. Herfra kan den efter behag fortsætte til det andet hjørne.

De muligheder er endda til stede i BASIC med den stærke SOUND-kommando. Brugervejledningen modsiger desværre sig selv,når den omtaler fordelingen af de tre tonekanaler på de to stereokanaler. Vær opmærksom herpå, når De forbinder Deres CPC med et stereoanlæg. Kun kanal B's toner lyder på begge stereoanlæggets kanaler.

Men hvordan frembringer PSG egentlig tonerne? Lad os engang i detaljer se på, hvad der foregår i en kanal.

Som allerede omtalt ledes alle toner fra clock-signalet i pin 15. For det første deles taktsignalet med 16. Derved opnås en styrefrekvens på 62,5 khz. Denne frekvens føres nu til en programmerbar frekvensdeler. I overensstemmelse med tonegenerator-registrets indhold deles styrefrekvensen yderligere for at opnå den ønskede tonefrekvens.

På den måde har IC's konstruktører grebet særlig dybt i trick-posen. Delekæden består ikke bare af flip-flops, som frekvensen kan dele med 2. Gennem en speciel teknik, er også ulige delefaktorer mulige. Derved kan styrefrekvensen også deles med 3 eller 17. På den måde kan alle nødvendige værdier i det høje frekvensområde frembringes.

Tonegenerator-registrets indhold bestemmer altså tonesignalets delefaktor. Lader man register 0 i PSG med værdien 100 og register 1 med værdien 0, deles styrefrekvensen med 100. Ved udgangen af kanal A's delekæde ligger et signal med frekvensen 625 Hz.

Signalet kan endnu ikke udtages ved udgang A. Til det formål må den tilsvarende kanal først aktiveres. Det opnås ved at slette den tilsvarende bit i register 7. Da vi har valgt vort eksempel med kanal A, skal vi slette bit 0. Vi må så passe på de øvrige bit's tilstande. I CPC betyder det konkret, at man ikke kan ændre bit 6 utilsigtet, da tastaturet i så fald spærres. Man vil sikkert endnu ikke kunne høre nogen tone, da den aktuelle kanals lydstyrke endnu ikke er indstillet. For kanal A skal register 8 bruges. En værdi på 1 frembringer nu en svag tone. Ved værdien 15 opnås den maximale lydstyrke.

Sætter vi bit 4 i lydstyrkeregistret, ignoreres informationerne i bits 0 til 3. Nu bestemmer registrene 11, 12 og 13 lydstyrken. Lydstyrken er altså ikke mere fastsat til en værdi, men er variabel.

Lad os se på register 13. Registeret har den officielle betegnelse "ENVELOPE-SHAPE/CYCLE-CONTROL-REGISTER". Den funktion kan bedst forklares med et eksempel.

Efter at have forsynet registrene 0, 1, 7 og 8 med passende værdier, skriver vi værdien 12 i register 13. Nu er bits 2 og 3 sat, og de nederste to bits slettet.

Registerbeskrivelsens tabel viser en række langsomt stigende og hurtigt faldende "takker" ved denne kombination. I praksis vil det sige, at tonens lydstyrke langsomt stiger til maximum. Så afbrydes tonen, og lydstyrken tiltager atter. Tilstanden vedvarer, indtil en ny ordre sendes til register 13.

Tiden for lydstyrkens stigning kan indstilles over registrene 11 og 12. Registrene påvirker yderligere en programmerbar delekæde i PSG, som vi kender det fra tonegenerator-registrene. Fordelingskæden forsynes med et signal, som svarer til det gennem 256 delte Clock-signal. Det giver en frekvens på 3906,25 Hz, svarende til en periodelængde på ca. 250 mikrosekunder.

Tilskriver man register 11 værdien 1, og værdien 0 til det high-byte-arbejdende register 12, bliver lydstyrken virkelig skruet helt op i løbet af 250 mikrosekunder. Men det ligger i det hørbare område og frembringer en tydelig piben, som overlejrer den ønskede tone.

Derfor vælges registerværdierne altid højere. Ved maximalværdien (255 i reg 11 og reg 12) varer stigningen til fuld lydstyrke hele 16,8 sekunder.

Påvirkningen af lydstyrken over envelope-registret anvendes ikke af CPC's software. ENV-kommandoen har kun indflydelse på lydstyrken gennem manipulation af de fire nederste bits i lydstyrkeregistret. I PSG findes ingen ENT-kommando, som svarer til CPC's. Funktionen frembringes ved passende ændringer i tonegeneratorregistret.

1.9. DISKETTESTATIONEN I CPC 664 OG 6128

I modsætning til 464 er floppy-interface og et drivværk integreret i 664 og 6128 computerne. Derved bliver ikke blot ekspansionsudgangen på bagsiden lettere tilgængelig i større omfang. Der opnås også et mindre behov for arbejdsplads, ligesom antallet af ledninger reduceres.

Kredsløbet i alle tre computere er fundamentalt, såvel funktions- som software-mæssigt fuldstændig kompatible. Al litteratur om Amstrad-floppymaskiner (f. eks. DATA BECKER, Den store Floppybog) er gyldig for alle computerne, således, at også ejeren af en CPC 664 eller 6128, kan anvende de indtil nu udkomne bøger.

Midten af Controller-Board udgør den integrerede floppy-disc-controller (FDC) uPD 765. Denne IC udgør interfacet diskdrevene og processoren i CPC. Man kan tilmed opbygge floppystationer uden FDC, men den høje "intelligens" i FDC forenkler hele konstruktionen. Omfanget af den nødvendige hardware, og den for systemet nødvendige software, reduceres kraftigt ved brug af FDC. Det tydeliggøres gennem et eksempel. Firmaet Commodores floppystation 1541, der kendes af mange som floppystation til Commodore C64, er opbygget uden FDC. Bortset fra den, konstruktivt betingede, langsomme hastighed af dataoverførelse (som De som CPC-ejer kun kan more Dem over) er forbruget af hardware betydeligt mere ødselt, end ved CPC-floppy. Digitalelektronik'en i 1541 indeholder sin egen processor, 2 40-polede periferi-IC og forskellige TTL-IC'er. En komplet CPC 664 indeholder et lignende antal kredse!

Styresystemet til 1541 er med sine 16 K dobbelt så stor som AMSDOS. Der er ingen tvivl om, at konstruktørerne (af bekvemmelighedsgrunde) og at forhandlerne (af prismæssige årsager) foretrækker den mere behagelige FDC.

Alt i alt består controllerens hardware blot af en håndfuld kredse. Det forbavsende ringe antal, opnås kun gennem den høje integration af 3 IC'er. Her menes FDC, dataseparatoren og ROM indeholdende AMSDOS.

Den 16 Kbyte store AMSDOS-ROM indeholder ca. 8 Kbyte væsentlige rutiner, der er nødvendige for driften af floppyen. Den 28-bens-ROM indeholder desuden, i de sidste 8 Kbytes, en del af den på diskette leverede CP/M 2.2-LOGO fortolker.

1.9.1. FDC 765

Den fra firmaet NEC uPD 765, firmaet ROCKWELL's R 6765 og INTEL's 8765 i leverede FDC, må betragtes som en højt specialiseret micro-processor. IC'ens muligheder er så omfattende og komplekse, at denne vurdering ikke kan betragtes som overdreven.

Det af FDC benyttede dataformat svarer til IBM-format 3740 i single-density og IBM-system 84 i double-density. Gennem det format kan desværre hverken Commodore- eller Apple-disketter læses eller skrives.

Med sine 40 pins understøtter den brugen af alle de i handelen tilgængelige drevformater. Gennem de eksisterende styresignaler er udvikleren i stand til at slutte FDC'en til næsten enhver processor. Således er der to grundlæggende muligheder for tilslutning og drift. Den første mulighed er DMA-driften. Sammen med en DMA-controller kan FDC m.h.t. dataoverførelse, ved læsen og skriven overtage kontrollen over computersystemets lager. Den henter eller skriver ved hjælp af DMA-controlleren de nødvendige nye data, efter at have overtaget kontrollen over hukommelsen. Denne meget hurtige metode for dataoverførelse bliver imidlertid ikke udnyttet i CPC og omtales kun her for fuldstændighedens skyld.

Ved den anden, i CPC anvendte metode, ovetages dataoverførelsen af processoren. Ved brug af den metode, må der skelnes mellem 2 mulige udnyttelser af FDC.

Det er først interrupt-metoden. Herved vil enhver dataoverførelse frembringe en interrupt. I processorens omtalte interrupt-rutine, skal de næste data- eller ordrebytes leveres af processoren, eller læses. Ved konstruktionen af CPC's hardware kom metoden heller ikke i betragtning. I stedet har konstruktørerne brugt Polling-metoden. Derfor må processoren regelmæssigt efterprøve hvilke aktioner, der skal udføres af FDC inæste procedure.

Lad os se på 765's data. Tænk i den forbindelse på, at udviklerne af Controller-Board ikke har udnyttet alle 765's muligheder.

Programmerbar sektorlængde
Alle drevdata er programmerbare
Mulighed for tilslutning af 4 drev
Valgfri dataoverførelse i DMA- eller ikke DMA-mode
Kan tilsluttes næsten alle gængse processortyper
Simpel 5 volt strømforsyning
Enkel 1-faset-takt på 4 eller 8 mhz
40-polet IC.

Den sidste oplysning i denne korte fremstilling, vil vi beskæftige os mere detailleret med.

1.9.2. PIN-BELÆGNINGEN PÅ FDC

Tilslutningerne på FDC 765 lader sig inddele i forskellige grupper. Den første gruppe udgør interfacet til systemprocessoren. Gennem tilslutningen bestemmes styringen af FDC fra processoren.

Den anden gruppe er kun nødvendig i forbindelse med DMA-funktioner. Gennem signalerne kommunikerer DMA-controlleren og FDC.

Interface til floppydrevene udgøres af den 3. gruppe, som er den talstærkeste med 19 tilslutninger.

Den 4. og sidste gruppe omfatter tilslutningerne til strømforsyningen og takten.

Lad os se nærmere på den første gruppe.

Processor-interface

RESET:
FDC's RESET-tilslutning er high-aktiv. Ved normal brug befinder denne tilslutning sig på GND. Gennem en high på RESET-pin bringes FDC i en foruddefineret tilstand.

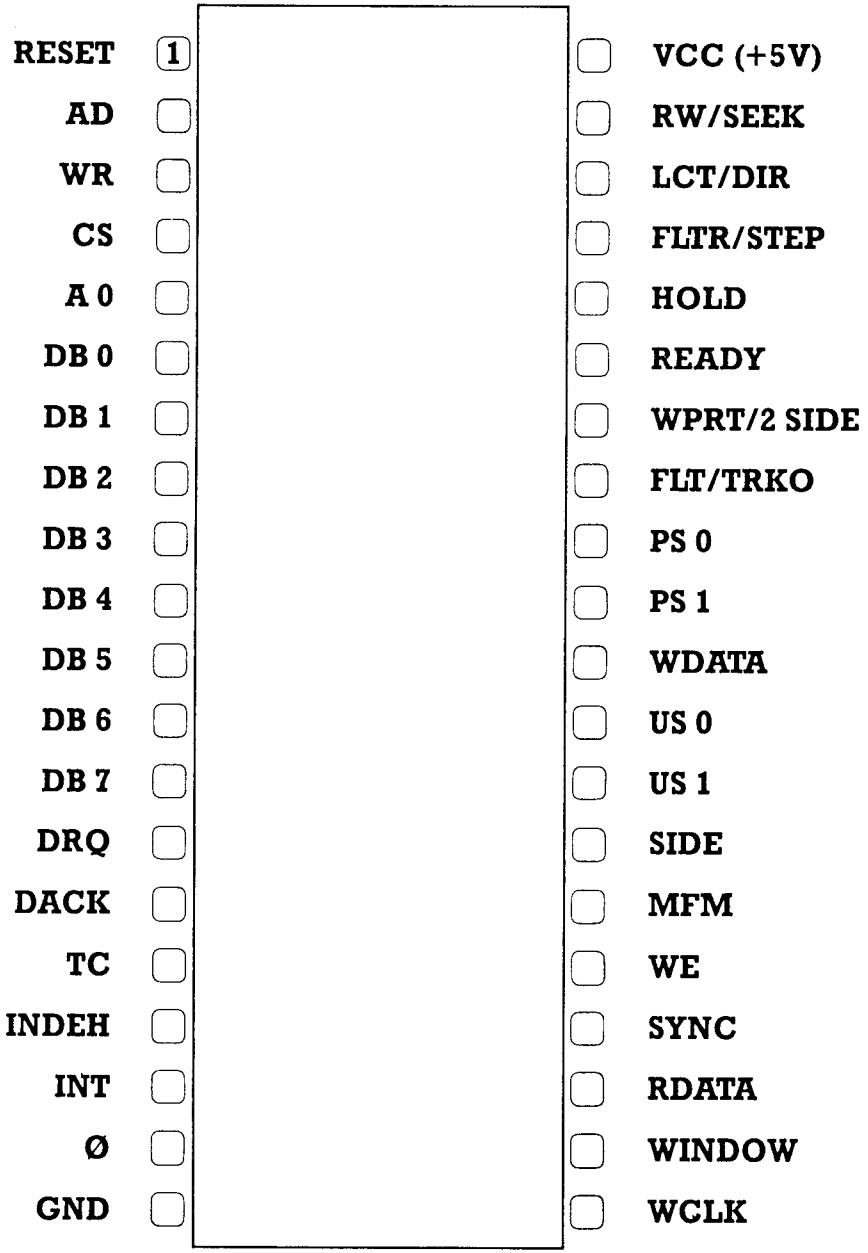
SC* : CHIP SELECT
Ved hjælp af et low på denne pin udvælges FDC. Først gennem CS* = low bliver RD* og WR* accepteret. Da konstruktøren af CS var frit stillet m.h.t. udviklingen af CS, kan man via memory-mapping henvende sig til FDC, d.v.s. via en del af hukommelsen eller via Portadresser.

RD* : READ*
Tilslutningen skal være forbundet med processorens RD*-signal. Hver gang processoren vil læse oplysninger fra FDC, må forbindelsen lægges i low.

WR* : WRITE*
Sådan som RD*-ledningen signaliserer læseordrer til processoren, sådan viser et low i WR*, at processoren skriver data eller kommandoer i FDC.

A0 : ADRESS LINE 0
FDC råder kun over to adresser, som kan påvirkes udefra. Valg mellem de to adresser foretages med signalet A0. Denne ledning er normalt forbundet med processorens nederste adressebit.

DB0 - DB7 : DATABUS 0-7
Tilslutningerne i FDC forbindes med systemdatabussen. Alle kommandoer og data transporteres over disse 8 bidirektionale tilslutninger. Den aktuelle dataretning bestemmes herved, enten af processoren eller af DMA-controlleren i DMA-mode.



1.9.1.1 Pinbelægning FDC 765.

INT : INTERRUPT

Via denne tilslutning kan FDC frembringe en interrupt i systemprocessoren. Interrupts frembringes under enhver bytes-transfer (ikke tilsluttet i CPC).

Signaler for DMA-modus (ikke anvendt i CPC)

DRQ : DMA REQUEST

Gennem tilslutningen giver FDC signal til DMA-controlleren, at der forestår et lagerindgreb. Ved først mulige lejlighed overtager DMA-controlleren systembussen. Herved afbrydes processoren.

DACK* : DMA ACKNOWLEDGE

Med signalet viser man FDC, at DMA-controlleren har overtaget bussen og har påbegyndt dataoverførelsen.

TC : TEMINAL COUNT

Gennem en high på tilslutningen afbrydes dataoverførelsen fra og til FDC. Skønt tilslutningen hovedsagelig anvendes i DMA-modus, kan man også afbryde dataoverførelsen i interruptstyrede systemer.

Floppy-interface

US0, US1 : UNIT SELECT 0/1

Gennem de to tilslutninger kan man direkte tilslutte 2 eller endog 4 drev ved hjælp af en 2-4-decoder. Via tilslutningen bliver det aktuelle drivværk beordret til at læse eller skrive data.

HD : HEAD SELECT

Eftersom FDC er forberedt til brugen af Double-Side-drev, kan man ved anvendelse af sådanne drev vælge mellem læse/skrivehovederne via tilslutningen.

HDL : HEAD LOAD

Signalet indføres næsten udelukkende ved brug af 8" drivværk.Disse drevs motorer startes ikke op for hver læsning/skrivning, men kører hele tiden. For at skåne disketter og læse/skrivehoveder flyttes læse/skrivehovederne efter behov til diskettens overflade. Styringen sker ved hjælp af HDL.

IDX : INDEX

Gennem tilslutningen bliver det, af index-udstandsningen frembragte, signal videreført og angiver den fysiske start på et spor til FDC.

RDY : READY

Det af floppy leverede signal READY angiver, at der befinder sig en diskette i drevet, og at den roterer med en vis mindste-hastighed. Først ved READY's opståen starter FDC læse/skrive-aktionen.

WE : WRITE ENABLE

Udgangen fra FDC skal være high, for at kunne skrive data på disketten.

RW/SEEK : READ WRITE/SEEK

Alt i alt leverer et floppydrivværk flere signaler, end hvad der står til rådighed for floppyinterface via 40-pins. Imidlertid er alle signaler ikke nødvendige samtidig. 8 af signalerne har man derfor delt op i 2 grupper, som efter brug lægges på FDC's 4 tilslutninger. Gennem tilslutningerne RW/SEEK udvælger FDC de til enhver tid nødvendige signaler.

FR/STP : FIT RESET/STEP

Det er det første af 4 dobbeltsignaler i FDC. Udgangene har efter udført operation forskellige betydninger. Man kan ved hjælp af denne tilslutning reset'te forskellige Error-flip-flops i et drev. En anden langt hyppigere anvendelse er styringen af drevets Step-indgang. Ved ethvert skift af læse/skrivehovede bliver de nødvendige impulser givet gennem denne tilslutning.

FLT/TR0 : FAULT/TRACK0

Også denne indgang kan forstå 2 forskellige signaler. Gennemfører man en SEEK-operation (se programmering af FDC), afventes der et TRACK0-signal fra drevet. Signalet styres ved hjælp af et lyssignal eller en kontakt, når skrive/læsehovedet befinder sig ved det fysiske spor 0. Den anden funktion, FAULT-signalet, frembringes af nogle drev i fejltilfælde og kan atter slettes fra FDC, med det tidligere beskrevne signal FR/STP. Signalet testes ved READ/WRITE-operationer i FDC.

LCT/DIR : LOW CURRENT/DIRECTION

Stepimpulser fra FR/STP angiver kun, at hovedet skal bevæges. LCT/DIR bestemmer dertil i SEEK/MODUS-retningen af bevægelsen. Ved skrivning af data er funktionen LOW CURRENT nødvendig. Ved hjælp af signalet aftager skrivestrømmen på det indre spor. De kan finde enkeltheder om signalet i beskrivelsen af det teoretiske grundlag for diskettelagring.

WP/TS : WRITE PROTECT/TWO SIDE

Uafhængig af de forskellige metoder (gennem de forskellige drevstørrelser) bliver tilstanden af skrivebeskyttelsen meldt som signal fra drev til controlleren. Signalet testes ved skrive/læse-operationer fra indgang WP/TS. Signalet TS testes ved Seek-operationer. Det er kun nødvendigt i forbindelse med Double-Side-drev.

WDA : WRITE DATA

Tilslutningen overfører de serielle skrivedata til drevet. Det sker såvel ved skrivning af data i en sektor, som ved de ved formattering nødvendige informationer.

PS0, 1 : PRE SHIFT0/1

Tilslutningen meddeler FDC ved dobbelt density-format (MFM) ved hjælp af en speciel elektronik, hvordan datastrømmen skal skrives på disketten. Der er tre mulige tilstande, EARLY, NORMAL og LATE.

RD : READ DATA

Indgangen indfører de fra disketten læste informationer i FDC. Fra den serielle bitstrøm genvindes de oprindelig skrevne bytes.

RDW : DATA WINDOW

Signalet indfanges af en dataseparator fra de læste data.

WCO : WCO SYNC

Signalet er nødvendigt for WCO's styring i PLL-dataseparator.

MFM : MFM MODE

Tilslutningen signalerer, om controlleren arbejder i single-density format eller i double-density format.

Strømforsyning og taktsignal.

Vcc : +5 volt.

Gennem tilslutningen får FDC sin forsyningsspænding. Spændingen på 5 volt skal ligge indenfor +/- 5%. Den nødvendige strøm andrager max. 150 mA.

GND : GROUND

CLK : CLOCK

Den nødvendige frekvens for FDC. Alt efter drevtype skal frekvens være 4 MHz (ved 5 1/4" og mindre) eller 8 MHz (ved 8").

WCK : WRITE CLOCK

Dette signals frekvens skal vælges i overensstemmelse med det valgte dataformat. Ved MF skal frekvensen være 500 khz og ved MFM 1 mhz. Disse frekvenser bestemmer overførelseshastigheden af data fra og til floppy.

1.9.3. BRUGEN AF FDC 765 I CPC

Desværre har konstruktørerne langtfra udnyttet alle FDC's muligheder. F.eks. kan kun 2 i stedet for 4 mulige drev tilsluttes. Også driften af dobbeltdrev er udelukket, da HEAD-SELECT-signalet ganske vist ledes ud, men ikke benyttes. Endnu værre er det gået med signalet HEAD LOAD, det er slet ikke tilsluttet. Ulykken er ikke så stor, idet brugen af 8"-drev ikke blot er uinteressant for den "gennemsnitlige forbruger", men også fordi yderligere koblinger udelukker brugen i controlleren.

Trods disse begrænsninger er controlleren i forbindelse med brugen af 3"-drev overordentlig gennemtænkt. Med et minimum af hardwareudstyr har man konstrueret en controller, der besidder nogle fremragende data.
På trods af konstruktørernes økonomisering, er det lykkedes at skabe et særdeles godt apparatur. Den udmærket egnede IC SMC 9216, blev derfor indsat som dataseparator i CPC. Det forhold er vigtigt, da dataseparatoren er ansvarlig for den fejlfri læsning af disketten.

Skønt DMA-drift er den enkleste og mest elegante metode, som kan bruges ved tilslutning af floppy-controlleren, har man besluttet at gå en anden vej, formodentlig af økonomiske årsager. Processoren synkroniserer datatransfer på baggrund af Status-Register. De af controlleren producerede interrupts bliver ikke benyttet. I virkeligheden er FDC's interrupt-tilslutning ikke anvendt.

M.h.t. adresser, ligger FDC på Port-adresserne &FB7E og &FB7F. På den første adresse finder vi hoved-statusregistret, den anden hører til dataregistret.

En tredie adresse bruges af controlleren. Ved porten &FA7E finder vi en flip-flop over hvilken drevmotorerne styres. Skriver man et 1 (OUT &FA7E,I i basic) på denne Port, så startes alle drevmotorer op. Skrives derimod et 0, så afbrydes de igen.

1.10. INTERFACES I CPC

Begrebet interface defineres som forbindelsesstedet mellem computer og omgivelser. Således kan såvel en anden computer, en printer eller en person være omgivelser. Efter denne definition vil vi ikke bare beskrive de på apparatets bagside anbragte forbindelser, men også tastatur, monitor- og recordertilslutning.

For brugeren er de vigtigste interfaces tastatur og monitor, da de udgør den umiddelbare forbindelse til computeren. Lad os derfor starte med dem.

1.10.1. TASTATURET

Vi finder ialt 74 taster på CPC-tastaturet. Da begge SHIFT-taster er koblet parallelt, er der altså tale om 73 enkelttaster.

Matrix, i hvilken tasterne er indordnet, består af 8 * 10 ledninger. Men da også joystick aktiveres via matrix, er der ialt 79 tastpositioner til rådighed. Det andet joystick, som er tilsluttet over det førstes bøsning, bliver ikke ført til sin specielle position i matrix, den tilhørende afbryder er anbragt parallelt med tasterne i tastaturet. Et overblik over matrix findes på tegningen 1.10.1.1.

M.h.t. hardware bliver tastaturet aktiveret over 8255 og Sound-chip'en. Det fungerer i hovedtræk som følger. 8255 leverer i portudgangene PC0 til PC3 en halvbyte (nibble), som gennem decoderen 74LS145 bliver forvandlet til en decimal information. Afhængigt af den eksisterende indgangsinformation bliver en af de 10 udgange low. Denne decoder bliver derfor også kaldt BCD-decimal-dekoder. Hvis indgangsinformationen ikke ligger i området fra 0 til 9, ligger alle udgange på high.

Tonechippens parallelindgang er programmeret som indgangsport m.h.t. aktivering af tastaturet. Ligger der intet signal på indgangen, får man ved at læse porten et 1-tal i alle indgange, ialt således &FF.

Er indgangsinformationen fra decoderen &04. I overensstemmelse hermed bliver udgangen pin 5 low. Så længe ingen tilsvarende taste aktiveres, ignorerer Sound-chip'ens udgangsport dette. Et tryk på ESC-tasten har f.eks. ingen virkning sålænge decoderens udgang på pin 8 er high. Aktiveres derimod SPACE-tasten, ændres den fra Sound-chip'en leverede værdi. Nu ligger indgangens bit 7 på GND og vi modtager værdien &7F fra Sound-chip'en.

1.10.1.1 Tastaturmatrix.

Alle taster scannes 50 gange i sekundet. Og værdierne 0 til 9 afleveres efter hinanden i de fire benyttede udgange i port C, og efter hver aflevering testes Sound-chip. Ved registrering af en aktiveret tast, bliver den lagret i en tabel og kan omregnes til tast-nummer og de tilsvarende tegn efter behov.

En stor behagelighed ved tastaturet er, at der mellemlagres op til 20 tegn. I BASIC-programmer kan man foretage input samtidig med, at computeren foretager beregninger eller arbejder med skærmbilledet. Kun ved brug af recorderen og ved listning af basic-programmer, såvel som ved visse disketteoperationer, er der spærret for aflæsning af tastaturet, da den nødvendige tid mangler. Eneste undtagelse er ESC-tasten, som eventuelt skal anvendes ved afbrydelse af operationen.

Iøvrigt er der en mindre ejendommelighed ved tastaturet. Prøv at trykke på tasterne J, K og L samtidig. Til Deres store overraskelse dukker et H op på skærmen. Det sker altid, når man trykker tre taster ned, som danner hjørnerne i en firkant i tastaturmatrix, det samme gælder f.eks. for 123 eller DFG. I dette tilfælde dukker matrix' fjerde tegn op samtidig. Den "fejl" skal man ikke tage så tungt, for samtidig kan aktivering af tasterne 2, 3 og E slutte programmet.

1.10.2. VIDEOTILSLUTNING

CPC's videotilslutning stiller alle signaler for monitorens drift til rådighed. Det er uden betydning, om det er den med computeren leverede monitor, eller en hvilken som helst anden.

Gate array leverer 4 signaler til monitoren. 3 signaler rummer farveinformationer, det 4. signal er en blanding af CRTC-signalerne V-sync og H-sync.

Signalerne blandes gennem modstande og forstærkes af en transistor. Det fremkomne udgangssignal har betegnelsen LUM og tjener som videosignal for den grønne monitor. De almindeligt solgte farvemonitorer med enkel videoindgang kan ved hjælp af signalet fungere til fremstilling af alle farver.

1.10.3. FLOPPYTILSLUTNING

Ikke blot CPC-computeren, men også de udgivne håndbøger er bedre og mere omfattende end mange konkurrerende produkter. Der har imidlertid indsneget sig små fejl her og der. Således omtales bagsidens tilslutninger i CPC 6128-håndbogen som værende de samme som i 664. Det er uden betydning vedrørende de andre tilslutninger, da såvel antallet af tilslutninger som belægning er de samme for begge computere. Kun ved floppy-tilslutning er der en mindre forskel. Tilslutningen i 664 er en 34-polet forbindende printplade. I 6128 er der derimod indbygget en 36-polet centronic-bøsning. Tilslutningerne 1 og 19 er ikke anbragt på denne bøsning. Da tilslutningerne er nummereret anderledes ved centronic-forbindelserne end ved printforbindelserne, stemmer tilslutningernes betegnelse desværre ikke med oplysningerne i håndbogen. Derimod er den fysiske ordning af tilslutningerne identiske med de i håndbogen bragte oplysninger. Man kan selv fremstille en adapter ved hjælp af et 34-polet fladbåndska-

bel og et fladt centronicstik og det tilsvarende stik til drevet. Man kan også tilslutte et 5 1/4" drev. Tilslutningen af floppy-forbindelserne skal ikke yderligere omtales her, da beskrivelsen allerede er bragt i kapitlet om floppy-controller.

1.10.4. RECORDEREN

Selvom der allerede er indbygget et floppy-drev i Deres CPC, råder computeren over en tilslutning til brug for en casette-recorder. For det første bliver det muligt at udnytte den forhåndenværende CPC 464 software, for det andet har man med casetterecorderen et meget prisbilligt backup-medie. Compatibiliteten mellem de forskellige CPC-maskiner tilgodeses også gennem denne tilslutning.

Alle recordere, som findes i handelen, kan tilsluttes. Det er dog vigtigt, at der er et udtag for hovedtelefon, og at recorderen ikke "lirum-larum'er" for meget. For stærke udsving i lydstrømmen forstyrrer timingen, især ved høj overførselshastighed.

Og så skal vi beskæftige os med det anvendte lagringsformat. Principielt kan recorderen og ligeledes disketten kun lagre bit-vis. Enhver byte, der skal lagres, må altså lægges i de enkelte bits og overdrages. I modsætning til disketten, hvor adskillelsen må overtages af controlleren, må adskillelsen af casettedata ske fra processor via software, hvorved den mest værdifulde bit i recorderen sendes først.

Signalet fra 8255 til recorderen er en firkanttone. Hver bit aftegnes som en firkanttone, hvor low-fasen er nøjagtig lige så lang, som high-fasen. En 0-bit kræver halv så lang tid som en 1-bit.

Derfor er oplysningerne om skrivehastigheden kun omtrentlige. Det er åbenbart, at en datablok af lutter 0-bytes kan lagres på halvdelen af den tid, det tager, at lagre en lige så omfangsrig blok, der udelukkende består af &FF. Da fordelingen af 0- og 1-bits imidlertid rent statistisk er ens i en datablok, kan man gå ud fra en angivelse af 1000 baud (1 baud = 1 bit pr. sekund) ved hjælp af SUPER-SAVE (SPEED WRITE 0) og 2000 baud ved hjælp af SPEED - LOAD (SPEED WRITE 1).

Hver casettefil, lige meget om det er programmer eller data, kan max. være 65536 bytes lang. Filerne overføres i blokke, som hver indeholder 2048 bytes. Hver blok indeholder max. 8, 256 bytes store datasegmenter. For hver blok vil der blive overført en header.

Selvom der ikke findes nogen elektrisk forbindelse til forstærkeren og højttaleren, kan man ved at skrue op for lyden høre dataoverførselen.

Blokkens hovede (head) er let at identificere akustisk. I begyndelsen af enhver blok høres en lang og regelmæssig tone.

Den lange, regelmæssige tone er en serie på 2048 1-bits. Efter disse bits følger en enkelt 0-bit, og derpå en synkroniseringsbyte. Den lange følge af 1-bits i starten er nødvendig for computeren, ved bestemmelsen af Write-Baud-Rate. 0-bit fortæller computeren, at indledningen er forbi, og sync-byte bliver nødvendig for at skille head'er-information og data.

Head'er-informationen befinder sig i et 64-byte langt dataområde, som overføres gennem enhver 2K-data-blok. I denne head'er-file findes informationerne om de egentlige data, f.eks. deres navne, om filen er beskyttet eller ej, om det drejer sig om et basicprogram eller ASCII-data, og om, hvor langt programmet er.

Den nøjagtige opbygning af denne head'er er som følger:

Bytes 0-15 : Filnavnet. Er det kortere end 16 bytes, udfyldes med 00.

Bytes 16 : Block-nummer, indeholder nummeret, der vises ved Load og visning af Catalog.

Byte 17 : Hvis der i denne byte står en anden værdi end 00, drejer det sig om den sidste blok i filen.

Byte 18 : Byten indeholder file-typerne. Informationen er skult i de enkelte bits. Bits betydning følger i tilslutning til tabellen.

Bytes 19, 20 : Bytes indeholder længden af blokkens file-information. Er blokken, altså de to K, helt udfyldt, så indeholder bytes værdien &0800. Ved den sidste blok eller ved programmer, der er kortere end 2 K, er antallet af bytes indeholdt i blokken.

Bytes 21, 22 : Bytes angiver load-adressen, fra hvilken data oprindeligt blev skrevet. Ved basic-programmer ses, at adressen 368 decimal ved binær files (maskinsprog) løber i lagerets program.

Byte 23 : Er indholdet i denne byte forskellig fra 0, drejer det sig om den første blok i filen.

Bytes 24, 25 : Indeholder filens længde.

Bytes 26, 27 : Startes et maskinprogram med file-navnet 'RUN, så bliver indholdet af disse head'er-bytes tolket som startadresse for en maskinsprog-file. Programmet bliver altså automatisk startet i den angivne adresse.

De resterende bytes 28 til 63 i head'eren benyttes ikke af systemet og står til rådighed for den erfarne programmør.

Men nu til inddelingen af head'erens bits i byte 8.

Bit 0 : Er den bit sat, så er den tilsvarende file beskyttet. Beskyttede programmer kan skabes i basic med kommandoen 'SAFE "NAME", p'.

Bits 1-3 : Bestemmer filens type. Skønt forskellige file-typer er mulige med 3 bits, bruges kun file-typerne basic-prag (0), binære files (1) og ASCII-data.

Bits 4-7 : I disse bits finder vi normalt et 0, kun ASCII-data har et 1 i bit 4.

Som allerede omtalt fordeles den lagrede information, i de enkelte blokke, videre ud til enkelte segmenter. Hvert segment består af 256 data-bytes og kontrolsum-bytes. Ethvert segments kontrolsum beregnes efter en særlig formel og tillader ved læsning af filen at prøve, om bitsene overføres i den rigtige rækkefølge. Hvis den beregnede kontrolsum ikke stemmer overens med de læste værdier, vises et READ ERROR B.

READ ERROR A viser, at en bit blev læst og fundet for lang for den beregnede værdi for 0 eller 1 bit. Fejlen opstår tit ved læsning af programmer, hvis cassetten "klemte" optagelserne og nu kikser ved load.

Den tredie mulighed for fejl er READ ERROR D. Denne fejl bør kun optræde i sjældne tilfælde, da den signaliserer, at den læste blok er længere end de tilladte 20 til 48 bytes. Det sker kun, hvis man tillader lagring af store værdier i head'er-information.

De kender sikkert basic-ordren "SPEED WRITE par". I overensstemmelse med den anvendte parameter lagres data med gennemsnitlig 1000 eller 2000 baud på cassetten. Dermed opnås dog ikke den øvre grænse for hastigheden. Ved anvendelse af en driftsrutine lader enhver baudhastighed sig indstille mellem 700 baud og ca 3600 baud. Den nødvendige adresse har sit udspring i adressen &BC68. Den afventer de to registerparametre og regulerer således skrivehastigheden.

En værdi sendes til HL-registerpar som bestemmer baud-hastigheden. Formelen til bestemmelse af denne værdi lyder:

Baudhastighed = 333333/halve længde af en 0-bit.

Ved 1000 baud giver det en tid på 666 microsekunder for en 0-bit, en 1-bit er nøjagtig dobbelt så lang.

Den i recorderen anvendte electronic udviser en ejendommelighed. Indlæses skiftevis 0- og 1-bits, så forsøger electronic'en at udjævne tidsforskellen. På den måde bliver en 1-bit kortere, og en 0-bit forekommer som en impuls af længere varighed end forventet. Derfor skal man kompensere forud, ved at sørge for, at 0-bit'en optegnes kortere. De for kompensationen nødvendige tegn overlades til rutinens akkumulator.

I forsøg på at bestemme den højeste halvvejs tilladte skrivehastighed, må man overlade en værdi på 10 til akkumulatoren. For at optegne med 3600 baud, må følgende rutine aktiveres:

```
LD HL,93
LD A,10
CALL &BC68
RET
```

Disse få bytes kan let lagres med følgende linie:

```
10 MEMORY HIMEM - 10
20 FOR I=1 TO 9
30 READ X : POKE HIMEM + I,X
40 NEXT I
50 CALL HIMEM + 1
60 DATA &21,&5D,&00,&3E,&0A,&CD,&68,&BC,&C9
```

Gennemfør trygt variationer over værdierne i HL og akku (den anden og femte værdi i datadelen), for at bestemme den max. optegnelsesfrekvens.Den afhænger af det anvendte cassettemateriale. Men recorderens regelmæssige løb spiller også en væsentlig rolle gennem højere optagelseshastigheders pålidelighed.

Vælges værdierne for lavt, kan CPC ikke tilgodese den krævede tid, og udsender da fejlmeldingen WRITE ERROR A.

Til slut endnu et tip, der gælder såvel for diskette- som for cassetteoperation:

De har sikkert bemærket, at lagrer man meget lange programmer med mange variabler, kan det tage op til 15 minutter at lagre data eller program. Det skyldes, at CPC har behov for et område på 2K for den overdragede lagerblok. Bufferen lægges på den øvre lagergrænse. Er området optaget af variabler, kopieres disse variabler i et andet lagerområde. Den proces kan sammenlignes med den meget frygtede garbadge-collection, som altid optræder, når der ikke er tilstrækkelig plads i lageret til tegnkæder og arrays. Den ventetid, der opstår ved variabelforskydning, kan man imidlertid reducere, idet denne 2K-buffer allerede er lagt og beskyttet i begyndelsen af det forhåndenværende program. En eventuel programstart kan se således ud:

```
10 OPENOUT "DUMMY"
20 MEMORY HIMEM-1
30 CLOSEOUT
40
50 'RESTEN AF PROGRAMMET
```

Fremgangsmåden er der dog kun mening i, når De også arbejder med data i det forhåndenværende program. Er det ikke tilfældet, kan De forlade Dem på de frembragte programdele og afgive ordren CLEAR, for den ønskede lagring. Derved bliver alle tidligere definerede variabeler udløst, og cassettebufferens indførelse finder sted uden nævneværdig tidslængde.

1.10.5. CENTRONIC-INTERFACE

I enhver computer kan man finde noget, som man mener bør forbedres. Ved CPC er det utivlsomt Interface. Skønt mange svagheder og fejl i CPC 464 blev rettet i efterfølgerne 664 og 6128, har man ikke gjort noget ved den mest irriterende svaghed, printer-interface. I disse computere er der stadig kun indbygget et 7-bits-interface. Men da de fleste printere har en 8-bits-indgang, kan mange kommandoer og muligheder i disse printere kun opnås ad omveje, eller slet ikke.

Men lad os betragte den hardware-mæssige opbygning af Interfacet.

Den består hovedsagelig af en 8-delt forbindelse 74LS273. De 8 enkelte latches arbejder som flip-flops. De informationer, som ligger ved indgangen, lagres med en high-low-side i taktindgang-pin 11 og står til rådighed for udgangen indtil et RESET eller en ny programmering foretages, uafhængig af disse regulerende indgangssignaler.

Taktsignalet, hvis high-lowside udvirker lagring af indgangsværdien, vises med et OR-gatter. Gatterudgang bliver low, når begge indgange er low.

Printertilslutningen kaldes ligeledes over indgangsadresseringen. Derfor ligger signalet IOWR* ved OR-gatters indgang, ved den anden indgang ligger adresseledningen A12. Som ved adresseringen af den andet periferikreds, er decodningen altså meget ufuldstændig. På samme måde skal alle adresseledninger, som ikke er nødvendige for decodningen, være high for at kollision med andre anvendte portadresser kan undgås. På den måde finder vi en effektiv indgangsadresse i &EFxx. Printer-latch's indgang er forbundet med processor-databus. Udgangene ligger på printertilslutningen. Kun bit-7 lægges over en som inverter benyttet, NAND-gatter i centronics-indgangen. Bitten fremstiller det nødvendige strobe-signal til printeren. Normalt er dette signal high. Vil computeren derimod sende et tegn til printeren, så lægges den byte, der skal sendes, på dataledningen og straks efter strobesignalet på low.Derved accepteres den sendte byte af printeren.

Forudsætningen er i alle tilfælde, at printerens signal busy er low. Busy-signalets tilstand bliver testet af bit-6 i 8255-indgang B.

Men hvordan frembringes strobe-signalet? Intet er lettere. Enhver byte, som skal overføres, må først aktiveres ved hjælp af &7F og AND. Derved frigøres den øverste bit i byten med sikkerhed. Denne byte bliver ved hjælp af OUT-kommando udsendt af printer-port.

Nu ligger den udsendte bit parat i printeren, strobesignalet er imidlertid stadig high over inverteren. Derfor bliver bit-7 fra den afgivende værdi slutteligt placeret ved hjælp af OR &80 og ligeledes afleveret på printerporten. Intet har altså ændret sig med den overdragende værdi, kun strobesignalet er blevet low ved hjælp af inverter.

Signalet skal igen være high, derfor slettes den øverste bit atter ved hjælp af AND, og byten udsendes endnu engang. Dermed har man sendt en byte fra computeren til printeren.

I basic er overføring til printeren intet problem. Men heller ikke i maskinsprog kan hele "molevitten" beskrives. Der gives flere rutiner ud over, hvad selv det største programudbud kan aftage.

Først er der rutinen med indførelse ved hjælp af &BD2B. Via den kan man frembringe et tegn i printeren. Dette tegn må befinde sig i akkumulatoren. Samtidig undersøger rutinen, om printeren er "busy". Svarer printeren ikke i løbet af 0,4 sekunder, vender rutinen tilbage med slettet carry-flag. Man må da starte et nyt forsøg med de samme tegn. Den rutine anvendes også af basic-interpreter. Ved vellykket overdragelse sættes carry. Derefter kan næste tegn sendes.

Endnu en rutine har sin begyndelse 3 bytes længere henne (&BD2E). Den rutine kan benyttes, når man ønsker at teste printerens tilstand. Er en printer ikke tilsluttet, eller melder printeren "busy", kan der momentant ikke modtages noget tegn, rutinen vender tilbage med sat carry, i andre tilfælde slettes carry.

Den tredie anvendelige rutine (&BD31) bestemmer alle processer, som er nødvendige for anbringelse af et tegn i printeren. Her må programmøren først prøve, om printeren er rede til at modtage, og derefter må han levere det ønskede tegn i akkumulatoren. Forsømmer man at efterprøve tilstanden, forsvinder tegnet eventuelt.

Hvordan disse rutiner lader sig indsætte omtales senere i bogen. Vi vil vise gennem eksempler, hvordan en tekst- og en grafik-hardcopy, samt andre rutiner indsættes.

Der forekommer endnu en ejendommelighed i forbindelse med at tilgodese de forhåndenværende tilsluttede centronics.

Printerindgangens kontaktbeliggenhed opfordrer direkte til at etablere de nødvendige forbindelser, ved at konstruere et fladbåndskabel. Er det en såkaldt "klemmeforbindelse", sa har også teknisk mindre erfarne konstrueret et sådant kabel på 5 - 10 minutter. På den måde lader alle printere sig forbinde med centronic-indgangen i CPC.

Men ved første afprøvning kommer man ud for en overraskelse. Printeren omgås forbløffende nok ødselt med papiret. Ethvert printet tegn efterfølges af en LF.

Det skyldes følgende årsager:

Efter tegn tilføjer CPC tegnfølgen CR/LF, altså ordren for vognretur og line-feed. Derved flyttes papiret yderligere en linie. Dertil kommer, at pin-14 i centronic-tilslutningen i CPC, uden indlysende grund, er forbundet med GND. Det forårsager yderligere en line-feed på de fleste printere.

Det kan afhjælpes ved at afbryde forbindelsen til pin-14 i printeren. Efter at have afbrudt forbindelsen, og at have foretaget en nødvendig indstilling af kontakten i printeren (f.eks. ved epsom), skulle alt være i orden.

1.10.6. TILSLUTNING AF JOYSTICK

Joystick-tilslutningen vil sikkert mest benyttes, så den passer til sit navn: Som indgang for test af et joystick. Over 7 af de 9 tilslutninger, som står til rådighed, lader andre taster eller switches sig også aktivere. Gennem omtalte programmering og ved at give afkald på interrupt og tastaturordrer kan disse 7 tilslutninger tilmed anvendes som udgang. Joystick-tilslutningen er forbundet med Sound-chip'ens bidirektionale indgang og kan også arbejde under de omtalte reducerede betingelser. Under alle omstændigheder er centronic-indgangen den letteste at have med at gøre.

Som allerede omtalt i kap. 1.10.1, så må joystick betragtes som en tast i tastaturet. Derfor er de nødvendige 7 indgange for sound-chip-porten lagt på joystickbøsningen. Dertil er der lagt endnu 2 udgange fra den omtalte BCD-decimal-decoder på bøsningen.

Hvert halvtredsindstyvende sekund scannes hele tastaturet. Derved testes også joystick's tilstand. I basic-programmer står tilstanden af joystick med JOY(nummer)-funktion til rådighed. Tilstanden af joystick kan også bestemmes med INKEY. Men også for assembler-fans er der en mulighed for, let at bestemme joysticks tilstand. Systemrutinen &BB24 returnerer i HL-dobbeltregister den aktuelle tilstand i bits. i H-registret og i akkumulatoren opnår man tilstanden joystick 0, ved brug af disse rutiner. L-registret gælder for joystick 1. Aflåsning af joystick-tasterne følger samme regel, som gælder for JOY (x)-funktionen, bit-0 sættes fremad, bit-1 tilbage, bit-2 til venstre o.s.v.

1.10.7. EXPANSION-CONNECTOR

Det Interface er det mest omfattende i CPC. I den 50-polede printforbindelse findes foruden alle processorens signaler også forskellige styresignaler. Her tilsluttes alle udvidelser af systemet.

Gennem beskrivelser af processoren kender vi allerede betydningen af signalerne 3 til 39. Derfor vil vi begrænse os til at beskæftige os med de resterende tilslutninger.

I pin-1 står lydsignalet endnu engang til rådighed. Signalet er imidlertid kun mono, alle tre kanaler føres direkte hertil.

Pin-2 og pin-49 er forbundet med strømfors ningens GROUND.

Sigalet BUS-RESET* er en ejendommelighed i pin-40. Ved at lægge dette signal på low, genneføres en reset af systemet.

Desværre sletter CPC ved reset hele lageret. Netop derfor er dette signal så virkningsfuldt som "nødbremse" for mislykkede maskinprogrammer gennem sluk-tænd operation.

I pin-41 står det egentlige RESET-signal til rådighed for eksterne udvidelser. *Vær imidlertid opmærksom på at man ikke kan tilgodese alle kredse med dette RESET-signal. 8255 f. eks. kræver signalet inverteret.*

ROMEN* og ROMDIS er interessante signaler. Den i pin-42 i ekspansionskonnektoren liggende ROMEN* signalerer ved hjælp af low et indgreb i den indbyggede 32K-rom. Indgrebet kan standses ved hjælp af high på pin-43, ROMDIS. Herved kan hele den indbyggede rom erstattes med eksterne roms eller eproms.

Tilsvarende dekodning af adresseledningen kan erstatte bestemte områder i den indbyggede rom.

De to signaler RAMRD* og RAMDIS har en lignende funktion ved læseindgreb i den interne ram. Man kan benytte de i pin-43 og -44 liggende signaler til udskiftning af bestemte ram-områder med rom eller ram.

Styringen af de eksterne ram er ikke så helt enkel i CPC. Besværet findes hovedsagelig i, at WR*-signalet til den interne RAM, ikke frembringes af processoren, men af gate-array. Den skriveimpuls kan intet programtrick hindre, så et skriveindgreb i en ekstern RAM vil også adressere den indre RAM. Kun gennem den beskrevne indsats af PAL-element, kan de i CPC 6128 indbyggede 64 Kbyte kaldes på den ønskede måde.

Signalet CURSOR, der er disponibel i pin-46, leveres ved en tilsvarende programmering af video-controlleren. CRTC råder imidlertid over hardware-cursorens muligheder. Efter programmering kommer et firkantsignal med en frekvens på 1,5 eller 3 hz til syne i denne udgang. Men også de konstante high- og low-målere lader sig programmere ved denne tilslutning.

Her ligger efter tilslutning af CPC en konstant low.

LPEN-indgangen (light-pen) i pin-47 er forbundet direkte med CRTC's lys-pen-indgang. IC'en råder over alle nødvendige registre til understøttelse af light-pen.

Imidlertid vil brug af light-pen, især ved højopløsende grafik i CPC være vanskelig, da videocontrolleren leverer MA-adressen for den øjeblikkelige light-pen position, men ikke angiver den aktuelle RA-adresse. Den oplysning er imidlertid nødvendig ved den specielle opbygning af video-ram, når man ønsker at tegne på skærmen med light-pen.

Indgangen pin-48 betegnes EXP* og er forbundet med 8255-port B bit-4. Denne tilslutning kan lægge en ydre udvidelse på massen, og kan på den måde fange operativsystemets opmærksomhed.

Det sidste nævneværdige signal i pin-50, er processorens taktsignal. Signalet kan anvendes som taktsignal for eksterne periferi-IC'er ved anvendelse af en frekvens på 4 mhz.

2. OPERATIVSYSTEMET

Bag dette, for den uindviede fuldstændigt intetsigende ord, skjuler kommandocentralen sig i computeren. Her finder vi den anordning, som sørger for indstillingen af forbindelserne mellem programmet og hardware.

I den sammenhæng ses også basic-interpreter som program, som, over operativsystemet, tager for sig af retterne fra computerens hardware.

Operativsystemets udformning er logisk klart opdelt i såkaldte packs, som hver især har en speciel opgave. Den begynder på MACHINE PACK's niveau, som befinder sig nærmest hardware, og som f. eks. betjener printer-port, Soundregister o.s.v., dernæst over SCREEN PACK, som tager sig af billedskærmen, og til sin tid aktiveres af TEXT PACK og GRAPHICS PACK.

Hver pack er i sig selv et lukket system, og at kommunikationen med andre packs følger eksakt definerede interfaces. Hver pack råder endvidere over et selvstændigt RAM-område som arbejdslager: Krav fra rutinen opfyldes over vektorer i RAM. Sjældent over den direkte ROM-adresse.

Det er nært at antage, at operativsystemet, sikkert p.gr.a. den til rådighed stående korte tid, programmeres yderligere for 1 eller flere pack's vedkommende, idet der hersker enighed om interfaces.

Hvordan det end hænger sammen, så åbner denne selvfølgelige opbygning af selv de mindste enkeltheder i operativsystemets vektorer, uanede og til dato ukendte perspektiver.

Som eksempel kan den mulighed nævnes: Driver for ægte 8-bits-printer (hvordan den så kan tænkes tilsluttet hardwaremæssigt) til at skrive og til at gøre vektoren MC WAIT PRINTER tilgængelig for hele systemet.

Dette tip kan også tjene som advarsel: De kan roligt betjene Dem af operativsystemets rutiner, men kun over vektorerne! I forvejen kan andre (ROM CARTRIDGE) have forrykket nogle vektorer, i den hensigt at lede visse funktioner via egne rutiner.

Med tiden vil De opdage, at nogle programmer kan skrives ved hjælp af minimalt opbud, hvis man betjener sig flittigt af vektorerne. Helt nyt er det, at BASICs aritmetik-rutiner foretages med denne mekanisme, hvilket kan bidrage til at udføre egne beregninger, lige som det, at tilføje egne programmer kan medføre større nøjagtighed o.s.v.

Da det nu har svirret Dem om ørerne med vektorer, så uddyber vi emnet i næste kapitel.

2.1. OPERATIVSYSTEMETS VEKTORER

På de følgende sider fremstilles RAM-adresser, som vil anskueliggøre, hvordan De kan udføre bestemte funktioner via egne programmer. Dels er det komplette rutiner, kopi-

eret i RAM, og som De kan springe ind i, og dels om RST 1 eller RST 5, efterfulgt af inline-adress, som opholder sig i ROM.

I tilgift vil De finde en liste over ROMs rutiner, som vil være til hjælp for hurtigt at genfinde samtlige rutiner.

2.1.1. OPERATIVSYSTEMETS VEKTORER I CPC 664

B900	KL U ROM ENABLE den aktuelle øverste ROM tilsluttes
B903	KL U ROM DISABLE øverste ROM afbrydes
B906	KL L ROM DISABLE nederste ROM afbrydes
B90C	KL ROM RESTORE genfremstilling af tidligere ROM-konfiguration
B90F	KL ROM SELECT udvælgelse af en bestemt øvre ROM
B912	KL CURR SELECTION hvilken øvre ROM er tilsluttet?
B915	KL PROBE ROM undersøgelse af ROM
B918	KL ROM DESELECT genfremstilling af tidligere øvre ROM konfiguration
B91B	KL LDIR brug af ELDIR ved blokerede ROMs
B91E	KL LDDR brug af LDDR ved blokerede ROMs
B921	KL POLL SYNCHRONOUS findes en event af højere prioritet end den forhåndenværende?
B941	RST 7 INTERRUPT ENTRY CONT'D indgreb i hardware-interrupts
B978	KL EXT INTERRUPT ENTRY
B984	KL LOW PCHL CONT'D indgreb i nederste ROM eller RAM
B98A	RST 1 LOW JUMP CONT'D fremkaldelse af en rutine i funktionssystemet eller den underliggende RAM
B9B9	KL FAR PCHL CONT'D
B9C1	KL FAR ICAL CONT'D
B9C7	RST 3 LOW FAR CALL CONT'D enhver rutine i ROM eller ROM kan fremkaldes
BA17	KL SIDE PCHL CONT'D
BA1D	RST 2 LOW SIDE CALL CONT'D bruges ved opkald af en rutine i ekspansions-ROM
BA35	RST 5 FIRM JUMP CONT'D muliggør spring til en rutine i funktionssystemet
BA51	KL L ROM INABLE CONT'D tilslutning af nederste ROM
BA58	KL L ROM DISABLE CONT'D afbrydelse af nederste ROM
BA5F	KL U ROM INABLE CONT'D tilslutning af øverste ROM
BA66	KL U ROM DISABLE CONT'D afbrydelse af øverste ROM
BA70	KL ROM RESTORE CONT'D genfremstilling af tidligere ROM-konfiguration
BA79	KL ROM SELECT CONT'D udvælgelse af en bestemt øvre ROM
BA7E	KL PROBE ROM CONT'D undersøgelse af ROM
BA87	KL ROM DESELECT CONT'D genfremstilling af tidligere øvre ROM-konfiguration
BA9D	KL CURR SELECTION CONT'D hvilken øvre ROM er tilsluttet?

BAA1 KL LDIR CONT'D anvendelse af LDIR ved en blokeret ROM
BAAD KL ROM OFF & KONFIG. SAVE
BAC6 RST 4 RAM LAM CONT'D læsning af indholdet i RAM uafhængig af tilstanden i ROM
BAD7 KL RAM LAM (IX) tilsvarende id a,(ix)

BB00 KM INITIALISE fuldstændig initialisering af tastaturbehandling
BB03 KM RESET tilbageførsel af tastaturbehandling
BB06 KM WAIT CHAR tegn fra tastaturet afventes
BB09 KM READ CHAR forhåndenværende tegn fra tastaturet hentes
BB0C KM CHAR RETURN tegn i tastaturbuffer deponeres for næste indgreb
BB0F KM SET EXPAND indretning af udvidelsesstrengen
BB12 KM GET EXPAND tegn hentes fra udvidelsesstrengen
BB15 KM EXP BUFFER anvisning af lager for udvidelsesstrengen
BB18 KM WAIT KEY afventning af tastaturanslag
BB1B KM READ KEY henter tastenummer, hvis en taste aktiveres
BB1E KM TEST KEY er en taste anslået?
BB21 KM GET STATE hente shift-status
BB24 KM GET JOYSTICK den aktuelle tilstand af joystick undersøges
BB27 KM SET TRANSLATE foretag indførelse i tastaturtabellen (1. niveau)
BB2A KM GET TRANSLATE hente indførsel fra tastaturtabellen
BB2D KM SET SHIFT foretag indførsel i tastaturtabellen (2. niveau)
BB30 KM GET SHIFT hente indførsel fra tastaturtabellen (2. niveau)
BB33 KM SET CONTROL foretag indførsel i tastaturtabellen (3. niveau)
BB36 KM GET CONTROL hente indførsel fra tastaturtabellen (3. niveau)
BB39 KM SET REPEAT angiv gentagelsesfunktion for en bestemt taste
BB3C KM GET REPEAT gentagelsesfunktionen for en bestemt taste er angivet
BB3F KM SET DELAY indsats af tastegentagelse og -hastighed
BB42 KM GET DELAY hente parameter for tastegentagelse og -hastighed
BB45 KM ARM BREAK adgang for break-taste
BB48 KM DISARM BREAK break-taste aflåses
BB4B KM BREAK EVENT udførelse af rutinen ved aktivering af break-taste

BB4E TXT INITIALISE fuldstændig initialisering af text-pack
BB51 TXT RESET tilbageførsel af text-pack
BB54 TXT VDU INABLE tegn kan fremkaldes på billedskærmen
BB57 TXT VDU DISABLE tegnfremstilling standses
BB5A TXT OUTPUT fremstilling eller udførsel af (styre-) tegn
BB5D TXT WR CHAR fremstilling af tegn
BB60 TXT RD CHAR læsning af tegn fra skærmen
BB63 TXT SET GRAPHIC fra- og tilslutning af fremstilling af styretegn
BB66 TXT WIN INABLE fastlæggelse af størrelse af akt. tekstvindue
BB69 TXT GET WINDOW hvilken størrelse har akt. tekstvindue?
BB6C TXT CLEAR WINDOW sletning af akt. tekstvindue
BB6F TXT SET COLUMN anbringelse af cursorens horisontale position
BB72 TXT SET ROW anbringelse af cursorens vertikale position
BB75 TXT SET CURSOR cursor anbringes på positionen
BB78 TXT GET CURSOR aflæsning af den aktuelle cursorposition
BB7B TXT CUR INABLE cursor tillades (anvendelsesprogram)

BB7E TXT CUR DISABLE cursor fastlåses (anvendelsesprogram)
BB81 TXT CUR ON cursor tillades (funktionssystemet)
BB84 TXT CUR OFF cursor fastlåses (funktionssystemet, højere prioritet end BB7B TXT CUR INABLE/BB7E TEXT CUR DISABLE)
BB87 TXT VALIDATE cursor inde i tekstvinduet?
BB8A TXT PLACE/REMOVE CURSOR anbringelse af cursor på billedskærm/ fjernelse af cursor fra billedskærmen
BB8D samme som BB8A
BB90 TXT SET PEN anbringelse af forgrundsfarve
BB93 TXT GET PEN hvilken forgrundsfarve?
BB96 TXT SET PAPER anbringelse af baggrundsfarve
BB99 TXT GET PAPER hvilken baggrundsfarve?
BB9C TXT INVERSE akt. ombytning af for- og baggrundsfarve
BB9F TXT SET BACK transparantmodus ind/ud
BBA2 TXT GET BACK hvilken transparantmodus?
BBA5 TXT GET MATRIX hente adressen for et tegns punktmønster
BBA8 TXT SET MATRIX anbringelse af adressen for et tegns punktmønster
BBAB TXT SET M TABLE anbringelse af startadresse og første tegn af en defineret punktmatrix
BBAE TXT GET M TABLE startadresse og første tegn for anvendelsesmatrix?
BBB1 TXT GET CONTROLS styretegnstabellens adresse hentes
BBB4 TXT STR SELECT valg af tekstvindue
BBB7 TXT SWAP STREAMS to tekstvinduers parametre (farve, vinduesgrænser osv.) ombyttes.

BBBA GRA INITIALISE fuldstændig initialisering af graphic-packs
BBBD GRA RESET tilbageførsel af graphic-packs
BBC0 GRA MOVE ABSOLUTE bevægelse til absolut position
BBC3 GRA MOVE RELATIVE bevægelse i forhold til den øjeblikkelige position
BBC6 GRA ARSK CURSOR hvor befinder akt. graphic-cursor sig?
BBC9 GRA SET ORIGIN fastsættelse af anvendelseskoordinaternes udgangspunkt
BBCC GRA GET ORIGIN udgangspunktet for anvendelseskoordinaterne hentes
BBCF GRA WIN WIDTH fastsættelse af venstre og højre begrænsning af grafik-vinduet
BBD2 GRA WIN HEIGHT fastsættelse af øvre og nedre begrænsning af grafik-vinduet
BBD5 GRA GET W WIDTH højre og venstre begrænsning af grafikvinduet?
BBD8 GRA GET W HEIGHT øvre og nedre begrænsning af grafikvinduet?
BBDB GRA CLEAR WINDOW grafikvinduet ophæves
BBDE GRA SET PEN indførelse af skrivefarven
BBE1 GRA GET PEN hvilken skrivefarve?
BBE4 GRA SET PAPER indførelse af baggrundsfarve
BBE7 GRA GET PAPER hvilken baggrundsfarve?
BBEA GRA PLOT ABSOLUTE indførelse af grafikpunkt (absolut)
BBED GRA PLOT RELATIVE indførelse af grafikpunkt (i forhold til akt. cursor)
BBF0 GRA TEST ABSOLUTE indførelse af punkt (absolut)?
BBF3 GRA TEST RELATIVE punkt indført (i forhold til akt. cursor)?
BBF6 GRA LINE ABSOLUTE en linie fra akt. trækkes til absolut position

BBF9 GRA LINE RELATIVE en linie fra akt. trækkes til relativ afstand
BBFC GRA WR CHAR anbringelse af et tegn på akt. cursor-position

BBFF SCR INITIALISE initialisering af screen-packs
BC02 SCR RESET tilbageførelse af screen-packs
BC05 SCR SET OFFSET indførelse af startadressen for første tegn i forhold til video-ram basisadresse
BC08 SCR SET BASE indførelse af video-rams basisadresse
BC0B SCR GET LOCATION akt. billedskærmstart? (basis + offset)
BC0E SCR SET MODE indførelse af billedskærmmodus
BC11 SCR GET MODE hente og afprøve billedskærmmodus
BC14 SCR CLEAR afbrydelse af billedskærm
BC17 SCR SHAR LIMITS hente størst mulig del- og spalteantal på skærmen (afhængig af modus)
BC1A SCR CHAR POSITION omlægning af de fysiske koordinater i skærmposition
BC1D SCR DOT POSITION formidling af skærmposition for en pixel
BC20 SCR NEXT BYTE en given skærmadresse for at videreregne en tegnposition
BC23 SCR PREV BYTE billedskærmadresse for at tilbageregne en position
BC26 SCR NEXT LINE skærmadresse for at videreregne en del
BC29 SCR PREV LINE skærmadresse for at tilbageregne en del
BC2C SCR INK ENCODE at bringe ink i aflåst form
BC2F SCR INK DECODE bringe ink i uaflåst form
BC32 SCR SET INK at tildele en ink-# farve (farver)
BC35 SCR GET INK farve (farver) fra ink-#?
BC38 SCR SET BORDER indførelse af rammefarve
BC3B SCR GET BORDER rammefarve?
BC3E SCR SET FLASHING indførelse af blinktid
BC41 SCR GET FLASHING blinktiden?
BC44 SCR FILL BOX udfyldning af et angivet vindue med en farve (positionen er skærmadressen, uafhængig af mode)
BC4A SCR CHAR INVERT ombytning af for- og baggrundsfarve ved et tegn
BC4D SCR HW ROLL skærmen en side op eller ned (hardwaremæssigt)
BC50 SCR SW ROLL skærmen en side op eller ned (softwaremæssigt)
BC53 SCR UNPACK forstørrelse af tegn-matrix (for mode 0/1)
BC56 SCR REPACK tegnmatrix overføres atter i originalformen
BC59 SCR ACCESS indførelse af styretegn synlig/usynlig
BC5C SCR PIXELS indførelse af punkt på skærmen
BC5F SCR HORIZONTAL tegne en horisontal linie
BC62 SCR VETICAL tegne en vertikal linie

BC65 CAS INITIALISE initialisering af cassettemanager
BC68 CAS SET SPEED indførelse af skrivehastigheden
BC6B CAS NOISY cassettesignaler ind/ud
BC6E CAS START MOTOR start af cassettemotor
BC71 CAS STOP MOTOR standsning af cassettemotoren
BC74 CAS RESTOR MOTOR fremkaldelse af tidligere motortilstand
BC77 CAS IN OPEN åbning for indkodningsdata

BC7A CAS IN CLOSE lukning for indkodningsdata
BC7D CAS IN ABANDON omgående lukning for indkodningsdata
BC80 CAS IN CHAR tegnlæsning (fra bufferen)
BC83 CAS IN DIRECT hente samtlige data fra lageret
BC86 CAS RETURN sidstlæste tegn returneres til buffer
BC89 CAS TEST EOF slut på data
BC8C CAS OUT OPEN åbning for udgangsdata
BC8F CAS OUT CLOSE lukning for udgangsdata
BC92 CAS OUT ABANDON omgående lukning for udgangsdata
BC95 CAS OUT CHAR skrivning af tegn (i bufferen)
BC98 CAS OUT DIRECT defineret lagerområde indføres i cassette (ikke over bufferen)
BC9B CAS CATALOG fremstiller et cassettekatalog på skærmen
BC9E CAS WRITE skrivning af block
BCA1 CAS READ læsning af block
BCA4 CAS CHECK sammenligning af block på båndet med lagerindholdet

BCA7 SOUND RESET tilbageførelse af sound-pack
BCAA SOUND QUEUE sætte tone i venteposition
BCAD SOUND CHECK er der yderligere plads i køen?
BCB0 SOUND ARM EVENT venteblok i den hensigt at gøre det klart, at der er en ledig plads i køen
BCB3 SOUND RELEASE tilladelse af tone
BCB6 SOUND HOLD øjeblikkelig ophør af tone
BCB9 SOUND CONTINUE tidligere stoppet tone viderebearbejdes
BCBC SOUND AMPL ENVELOPE indretning af indhyldningskurve for lydstyrke
BCBF SOUND TONE ENVELOPE indretning af indhyldningskurven
BCC2 SOUND A ADRESS en indhyldningskurves adresse hentes
BCC5 SOUND T ADRESS en toneindhyldningskurve etableres

BCC8 KL CHOKE OFF tilbageførelse af kernen
BCCB KL ROM WALK tilfældige rom-udvidelser?
BCCE KL INIT BACK romudvidelser etableres
BCD1 KL LOG EXT indførelse af residente udvidelser
BCD4 KL FIND COMMAND opsøgelse af ordrer i alle indførte lagerområder
BCD7 KL NEW FRAME FLY indretning og indførelse af eventblock
BCDA KL ADD FRAME FLY indførelse af eventblock
BCDD KL DEL FRAME FLY fjernelse af eventblock
BCE0 KL NEW FAST TICKER som BCD7
BCE3 KL ADD FAST TICKER som BCDA
BCE6 KL DEL FAST TICKER som BCDD
BCE9 KL ADD TICKER indførelse af ticker-block
BCEC KL DEL TICKER fjernelse af ticker-block
BCEF KL INIT EVENT indførelse af event-block
BCF2 KL EVENT eventblock "kick'es"
BCF5 KL SYNC RESET afbrydelse af sync-pending-kø
BCF8 KL DEL SYNCHRONOUS afbrydelse af en bestemt blok i pending-kø
BCFB KL NEXT SYNC aflever den næste

BCFE KL DO SYNC udførelse af eventrutine
BD01 KL DONE SYNC eventrutine afsluttet
BD04 KL EVENT DISABLE afspærring af normale samtidige data. Presserende samtidige data spærres ikke
BD07 KL EVENT ENABLE sanktion af normale samtidige data
BD0A KL DISARM EVENT fastlåsning af event-block (tælleren negativ)
BD0D KL TIME PLEASE hvor lang tid er der gået?
BB10 KL TIME SET ansættelse af tiden for en i forvejen given værdi

BD13 MC BOOT PROGRAM stil driftssystemet tilbage og tilfør styringen en rutine (hl)
BD16 MC START PROGRAM initialisering af systemet og fremkald af program
BD19 MC WAIT FLYBACK afvent tilbageløb af stråle
BDIC MC SET MODE indførelse af skærmmodus
BD1F MC SCREEN OFFSET indførelse af skærm-offset
BD22 MC CLEAR INKS indførelse af skærmrand og farve af ink
BD25 MC SET INKS indførelse af farve for alle inks
BD28 MC RESET PRINTER tilbageførelse af det indirekte forgreningspunkt for printeren
BD2B MC PRINT CHAR aktivering af tegn når dette er muligt
BD2E MC BUSY PRINTER arbejder printer stadig?
BD31 MC SEND PRINTER aktivering af tegn (vent, til det er muligt)
BD34 MC SOUND REGISTER at forsyne tone-controller med data

BD37 JUMP RESTORE initialisering af alle springvektorer

BD3A KM SET STATE
BD3D KM BUFFER TØMMES
BD40 TXT AKTUEL CURSOR FLAG TIL AKKU
BD43 GRA NN
BD46 GRA PARAM SIKRES
BD49 GRA MASK PARAM SIKRES
BD4C GRA MASK PARAM SIKRES
BD4F GRA KOORD. KONVERTERES logisk i fysiske koordinater
BD52 GRA FILL fillrutine
BD55 SCR ÆNDRING AF SKÆRMSTART
BD58 MC TEGNTILORDNING

Følgende vektorer benyttes i BASIC.

BD5B EDIT

BD5E FLO VARIABLE FRA (DE) TIL (HL) KOPIERES
BD61 FLO INTEGER EFTER FLYDENDE KOMMA
BD64 FLO 4-BYTE-VÆRDI TIL FLO
BD67 FLO FLO TIL INT
BD6A FLO FLO TIL INT
BD6D FLO FIX
BD70 FLO INT

BD73 FLO
BD76 FLO TAL GANGES MED 10↑A
BD79 FLO ADDITION
BD7C FLO RND
BD7F FLO SUBTRAKTION
BD82 FLO MULTIPLIKATION
BD85 FLO DIVISION
BD88 FLO AFHENTNING AF SIDSTE RND-VÆRDI
BD8B FLO SAMMENLIGNING
BD8E FLO FORTEGNSSKIFT
BD91 FLO SGN
BD94 FLO DEG/RAD
BD97 FLO PI
BD9A FLO SQR
BD9D FLO POTENSOPLØFTNING
BDA0 FLO LOG
BDA3 FLO LOG10
BDA6 FLO EXP
BDA9 FLO SIN
BDAC FLO COS
BDAF FLO TAN
BDB2 FLO ATN
BDB5 FLO 4-BYTE-VÆRDI TIL FLO
BDB8 FLO RND INIT
BDBB FLO SET RND SEED

Her begynder de såkaldte INDIRECTIONS. Det er spring i funktionssystemet, som ikke foretages globalt, men individuelt af hver pack, når disses RESET eller INITIALISE gennemløbes.

BDCD TXT DRAW/UNDRAW CURSOR anbringelse/fjernelse af cursor
BDD0 TXT DRAW/UNDRAW CURSOR anbringelse/fjernelse af cursor
BDD3 TXT WRITE CHAR indførelse af et tegn på skærmen
BDD6 TXT UBWRITE CHAR læsning af tegn fra skærmen
BDD9 TXT OUT ACTION tegn leveres til skærmen eller en styrekode eksekveres

BDDC GRA PLOT fremstilling af et punkt på skærmen
BDDF GRA TEST aktuel grafikposition afleveres til ink
BDE2 GRA LIME tegn en linie

BDE5 SCR READ læsning af en pixel og lukning af ink
BDE8 SCR WRITE pixel (s) skrives
BDEB SCR CLEAR skærmen afbrydes

BDEE KM TEST BREAK ESC, SHIFT og CTRL tilbageører hele systemet
BDF1 MC WAIT PRINTER send et tegn til printeren, er denne ikke parat, så vent et øjeblik

BDF4 KM UPDATE KEY STATE MAP

2.1.2. OPERATIVSYSTEMETS VEKTORER I CPC 6128

B900 KL U ROM ENABLE tilslutning af den aktuelle øvre ROM
B903 KL U ROM DISABLE afbrydelse af øvre ROM
B906 KL L ROM ENABLE nedre ROM tilsluttes
B909 KL L ROM DISABLE nedre ROM afbrydes
B90C KL ROM RESTORE gammel ROM-konfiguration genfremstilles
B90F KL ROM SELECT udvælgelse af en bestemt øvre ROM
B912 KL CURR SELECTION hvilken øvre ROM er tilsluttet?
B915 KL PROBE ROM undersøgelse af ROM
B918 KL ROM DESELECT genfremstilling af gammel øvre-ROM-konfiguration
B91B KL LDIR ldir ved blokerede ROM's
B91E KL LDDR lddr ved blokerede ROM's

B921 KL POLL SYNCHRONOUS finder vi en event med højere prioritet end de aktuelt løbende
B941 RST 7 INTERRUPT ENTRY CONT'D indgang for hardware-interrupts
B978 KL EXT INTERRUPT ENTRY
B984 KL LOW PCHL CONT'D spring til nedre ROM eller RAM
B98A RST I LOW JUMP CONT'D tilkald af en rutine i funktionssystemet eller den underliggende RAM
B9B9 KL FAR PCHL CONT'D
B9C1 KL FAR ICALL CONT'D
B9C7 RST 3 LOW FAR CALL CONT'D overalt i RAM eller ROM kan der tilkaldes en rutine
BA17 KL SIDE PCHL CONT'D
BA1D RST 2 LOW SIDE CALL CONT'D bruges ved tilkald af en rutine i ekspansions-ROM
BA35 RST 5 FIRM JUMP CONT'D muliggøre spring til en rutine i funktions-systemet
BA51 KL L ROM ENABLE CONT'D tilslutning af nedre ROM
BA58 KL L ROM DISABLE CONT'D afbrydelse af nedre ROM
BA5F KL U ROM ENABLE CONT'D tilslutning af øvre ROM
BA66 KL U ROM DISABLE CONST'D afbrydelse af øvre ROM
BA70 KL ROM RESTORE CONT'D genfremstilling af gammel ROM-konfiguration
BA79 KL ROM SELECT CONT'D udvælgelse af en bestemt øvre ROM
BA7E KL PROBE ROM CONT'D undersøgelse af ROM
BA87 KL ROM DESELECT CONT'D genfremstilling af gammel øvre ROM-konfiguration
BA9D KL CURR SELECTION CONT'D hvilken øvre ROM er tilsluttet?
BAA1 KL LDIR CONT'D ldir ved blokerede ROM's
BAA7 KL LDDR CONT'D lddr ved blokerede ROM's
BAAD KL ROM OFF & KONFIG. SAVE
BAC6 RST 4 RAM LAM CONT'D læsning af RAM-indhold uafhængig af ROM-tilstand
BAD7 KL RAM LAM (IX) svarer til ld a,(ix)

BB00 KL INITIALISE fuldstændig initialisering af tastaturadministration
BB03 KL RESET tilbageførelse af tastaturbehandling
BB06 KM WAITE CHAR afvente tegn fra tastaturet
BB09 KM READ CHAR hente tegn fra tastaturet, hvis et sådant er til rådighed
BB0C KM CHAR RETURN henlægge tegn i tastaturbuffer klar til næste operation
BB0F KM SET EXPAND udvidelsesstreng klargøres
BB12 KM GET EXPAND hente tegnet fra udvidelsesstreng
BB15 KM EXP BUFFER anvise lager for udvidelsesstrengen
BB18 KM WAIT KEY afvente tastetryk
BB1B KM READ KEY hente tastenummer hvis taste er aktiveret
BB1E KM GET STATE hente shift-status
BB24 KM GET JOYSTICK spørges om joystick's aktuelle tilstand
BB27 KM SET TRANSLATE indførsel i tastaturtabellen (1. plan)
BB2A KM GET TRANSLATE henten fra tastaturtabellen (1. plan)
BB2D KM SET SHIFT indførsel i tastaturtabellen (2. plan)
BB30 KM GET SHIFT henten fra tastaturtabellen (2. plan)
BB33 KM SET CONTROL indførsel i tastaturtabellen (3. plan)
BB36 KM GET CONTROL henten fra tastaturtabellen (3. plan)
BB39 KM SET REPEATE indførelse af gentagelsesfunktion for en bestemt taste
BB3C KM GET REPEATE gentagelsesfunktionen er sat?
BB3F KM SET DELAY ts indførelse af tastegentagelsesindsats og -hastighed
BB42 KM GET DELAY parametre for tastegentagelsesindsats og -hastighed hentes
BB45 KM ARM BREAK breaktaste sanktioneres
BB48 KM DISARM BREAK låsning af breaktaste
BB4B KM BREAK EVENT rutine ved tryk på break-taste udføres
BB4E TXT INITIALISE fuldstændig initialisering af text-packs
BB51 TXT RESET tilbageførelse af text-packs
BB54 TXT VDU ENABLE der kan skrives på skærmen
BB57 TXT VDU DISABLE tegnfremstillingen blokeres
BB5A TXT OUTPUT (styre-)tegn fremstilles eller føres ud
BB5D TXT WR CHAR fremstilling af tegn
BB60 TXT RD CHAR læsning af tegn
BB63 TXT SET GRAPHIC fremstilling af styretegn aktiveres eller afbrydes
BB66 TXT WIN ENABLE størrelse af aktuel textvindue fastlægges
BB69 TXT GET WINDOW hvilken størrelse har aktuel textvinduet?
BB6C TXT CLEAR WINDOW afbrydelse af aktuel textvindue
BB6F TXT SET COLUMN fastsættelse af cursors horisontale position
BB72 TXT SET ROW fastsættelse af cursors vertikale position
BB75 TXT SET CURSOR cursors position fastsættes
BB78 TXT GET CURSOR information om cursors øjeblikkelige position
BB7B TXT CUR ENABLE cursor sanktioneres (anvendelsesprogram)
BB7E TXT CUR DISABLE låsning af cursor (anvendelsesprogram)
BB81 TXT CUR ON cursor sanktioneres (funktionssystemet)
BB84 TXT CUR OFF låsning af cursor (funktionssystem, højere prioritet end BB7B TEXT CUR ENABLE/BB7E TEXT CUR DISABLE)
BB87 TXT VALIDATE befinder cursor sig i textvinduet?
BB8A TXT PLACE/REMOVE CURSOR cursor tilføres/fjernes

BB8D TXT PLACE/REMOVE CURSOR cursor tilføres/fjernes
BB90 TXT SET PEN indførelse af forgrundsfarve
BB93 TXT GET PEN hvilken forgrundsfarve?
BB96 TXT SET PAPER indførelse af baggrundsfarve
BB99 TXT GET PAPER hvilken baggrundsfarve?
BB9C TXT INVERSE ombytning af for- og baggrundsfarve
BB9F TXT SET BACK transparantmodus ind/ud
BBA2 TXT GET BACK hvilken transparantmodus?
BBA5 TXT GET MATRIX hente adresse for et tegns punktmønster
BBA8 TXT SET MATRIX indføring af adresse for et bestemt punktmønster
BBAB TXT SET M TABLE indføring af startadresse og første tegn for punktmatrix
BBAE TXT GET M TABLE startadresse og første tegn for en anvendelsesmatrix?
BBB1 TXT GET CONTROLS adresse hentes fra styretegnsspringtabellen
BBB4 TXT STR SELECT valg af textvindue
BBB7 TXT SWAP STREAMS parametre for to textvinduer ombyttes

BBBA GRA INITIALISE fuldstændig initialisering af graphic-packs
BBBD GRA RESET tilbageførelse af graphic-packs
BBC0 GRA MOVE ABSOLUTE bevægelse til en absolut position
BBC3 GRA MOVE RELATIVE bevægelse i forhold til momentan position
BBC6 GRA ASK CURSOR hvor er graphic-cursor?
BBC9 GRA SET ORIGIN grundliggende punkt for anvendelseskoordinaterne sættes
BBCC GRA GET ORIGIN grundl. punkt for anvendelseskoordinaten hentes
BBCF GRA WIN WIDTH venstre og højre begrænsning af grafik-vindue sættes
BBD2 GRA WIN HEIGHT øvre og nedre begrænsning af grafik-vinduet sættes
BBD5 GRA GET W WIDTH venstre og højre begrænsning af grafik-vinduet?
BBD8 GRA GET W HEIGHT øvre og nedre begrænsning af grafik-vinduet?
BBDB GRA CLEAR WINDOW grafik-vindue afbrydes
BBDE GRA SET PEN indføring af skrivefarve
BBE1 GRA GET PEN hvilken skrivefarve?
BBE4 GRA SET PAPER indføring af baggrundsfarve
BBE7 GRA GET PAPER hvilken baggrundsfarve
BBEA GRA PLOT ABSOLUTE indføring af grafikpunkt (absolut)
BBED GRA PLOT RELATIVE indføring af grafikpunkt (i forhold til cursor)
BBF0 GRA TEST ABSOLUTE indføring af punkt (absolut)?
BBF3 GRA TEST RELATIVE indføring af punkt (i forhold til cursor)?
BBF6 GRA LINE ABSOLUTE linie trækkes fra aktuel til absolut position
BBF9 GRA LINE RELATIVE linie trækkes fra aktuel til relativ afstand
BBFC GRA WR CHAR tegn på den aktuelle graphic-cursor position

BBFF SCR INITIALISE initialisering af screen-packs
BC02 SCR RESET tilbageføring af screen-packs
BC05 SCR SET OFFSET anbringelse af startadresse for 1. tegn i forhold til video-RAMs basisadresse
BC08 SCR SET BASE inføring af video-RAMs basisadresse
BC0B SCR GET LOCATION aktuel skærmstart? (basis + offset)

BC0E SCR SET MODE indføring af billedskærmmodus
BC11 SCR GET MODE skærmmodus hentes og testes
BC14 SCR CLEAR skærmen afbrydes
BC17 SCR CHAR LIMITS størst muligt antal del- og spaltetal hentes (afhængig af modus)
BC1A SCR CHAR POSITION koordinaterne transformeres i skærmen
BC1D SCR DOT POSITION fastsættelse af skærmpositionen for en pixel
BC20 SCR NEXT BYTE en given skærmadresse i den hensigt at arbejde videre med en tegnposition
BC23 SCR PRV BYTE skærmadresse for at tilbageføre en position
BC26 SCR NEXT LINE skærmadresse for at arbejde videre med en del
BC29 SCR PREV LINE skærmadresse for at tilbageføre en del
BC2C SCR INK ENCODE at bringe en ink i kodet form
BC2F SCR INK DECODE at bringe en ink i afkodet form
BC32 SCR SET INK ink-# farve(r) indgrupperes
BC35 SCR GET INK ink-# farve
BC38 SCR SET BORDER indførelse af rammefarve(r)
BC3B SCR GET BORDER rammefarve(r)?
BC3E SCR SET FLASHING indførelse af blinktid
BC41 SCR GET FLASHING blinktiden?
BC44 SCR FILL BOX aktuelt vindue tilføres farve (position m. henblik på tegn, afhængig af mode)
BC47 SCR FLODD BOX aktuelt vindue tilføres farve (positionen er skærm-adresse, uafhængig af mode)
BC4A SCR CHAR INVERT ombytning af for- og baggrundsfarve ved hjælp af et tegn
BC4D SCR HW ROLL skærmen en linie op eller ned (hardwaremæssig)
BC50 SCR SW ROLL skærmen en linie op eller ned (softwaremæssig)
BC53 SCR UNPACK forstørrelse af tegnmatrix (for mode 0/1)
BC56 SCR REPACK tegnmatrix stables atter på originalform
BC59 SCR ACCESS indførelse af styretegn synlig/usynlig
BC5C SCR PIXELS indførelse af punkt på skærmen
BC5F SCR HORIZONTAL horisontal linie trækkes
BC62 SCR VERTICAL vertikal linie trækkes

BC65 CAS INITIALISE initialisering af cassette-manager
BC68 CAS SET SPEED indførelse af skrivehastighed
BC6B CAS NOICY cassettemelding ind/ud
BC6E CAS START MOTOR cassettemotor startes
BC71 CAS STOP MOTOR cassettemotor stoppes
BC74 CAS RESTORE MOTOR genfremstilling af tidligere motortilstand
BC77 CAS IN OPEN åbning af indførelsesdata
BC7A CAS IN CLOSE rækkevis lukning af indførelsesdata
BC7D CAS IN ABANDON øjeblikkelig lukning af indførelsesdata
BC80 CAS IN CHAR læsning af tegn (fra buffer)
BC83 CAS IN DIRECT at tilføre lageret de samlede data
BC86 CAS RETURN sidst læste tegn atter retur til buffer
BC89 CAS TEST EOF slut på data?
BC8C CAS OUT OPEN åbning af udgangsdata

BC8F CAS OUT CLOSE rækkevis lukning af udgangsdata
BC92 CAS OUT ABANDON øjeblikkelig lukning af udgangsdata
BC95 CAS OUT CHAR skrivning af tegn (i buffer)
BC98 CAS OUT DIRECT skrivning af defineret lagerområde på cassetten (ikke over buffer)
BC9B CAS CATALOG udskriver katalog over cassetten på skærmen
BC9E CAS WEITE en block skrives
BCA1 CAS READ en blok læses
BCA4 CAS CHECK sammenligning af block på båndet med indhold af lager

BCA7 SOUND RESET tilbageførelse af sound-packs
BCAA SOUND QUEUE venterækken tilføres tone
BCAD SOUND CHECK er der endnu plads i venterækken?
BCB0 SOUND ARM EVENT at gøre event-block parat, for det tilfælde, at der bliver en ledig plads i venterækken
BCB3 SOUND RELEASE tone tillades
BCB6 SOUND HOLD øjeblikkelig stop for tone
BCB9 SOUND CONTINUE tidligere stoppet tone videreforarbejdes
BCBC SOUND AMPL ENVELOPE indhyldningskurve for lydstyrke etableres
BCBF SOUND TONE ENVELOPE indhyldningskurven for tone etableres
BCC2 SOUND A ADRESS adressen for indhyldningskurven for lydstyrken hentes
BCC5 SOUND T ADRESS adressen for indhyldningskurven for tonen hentes

BCC8 KL CHOKE OFF tilbageførelse af kernen
BCCB KL ROM WALK hvilken som helst ROM-udvidelse?
BCCE KL IMIT BACK tilførelse af ROM-udvidelse
BCD1 KL LOG EXT tilførelse af tilstedeværende udvidelser
BCD4 KL FIND COMMAND opsøgning af ordrer i alle tilsluttede lagerområder
BCD7 KL NEW FRAME FLY indretning af event-block og tilslutning
BCDA KL ADD FRAME FLY tilslutning af event-block
BCDD KL DEL FRAME FLY afbrydelse af event-block
BCE0 KL NEW FAST TICKER som BCD
BCE3 KL ADD FAST TICKER som BCDA
BCE6 KL DEL FAST TICKER som BCCD
BCE9 KL ADD TICKER tilslutning af ticker-block
BCEC KL DEL TICKER afbrydelse af ticker-block
BCEF KL INIT EVENT indstilling af event-block
BCF2 KL EVENT "kicke" event-block
BCF5 KL SYNC RESET frigørelse af sync pending række
BCF8 KL DEL SYNCHRONOUS frigørelse af en bestemt block i pending række
BCFB KL NEXT SYNC kom med den næste
BCFE KL DO SYNC udførelse af event-rutine
BD01 KL DONE SYNC event-rutine afsluttet
BD04 KL EVENT DISABLE spærring af normale sammentræf. Hurtige sammentræf spærres ikke
BD07 KL EVENT ENABLE normale sammentræf tillades
BD0A KL DISARM EVENT event-block fastlåses (tælleren negativ)
BD0D KL TIME PLEASE hvor lang tid er der gået?

BD10 KL TIME SET ansæt tiden til en bestemt værdi

BD13 MC BOOT PROGRAM tilbageføer funktionssystemet og overfør styringen af en til (hl).
BD16 MC START PROGRAM initialisering af systemet og fremkaldelse af et program
BD19 MC WAIT FLY BACK afventning af stråletilbageløb
BD1C MC SET MODE indførelse af skærmmodus
BD1F MC SCREEN OFFSET indførelse af skærmoffset
BD22 MC CLEAR INKS indførelse af skærmrand og indførelse af farve på ink
BD25 MC SET INKS indførelse af farve for alle inks
BD28 MC RESET PRINTER tilbageførelse af det indirekte forgreningspunkt for printer
BD2B MC PRINT CHAR trykning af tegn om muligt
BD2E MC BUSY PRINTER arbejder printer stadig?
BD31 MC SEND PRINTER trykning af tegn (vent, til det er muligt)
BD34 MC SOUND REGISTER tonekontrol forsynes med data

BD37 JUMP RESTORE initialisering af alle spring-vektorer

BD3A KM SET STATE
BD3D KM BUFFER TØMMES
BD40 TXT LFD. CURSOR FLAG TIL AKKU
BD43 GRA NN
BD46 GRA PARAM SIKRES
BD49 GRA MASK PARAM SIKRES
BD4C GRA MASK PARAM SIKRES
BD4F GRA KOORD. KONVERTERES logisk i fysiske koordinater
BD52 GRA FILL
BD55 SCR ÆNDRING AF SKÆRM-START
BD58 MC TEGNGRUPPERING
BD5B KL RAM-KONFIGURATION INDFØRES

BASIC-vektorer

BD5E EDIT

BD61 FLO VARIABLE FRA (DE) TIL (HL) KOPIERES
BD64 FLO INTEGER EFTER FLYDENDE KOMMA
BD67 FLO 4-BYTE-VÆRDI EFTER FLO
BD6A FLO FLO EFTER INT
BD6D FLO FLO EFTER INT
BD70 FLO FIX
BD73 FLO INT
BD76 FLO
BD79 FLO TAL GANGES MED 10↑A
BD7C FLO ADDITION
BD7F FLO REND
BD82 FLO SUBTRAKTION

BD85 FLO MULTIPLIKATION
BD88 FLO DIVISION
BD8B FLO HENTE SIDSTE RND-VÆRDI
BD8E FLO SAMMENLIGN
BD91 FLO FORTEGNSSKIFT
BD94 FLO SGN
BD97 FLO DEG/RAD
BD9A FLO PI
BD9D FLO SQR
BDA0 FLO POTENSOPLØFTNING
BDA3 FLO LOG
BDA6 FLO LOG10
BDA9 FLO EXP
BDAC FLO SIN
BDAF FLO COS
BDB2 FLO TAN
BDB5 FLO ATN
BDB8 FLO 4-BYTE-VÆRDI EFTER FLO
BDBB FLO RND INIT
BDBE FLO SET RND SEED

INDIRECTIONS

BDCD TXT DRAW/UNDRAW CURSOR sæt/slet cursor
BDD0 TXT DRAW/UNDRAW CURSOR sæt/slet cursor
BDD3 TXT WRITE CHAR skriv et tegn på billedskærmen
BDD6 TXT UNWRITE CHAR læs et tegn på billedskærmen
BDD9 TXT OUT ACTION skriv et tegn på skærmen eller udfør styrekoden

BDDC GRA PLOT sæt et punkt på skærmen
BDDF GRA TEST farv den øjeblikkelige grafik-position
BDE2 GRA LINE tegn en linie

BDE5 SCR READ læs en pixel og sluk farven
BDE8 SCR WRITE skriv pixel
BDEB SCR CLEAR slet skærm

BDEE KM TEST BREAK ESC, SHIFT og CTRL laver en total reset af systemet

BDF1 MC WAIT PRINTER sender et tegn til printeren, hvis printeren ikke er klar. Så vent

BDF4 KM UPDATE KEY STATE MAP

2.2. OPERATIVSYSTEMETS RAM

Her følger en udlistning af operativsystemets RAM i det omfang, vi har været i stand til at finde de enkelte adressers funktioner.

En forudsætning for at kunne bruge listningen er, at man har forstået virkningen af manipulationerne. Ellers kan man ved "blinde" indgreb forårsage uhensigtsmæssige og endda for systemet fatale ændringer.

2.2.1. OPERATIVSYSTEMETS RAM I CPC 664

B82D KL Start Int Pending Queue
B831 KL Diverse flag for Int Routine
B832 KL sp save
B8B4 KL Timer Low
B8B6 KL Timer High
B8B8 KL Timerflag
B8B9 KL Start Frame Fly Chain
B8BB KL Start Fast Ticker Chain
B8BD KL Start Ticker Chain
B8BF KL Count For Ticker
B8C0 KL Start Sync Pending Queue
B8C2 KL Løbende Event's prioritet
B8C3 KL Kommando for udførelse
B8D5 KL Løbende exp-ROM
B8D6 KL Tilgang løbende ROM
B8D8 KL Løbende ROM-konfiguration

B7C3 SCR Current Screen Mode
B7C4 SCR Position i en linie
B7C6 SCR High Byte Screen Start
B7C7 SCR Write Indirection
B7D2 SCR Flash Periods
B7D3 SCR Flash Period Color #1
B7D4 SCR Farvehukommelse 2. farve
B7E5 SCR Farvehukommelse 1. farve
B7F6 SCR Flag for løbende farvevalg
B7F8 SCR Current Flash Period
B7F9 SCR Event Block: Set Inks

B6B5 TXT Løbende skærmvindue
B6B6 TXT Start af parametre for vindue 0
B726 TXT Løbende cursorposition (Row,Column)
B728 TXT Vindues flag (0 = hele skærmen)
B729 TXT Løbende vindue, øverst
B72A TXT Løbende vindue, venstre
B72B TXT Løbende vindue, nederst

B72C	TXT Løbende vindue, højre
B72D	TXT Løbende Roll Count
B72E	TXT Løbende Cursor Flag
B72F	TXT Løbende Pen
B730	TXT Løbende Paper
B731	TXT Løbende Background Mode
B733	TXT Graph Char Write Mode (0 = disabled)
B734	TXT User Matrix Char #1
B736	TXT User Matrix Adress
B758	TXT Char Counter Control Buffer
B759	TXT Start, Control Buffer
B763	TXT Control Char Jump Array

B693	GRA X Origin
B695	GRA Y Origin
B697	GRA Løbende X-koordinat
B699	GRA Løbende Y-koordinat
B69B	GRA X-koordinat for GRA vindue, venstre
B69D	GRA X-koordinat for GRA vindue, højre
B69F	GRA Y-koordinat for GRA vindue, foroven
B6A1	GRA Y-koordinat for GRA vindue, nederst
B6A3	GRA Pen
B6A4	GRA Paper
B6A5	GRA Regnebuffer for X-koordinater
B6A7	GRA Regnebuffer for Y-koordinater

B628	KM Exp. String Pointer
B62A	KM Put Back Buffer
B62B	KM Adresse for start af Exp. buffer
B62D	KM Adresse for slut på Exp. buffer
B62F	KM Adresse for start af fri exp. buffer
B631	KM Shift Lock Status
B632	KM Caps Lock Status
B633	KM Delay
B635	KM Key State Map
B637	KM Key 16..23
B62B	KM Joystick 1
B63E	KM Joystick 0
B63F	KM Aktiverede taster under Scan
B649	KM Key Repetition for B63F
B657	KM Break Event Block
B68B	KM Adresse for Key Translation Table
B68D	KM Adresse for Key SHIFT Table
B68F	KM Adresse for Key CTRL Table
B691	KM Adresse på Repeat Tabel

B1ED	SOUND Udført Sound-aktivitet (Efter HOLD)
B1EE	SOUND Løbende Sound aktivitet
B1F8	SOUND Parametre for kanal A

B237 SOUND Parametre for kanal B
B276 SOUND Volumen for indhyldningskurver
B396 SOUND Toneindhyldningskurver

B118 CAS Cassette Message-flag
B11A CAS Input buffer status
B11B CAS Adresse for start på input buffer
B11D CAS Pointer for input buffer
B11F CAS File header input
B15F CAS Output buffer status
B160 CAS Adresse for start output buffer
B162 CAS Pointer output buffer
B164 CAS File header output
B1E9 CAS Cassette speed

B115 EDIT Insert flag

2.2.2. OPERATIVSYSTEMETS RAM I CPC 6128

B82D KL Start Int Pending Queue
B831 KL Diverse flag for Int Routine
B832 KL Sp save
B8B4 KL Timer Low
B8B5 KL Løbende RAM-konfiguration
B8B6 KL Timer high
B8B8 KL Timer flag
B8B9 KL Start Frame Fly Chain
B8BB KL Start Fast Ticker Chain
B8BD KL Start Ticker Chain
B8BF KL Count for Ticker
B8C0 KL Start Sync Pending Queue
B8C2 KL Prioritet løbende Event
B8C3 KL Kommando for udførelse
B8D6 KL Løbende Exp.-ROM
B8D7 KL Tilgang løbende ROM
B8D9 KL Løbende ROM-konfiguration

B7C3 SCR Current Screen-mode
B7C4 SCR Position i en linie
B7C6 SCR High-byte for Screen-start
B7C7 SCR Write Indirection
B7D2 SCR Flash Periods
B7D3 SCR Flash Period Color #1
B7D4 SCR Farvehukommelse 2. Farver
B7E5 SCR Farvehukommelse 1. Farver
B7F6 SCR Flag for løbende farvesæt
B7F8 SCR Current Flash Period

B7F9 SCR Event Block: Set Inks

B6B5 TXT Løbende skærmvindue
B6B6 TXT Start på parametre for vindue 0
B726 TXT Løbende cursorposition (Row,Column)
B728 TXT Vindues-flag (0 = hele skærmen)
B729 TXT Løbende vindue, øverst
B72A TXT Løbende vindue, venstre
B72B TXT Løbende vindue, nederst
B72C TXT Løbende vindue, højre
B72D TXT Løbende Roll Count
B72E TXT Løbende cursor flag
B72F TXT Løbende Pen
B730 TXT Løbende Paper
B731 TXT Løbende Background Mode
B733 TXT Graph Char Write Mode (0 = disabled)
B734 TXT 1. tegn User Matrix
B736 TXT Adresse for User Matrix
B758 TXT Tegn-tæller for Control Buffer
B759 TXT Start Control Buffer
B763 TXT Jump tabel for styretegn

B693 GRA X Origin
B695 GRA Y Origin
B697 GRA løbende X-koordinat
B699 GRA løbende Y-koordinat
B69B GRA X-koordinat. GRA vindue, venstre
B69D GRA X-koordinat. GRA vindue, højre
B69F GRA Y-koordinat. GRA vindue, øverst
B6A1 GRA Y-koordinat, GRA vindue, nederst
B6A3 GRA Pen
B6A4 GRA Paper
B6A5 GRA Regnebuffer for X-koordinater
B6A7 GRA Regnebuffer for Y-koordinater

B628 KM Exp. String Pointer
B62A KM Put Back Buffer
B62B KM Adresse for start på Exp. Buffer
B62D KM Adresse for slut på Exp. Buffer
B62F KM Adresse for start på fri Exp. Buffer
B631 KM Shift Lock Status
B632 KM Caps Lock Status
B633 KM Delay
B635 KM Key Status Map
B637 KM Key 16..23
B62B KM Joystick 1
B63E KM Joystick 0
B63F KM Aktiverede taster under Scan
B649 KM Key Repetition, se B53F

B657 KM Break Event Block
B68B KM Adresse for Key Translation Table
B68D KM Adresse for Key Shift Table
B68F KM Adresse for Key CTRL Table
B691 KM Adresse på Repeat Table

B1ED SOUND Udført lyd (efter HOLD)
B1EE SOUND Løbende SOUND aktivitet
B1F8 SOUND Parametre for kanal A
B237 SOUND Parametre for kanal B
B276 SOUND Parametre for kanal C
B2A2 SOUND Volumen indhyldningskurver
B396 SOUND Tone indhyldningskurver

B118 CAS Cassette Message Flag
B11A CAS I4nput Buffer Status
B11B CAS Adresse start input buffer
B11D CAS Pointer Input Buffer
B11F CAS File Header Input
B15F CAS Output Buffer Status
B160 CAS Adresse for start output buffer
B162 CAS Pointer for output buffer
B164 CAS File Header Output
B1E9 CAS Cassette speed

B115 EDIT Insert Flag

2.3. UDNYTTELSE AF OPERATIVSYSTEMETS RUTINER

CPC indeholder flere hundrede nyttige rutiner eller funktioner, der kan gøre livet lettere for en programmør. Der findes f.eks. rutiner til aflæsning af tastaturet, til udlæsning af tegn på skærmen, forvaltning af vinduer eller til styring af printeren. Trods det store antal funktioner i operativsystemet, er der ting, som CPC ikke har indbygget. Der mangler eksempelvis muligheden for at lave SCREEN DUMP, d.v.s. kopiering af skærmbilledet til printeren.

Det vil vi råde bod på, ved at præsentere hele to eksempler på en sådan HARD COPY. I det første eksempel, drejer det sig om at lave en tekst-hardcopy, der fungerer sammen med enhver tilsluttet printer. Den anden rutine gør det muligt at printe alle tegn inklusiv CPC-grafiktegnene. Grafik i højopløselig form, kan ligeledes printes ud. Vi har valgt at bruge printeren NLQ 401. Denne relativt billige printer, har en forbavsende lighed med EPSON MX/RX/FX, hvad angår forrådet af styretegn. Med andre ord, kan de to rutiner køre med de nævnte printere (også de EPSON-kompatible!).

Når vi når slutningen af dette kapitel, har læseren ikke kun erhvervet sig to hurtige rutiner til hardcopies, men også fået et indblik i udnyttelsen af nogle rutiner i operativsystemet.

For at kunne udskrive skærmbilledets indhold på en printer, skal tegnene læses linievis fra skærmen. Den specielle opbygning af Video-RAM tillader desværre ikke udlæsning af enkelte tegn.

Gennem en omvej via en rutine i operativsystemet kan tegnet derimod læses på den aktuelle cursorposition. Rutinen (TXT RD CHAR, &BB60) lægger tegnet i akk. og sætter carry-flag, hvis der findes et tegn. Er der ikke noget tegn (tilhørende CPC's tegnsæt) på den aktuelle cursorposition, indeholder akk. et 0, flaget er slettet.

Der skal yderligere anvendes en rutine, der kan positionere cursoren. Det er nødvendigt, for at tegnene kan læses i rækkefølge. Funktionen udføres af TXT SET CURSOR, &BB75. Kaldes denne adresse, vil indholdet af register H og register L blive fortolket som henholdsvis kolonne og linie. Herved er positionen øverst til venstre tilgængelig via &0101.

Der opstår dog et mindre problem. Efter at vi gennem aflæsningen har flyttet cursoren rundt på skærmen, skal den sættes på sin oprindelige position igen. Derfor skal vi, inden den første positionering, kende cursorens position og gemme den. Her får vi hjælp af TXT GET CURSOR, &BB78. Efter kald af TXT GET CURSOR indeholder registerparret HL den aktuelle position. Værdien skal gemmes og bruges ved afslutning af hardcopy-funktionen.

De ved TXT RD CHAR fremkomne tegn, skal sendes videre til printeren. Det sker med MC SEND PRINTER, som kaldes ved &BD31. Det i akk liggende tegn vil blive sendt til printerporten med alle de nødvendige handshake-signaler.

Dog forudsætter rutinen, at printeren er klar til at modtage. Det kan undersøges med MC PRINTER BUSY, &BD2E. Er printeren ikke klar eller ikke tilsluttet, vil rutinen levere et sat carry-flag. Her kan man lade rutinen udføre, indtil flaget kommer tilbage slettet. Først på det tidspunkt, kan det ønskede tegn printes.

Det kan forekomme, at man ikke ønsker at lade en hardcopy udføre fuldstændig. Ved et tryk på DEL-tasten, vil rutinen blive afbrudt og printeren standser. Kaldes KM TEST KEY, &BB1E med en gyldig karakterkode i akk, så vil zero-flag være slettet efter RTS.

Nu er vi nået så langt, at alle system-rutiner til opbygningen af en hardcopy-procedure er til stede. Men senest på det tidspunkt, hvor vi programmerer, vil vi opdage, at der ikke er skelnet mellem 20, 40 eller 80 tegn. Man kunne selvfølgelig sige, at hardcopyrutinen kun fungerer i skærm-mode X, men det ville være for nemt.

SCR GET MODE med tilgang i &BC11 oplyser os i akk og i de to flag carry og zero, i hvilken mode, der for øjeblikket arbejdes. Således kan vi lade en hardcopy udføre, med det anvendte tegntal.

Men nu til det egentlige program. Læsere, der ikke er i besiddelse af en assembler, kan anvende det basicprogram, der er listet i slutningen af kapitlet. Programmet indeholder begge hardcopyprogrammer i data-linier.

```
        BB78    GETCRS  EQU     #BB78
        BB75    SETCRS  EQU     #BB75
        BB60    RDCHAR  EQU     #BB60
        BD2E    TSTPTR  EQU     #BD2E
        BD31    PRTCHR  EQU     #BD31
        BC11    GETMOD  EQU     #BC11
        BB1E    TSTKEY  EQU     #BB1E

A100    CD78BB          CALL    GETCRS      ; gem oprindelig cursorposition
A103    2264A1          LD      (OLDPOS),HL
A106    CD11BC          CALL    GETMOD      ; hent skaermmode
A109    17              RLA                 ; antal tegn/20
A10A    3263A1          LD      (MODE),A    ; og gem resultatet
A10D    210101          LD      HL,#0101    ; cursor i oeverste venstre
                                              hjoerne
A110    2266A1          LD      (CRSPOS),HL
A113    3A63A1  LL1     LD      A,(MODE)
A116    47              LD      B,A         ; 1, 2 eller 4 gange
A117    0E14    LOOP    LD      C,20        ; 20 tegn i en linie
A119    C5      LLOOP   PUSH    BC
A11A    E5              PUSH    HL
A11B    CD75BB          CALL    SETCRS      ; placer cursor
A11E    E1              POP     HL
A11F    CD60BB          CALL    RDCHAR      ; og bestem tegnet
A122    C1              POP     BC
A123    3802            JR      C,GOOD      ; gyldigt tegn?
A125    3E20            LD      A,32        ; ellers mellemrum
                                              <BLANK>
A127    CD58A1  GOOD    CALL    PRTOUT      ; udskrivning
A12A    E5              PUSH    HL
A12B    C5              PUSH    BC
A12C    3E42            LD      A,66        ; ESC trykket?
A12E    CD1EBB          CALL    TSTKEY
A131    C1              POP     BC
A132    E1              POP     HL
A133    201C            JR      NZ,EXIT     ; hvis ja, saa afslut
A135    24      WEITER  INC     H
A136    0D              DEC     C
A137    20E0            JR      NZ,LLOOP    ; 20 tegn printet?
A139    10DC            DJNZ    LOOP        ; hele linien?
A13B    3E0D            LD      A,#0D       ; print CR/LF
A13D    CD58A1          CALL    PRTOUT
A140    3E0A            LD      A,#0A
A142    CD58A1          CALL    PRTOUT
A145    2A66A1          LD      HL,(CRSPOS) ; bestem cursorposition for
A148    2C              INC     L           ; naeste raekke
A149    2266A1          LD      (CRSPOS),HL
A14C    7D              LD      A,L
A14D    FE1A            CP      26          ; 25 linier printet?
```

```
A14F  20C2          JR      NZ,LL1
A151  2A64A1  EXIT    LD      HL,(OLDPOS) ; hvis ja, genopret
A154  CD75BB        CALL    SETCRS      ; oprindelig cursorposition
A157  C9            RET                 ; og returner
A158  C5      PRTOUT PUSH   BC
A159  CD2EBD  P1     CALL    TSTPTR      ; printer BUSY?
A15C  38FB          JR      C,P1
A15E  CD31BD        CALL    PRTCHR      ; udskriv tegn
A161  C1            POP     BC
A162  C9            RET
A163  00      MODE  DEFB    0
A164  0000    OLDPOS DEFW   0000
A166  0000    CRSPOS DEFW   0000
```

Via kommentarerne i listningen, skulle programmet være selvforklarende. Den eneste undtagelse, er metoden til beregning af antallet af tegn, der skal udskrives pr. linie. Derfor gennemgår vi den kort her.

Efter kald af SCR GET MODE indeholder akk, alt efter MODE 0, 1 eller 2.

I tilgift, har carry- og zero-flag følgende tilstande:

MODE 0 = carry 1, zero 0
MODE 1 = carry 0, zero 1
MODE 2 = carry 0, zero 0

Kommandoen SLA forskyder indeholdet i akk. en bit mod venstre, hvilket svarer til en multiplikation med 2. Desuden flyttes carry-flagets tilstand over i den nu frie bit 0 i akk. Den "bortfaldne" bit 7 overtages i carry.

I MODE 0 roteres det i akk. liggende 0. Det har ikke nogen indflydelse på indholdet i akk. Da det af SCR GET MODE satte carry-flag flyttes til bit 0 i akk, indeholder akk. tilstanden 1. Denne tilstand betyder, at der printes 20 tegn pr. linie.

I MODE 1 indeholder akk. 1. Carry er slettet i denne MODE. Efter SLA indeholder akk 2. Her printes der 2 gange 20 tegn pr. linie. Det samme gør sig gældende for MODE 2. Dog er resultatet af SLA her 4 i akk, hvilket som bekendt bevirker printning af 4 gange 20 tegn pr. linie.

Noget anderledes er det, når det drejer sig om at skabe en grafik-hardcopy. Her hjælper rutinerne TXT SET CURSOR og TXT RD CHAR ikke noget. Det første, der sker, er at GRA INITIALIZE indkobler grafik-MODE. Herefter bestemmes med GRA GET PAPER baggrundens farvenummer. Med denne værdi sammenlignes alle skærmens punkter. Er farven forskellig (en pixel) fra baggrunden, laves der et punkt på papiret.

Desværre, er CPC kun forsynet med en 7 bits printertilslutning, hvilket afstedkommer visse komplikationer. Det betyder, at vi på een gang kan udskrive 7 punkter, der er ordnet under hinanden. Grafikken på CPC har en vertikal opløsning på 200 punkter. Tallet kan ikke deles med 7 uden at give en rest af pixellinier, der skal behandles for sig. Derimod, er der ingen problemer med udskrivning af forskellige tekstmodes.

Et andet problem med 7 bits porten, udgøres af kommandooverførselen til printeren. Indkoblingen af grafik med ESC L kræver for de 640 pixels pr. linie en angivelse, der slet ikke kan overføres. For at få det krævede antal grafikpunkter, lyder styresekvensen til printeren:

```
PRINT #8,CHR$(27);"L";CHR$(128)CHR$(2)
```

Det er værdien 128, der skaber problemet. Binært udtrykt, svarer tallet til en værdi udtrykt med 8. bit sat. De øvrige bits er alle 0, da den 8. bit anvendes som strobe, og ikke udskrives på printeren.

Vi har omgået problemet på en mindre elegant måde, idet vi kun udskriver 639 punkter horisontalt. Det er et punkt mindre, end der findes på skærmen. Men det gør, at den første værdi, der overføres er tallet 127.

Inden vi kommer til listningen af grafik-hardcopy, skal vi gøre opmærksom på endnu en undtagelse. Selvom skærmen rent fysisk kun viser 200 rasterlinier, regnes der internt i CPC med en vertikal opløsning på 400 punkter, hvilket giver et tydeligt bedre forhold imellem X- og Y-retningen end, hvis man regnede med de 200 linier.

Effekten er nem at se, hvis man prøver at udføre det i håndbogen viste program, til tegning af en cirkel. Cirklen er næsten helt rund. Uden denne korrektion, ville den blive vist som en elipse på siden.

Den selv samme korrektion skal finde sted i vores hard-copy, men blot i den modsatte form. Vi indgiver også grafik-koordinaterne i et raster på 400*640 punkter, men på printeren udskrives kun 200 punkter i vertikal retning, for at formindske fortegningen.

```
A000            ORG   #A000
BBA0   GRINIT   EQU   #BBBA
BBE7   GETPAP   EQU   #BBE7
BBF0   TSTPOI   EQU   #BBF0
BD2B   PRINTO   EQU   #BD2B
BD2E   TSTPTR   EQU   #BD2E
BB1E   TSTKEY   EQU   #BB1E

A000  CDBABB          CALL  GRINIT      ; Grafik-mode On
A003  CDE7BB          CALL  GETPAPER    ; Baggrundsfarve
A006  32BDA0          LD    (PAPER),A
A009  CD6CA0          CALL  INITP       ; Printer til 7/72 tomme
A00C  218F01          LD    HL,399      ; Start øverst til
A012  110000          LD    DE,0        ; venstre
A015  3E07            LD    A,7         ; desværre kun med & nåle
A017  32C0A0          LD    (ANZAHL),A
A01A  CD7CA0  LLOOP   CALL  PRTSEC      ; ESC-sekvens for grafik
A01D  0E00    LL1     LD    C,0         ; C indeholder bitmønster
A01F  3AC0A0          LD    A,(ANZAHL)  ; for printer
A022  47              LD    B,A         ; B=dotlinie-tæller
A023  E5      BYTLP   PUSH  HL
A024  D5              PUSH  DE
A025  C5              PUSH  BC
```

```
A026  CDF0BB          CALL  TSTPOINT  ; bestem pixelfarve på
                                      ; position (HL/DE)
A029  C1              POP   BC
A02A  D1              POP   DE
A02B  21BDA0          LD    HL,PAPER
A02E  BE              CP    (HL)      ; Pixelfarve=baggrundsfarve?
A02F  E1              POP   HL
A030  37              SCF             ; Hvis pixel <> paper, saa
A031  2001            JR    NZ,DOT    ; sæt carry-flag, ellers
A033  A7              AND   A         ; slet carry
A034  CB11    DOT     RL    C         ; forskyd carry til nederste
A036  2B              DEC   HL        ; bit i C-register
A037  2B              DEC   HL        ; HL=HL-2, naeste punkt
A038  10E9            DJNZ  BYTLP     ; og det hele gentages 7 gange
A03A  CDAFA0          CALL  TEST      ; særbehandling af sidste
A03D  79              LD    A,C       ; bitmønster i akk
A03E  CDA6A0          CALL  PRINT     ; og print
A041  13              INC   DE
A042  E5              PUSH  HL
A043  217F02          LD    HL,639    ; en linie printet?
A046  37              SCF
A047  ED52            SBC   HL,DE
A049  E1              POP   HL
A04A  3805            JR    C,NXTROW
A04C  2ABEA0          LD    HL,(Y-MERK)
A04F  18CC            JR    LL1
A051  23      NXTROW  INC   HL        ; saerbehandling af de sidste
A052  7C              LD    A,H       ; 4
A053  B5              OR    L
A054  C8              RET   Z
A055  2B              DEC   HL
A056  110000          LD    DE,0      ; forbered naeste linie
A059  22BEA0          LD    (Y-MERK),HL ; for print
A05C  3E07            LD    A,7
A05E  BD              CP    L         ; sidste 7-er linie?
A05F  20B9            JR    NZ,LLOOP
A061  7C              LD    A,H
A062  B4              OR    H
A063  20B5            JR    NZ,LLOOP
A065  3E04            LD    A,4       ; så kun 4 raekker tilbage
A067  32C0A0          LD    (ANZAHL),A
A06A  18AE            JR    LLOOP
A06C  3E1B    INITP   LD    A,27      ; for NLQ,MX,RX,FX
A06E  CDA6A0          CALL  PRINT
A076  3E07            LD    A,7
A078  CDA6A0          CALL  PRINT
A07B  C9              RET
A07C  E5      PRTESC  PUSH  HL        ; DEL-tasten aktiveret?
A07D  3E42            LD    A,66      ; hvis ja; så afbryd
```

```
A07F CD1EBB          CALL  TSTKEY
A082 E1              POP   HL
A083 2802            JR    Z,NOKEY     ; DEL ikke aktiveret.
A085 E1              POP   HL          ; Manipuler stack
A086 C9              RET               ; for at komme retur
A087 3E0D    NOKEY   LD    A,#0D       ; CR/LF udskrivning
A089 CDA6A0          CALL  PRINT
A08C 3E0A            LD    A,10
A08E CDA6A0          CALL  PRINT
A091 3E1B            LD    A,27        ; ESC L 127 2 = grafik
A093 CDA6A0          CALL  PRINT       ; med 639 punkter.
A096 3E4C            LD    A,76
A098 CDA6A0          CALL  PRINT
A09B 3E7F            LD    A,127
A09D CDA6A0          CALL  PRINT
A0A0 3E02            LD    A,2
A0A2 CDA6A0          CALL  PRINT
A0A5 C9              RET
A0A6 CD2EBD  PRINT   CALL  TSTPTR      ; printer busy?
A0A9 38FB            JR    C,PRINT
A0AB CD2BBD          CALL  PRINTOUT    ; print tegn
A0AE C9              RET
A0AF 3AC0A0  TEST    LD    A,(ANZAHL)  ; behandling af sidste
A0B2 FE07            CP    7           ; 4 dotrækker
A0B4 C8              RET   Z
A0B5 AF              XOR   A
A0B6 CB11            RL    C           ; forskyd 0 tre gange via
A0B8 CB11            RL    C           ; carry i C—register
A0BA CB11            RL    C
A0BC C9              RET
A0BD 00      PAPER   DEFB  0
A0BE 0000    Y-MERK  DEFW  0000
A0C0 00      ANZAHL  DEFB  0
```

Til slut følger den lovede BASIC-loader. Med den kan man anvende programmet, hvis man ikke råder over en monitor eller assembler.

```
100 REM Grafik-Hardcopy fuer cpc mit NLQ/MX/RX/FX
110 REM hardcopy wird mit 'CALL &A000' aufgerufen
120 REM text-hardcopy fuer den cpc
130 REM hardcopy wird mit 'call &a100' aufgerufen
140 FOR i=&A000 TO &A0BF
150 READ byte:POKE i,byte:s=s+byte:NEXT
160 DATA &cd,&ba,&bb,&cd,&e7,&bb,&32,&bd
165 DATA &a0,&cd,&6c,&a0,&21,&8f,&01,&22
170 DATA &be,&a0,&11,&00,&00,&3e,&07,&32
175 DATA &c0,&a0,&cd,&7c,&a0,&0e,&00,&3a
180 DATA &c0,&a0,&47,&e5,&d5,&c5,&cd,&f0
185 DATA &bb,&c1,&d1,&21,&bd,&a0,&be,&e1
```

```
190 DATA &37,&20,&01,&a7,&cb,&11,&2b,&2b
195 DATA &10,&e9,&cd,&af,&a0,&79,&cd,&a6
200 DATA &a0,&13,&e5,&21,&7f,&02,&37,&ed
205 DATA &52,&e1,&38,&05,&2a,&be,&a0,&18
210 DATA &cc,&23,&7c,&b5,&c8,&2b,&11,&00
215 DATA &00,&22,&be,&a0,&3e,&07,&bd,&20
220 DATA &b9,&7c,&b4,&20,&b5,&3e,&04,&32
225 DATA &c0,&a0,&18,&ae,&3e,&1b,&cd,&a6
230 DATA &a0,&3e,&31,&cd,&a6,&a0,&00,&00
235 DATA &00,&00,&00,&c9,&e5,&3e,&42,&cd
240 DATA &1e,&bb,&e1,&28,&02,&e1,&c9,&3e
245 DATA &0d,&cd,&a6,&a0,&3e,&0a,&cd,&a6
250 DATA &a0,&3e,&1b,&cd,&a6,&a0,&3e,&4c
255 DATA &cd,&a6,&a0,&3e,&7f,&cd,&a6,&a0
260 DATA &3e,&02,&cd,&a6,&a0,&c9,&cd,&2e
265 DATA &bd,&38,&fb,&cd,&2b,&bd,&c9,&3a
270 DATA &c0,&a0,&fe,&07,&c8,&af,&cb,&11
275 DATA &cb,&11,&cb,&11,&c9,&00,&00,&00
280 IF s<>23151 THEN PRINT "error in grafik-hc":END
290 PRINT "grafik-hc korrekt geladen"
300 s=0:FOR i=&A100 TO &A162
310 READ byte:POKE i,byte:s=s+byte:NEXT
320 DATA &cd,&78,&bb,&22,&64,&a1,&cd,&11
325 DATA &bc,&17,&32,&63,&a1,&21,&01,&01
330 DATA &22,&66,&a1,&3a,&63,&a1,&47,&0e
335 DATA &14,&c5,&e5,&cd,&75,&bb,&e1,&cd
340 DATA &60,&bb,&c1,&38,&02,&3e,&20,&cd
345 DATA &58,&a1,&e5,&c5,&3e,&42,&cd,&1e
350 DATA &bb,&c1,&e1,&20,&1c,&24,&0d,&20
355 DATA &e0,&10,&dc,&3e,&0d,&cd,&58,&a1
360 DATA &3e,&0a,&cd,&58,&a1,&2a,&66,&a1
365 DATA &2c,&22,&66,&a1,&7d,&fe,&1a,&20
370 DATA &c2,&2a,&64,&a1,&cd,&75,&bb,&c9
375 DATA &c5,&cd,&2e,&bd,&38,&fb,&cd,&31
380 DATA &bd,&c1,&c9
390 IF s<>11873 THEN PRINT "error in text-hc":END
400 PRINT "text-hc korrekt geladen"
```

2.4. INTERRUPTS I OPERATIVSYSTEMET

Den hurtigste og mest effektive måde at reagere på bestemte hændelser inden for et operativsystem, er interrupteknikken. Læseren er sikkert klar over, hvad det dækker over. Hvis ikke, så følger her hovedtrækkene:

En interrupt (dansk: afbrydelse) er i regelen en hardware-foreteelse, hvor et kørende program informerer om en hændelse. Ud fra denne hændelse betinges andre udførelsesmuligheder, d.v.s., at softwaren afhængigt af en tilstand skal udføre en bestemt sekvens, der har prioritet frem for den normale procedure. En sådan aktivitet kan f.eks. være scrollning af skærmbilledet i den tid, hvor skærmens katodestråle er slukket, således at scrollning for brugeren vil synes flimmerfrit.

Interrupteknikken gør det muligt, kun at afbryde det normale programforløb, når det virkeligt er nødvendigt, således at softwaren ikke behøver løbende at holde sig underrettet om, der sker noget eller ej.

Der er selvfølgelig mange måder at integrere en sådan teknik i et styresystem på (nogen påstår, at der er lige så mange muligheder, som der er programmører), men vi må tilstå, at vi aldrig tidligere er stødt på den teknik, der anvendes i CPC.

Det drejer sig her om en raffineret blanding af hardware-interrupt (afbrydelse, hvis påkrævet) og regelmæssig test for interrupt. Hvor påkrævet en bestemt interrupt er, angives i programmørens tilhørende rutine. Kort sagt:

Der findes i maskinen kun en enkelt interrupt, og det er timeren, kaldet Fast Ticker, hver 1/300 sekund laves et interrupt. Resten foregår som vist i det følgende.

Det er nu på tide at indføre nogle nye begreber, som man ofte støder på i ROM-listningen, men fra nu af, altså også her.

1. EVENT betyder HÆNDELSE, hvilket i denne sammnehæng skal opfattes som en slags softwarestyret interrupt.
2. FRAME FLYBACK er det før omtalte stråle-tilbageløb i skærmens billedrør, som sker hvert 1/50 sekund.
3. TICKER er en samling FAST TICKER, der udføres hvert 1/50 sekund.

Det hele går ud på, at programmøren selv kan bestemme, hvor tit et program skal udføre bestemte rutiner på tidspunktet for Frame Flyback, Ticker eller sågar Fast Ticker. Det eneste, operativsystemet skal vide, er adressen på den rutine, der skal udføres. Resten foregår automatisk. Den forberedende information udgøres af EVENT BLOCK, hvori der anføres, hvornår rutinen skal udføres, samtidigt med andre rutiner eller med prioritet o.s.v.

Ved hver Ticker, Fast Ticker eller Frame Fly, kigger operativsystemet efter, om der findes en tilhørende Event Block. Er det tilfældet, vil den blive udført i overenstemmelse med sin prioritet. Der findes iøvrigt altid et antal Event Blocks, f.eks. til ajourføring af farveregister ved Frame Fly.

De til en bestemt hændelse tilordnede blokke er ordnet i kæder bestemt af pointere. Det er således ikke af betydning i hvilken adresse en blok befinder sig på så længe den ikke ligger i de centrale 32 K RAM. Indskrænkningen skyldes, at dette område er det eneste, der kan accesseres uafhængigt af den øvrige ROM-konfiguration.

Skal en sådan blok udføres, drejer det sig om en anden type kæde, den såkaldte PENDING QUEUE. Proceduren kaldes KICK. Pending Queue udføres i slutningen af systemets interruptrutine. Man kunne tænke sig, at en tilstedeværende blok altid skal udføres, så hvorfor indrette en særlig kø til den?

Helt så selvfølgeligt er det dog ikke, for man har hele tiden muligheden for at udsætte behandlingen af en blok, uden at den skal findes i primærkøen. Iøvrigt er der ikke kun denne Timer-interrupt. Hardware-freaks har uden videre adgang til at opbygge en interrupt via Expansion-Bus (asynkront), blot skal der altid findes en tilhørende rutine, der kan KICKe en Event-Block.

Lad os tage et konkret eksempel. Hvad skal man gøre, hvis man vil gøre brug af denne mekanisme?

Der skal selvfølgelig oprettes en Event Block, der skal opbygges som i følgende beskrivelse. Alle Event Blocks har det tilfældes:

Byte 0+1 Kædningsadresse for Pending Que. Dette felt må kun anvendes af operativsystemet!

Byte 2 Tæller.
Så længe tælleren > 0, forbliver Block i Pending Queue, dvs. at rutinen udføres ubetinget indtil den er = 0.
Er tælleren < 0 (dvs. > 127), forbliver Blocken i den aktuelle kæde (Ticker o.s.v.). Selv en KICK fører ikke til udførelse af rutinen, hvilket ellers ville føre til en forhøjelse af tælleren, samt en tilgang ved næste lejlighed.

Byte 3 Klasse.
Bit 0 = 1, Ved hopadresse er det en NEAR ADRESS, dvs. en adresse i den centrale RAM, henholdsvis nedre ROM.
Bit 0 = 0, Hopadressen er en FAR ADRESS, altså i øvre ROM.
Bits 1-4 bestemmer prioriteten.
Bit 5 skal altid være 0!
Bit 6 = EKSPRESS. Denne event har en højere prioritet end normale events med højeste prioritet.
Bit 7 = 1, Asynkron event. Disse events har ikke nogen kø, men placeres ved KICK øjeblikkeligt i Interrupt Pending Queue. Drejer det sig om en EKSPRESS, vil den blive udført øjeblikkeligt, ellers først ved afslutning af interrupt rutinen.
OBS: En rutine for asynkron ekspress-events, skal ligge i den centrale RAM.

Byte 4+5 Rutinens adresse.

Byte 6 ROM selection, hvis hopadressen er af typen FAR, ellers ubenyttet.

Byte 7 Her begynder brugerfeltet, der kan være af vilkårlig længde.
Det kan tjene til overførsel af parametre til rutinen. Ved tilgang til en Event rutine indeholder HL adressen for byte 5 i Event Block, hvis det drejer sig om en NEAR adresse, ellers adressen for byte 6.
Det gør det muligt at lægge flere blokke for den samme rutine, hvilke på baggrund af parametrene kan se, hvilken blok, der kaldes.

Afhængig af eventtype, Ticker, Fast Ticker eller Frame Fly, foranstilles den samlede del endnu 2 eller 6 bytes. I tilfældet af Fast Ticker og Frame Fly er der kun 2 bytes for kædningen (må ikke ændres!) i Fast Ticker List/Frame Fly List.

De 6 bytes for Ticker har følgende betydning:

Bytes 0+1 Kædning for Ticker List (må ikke ændres!).

Bytes 2+3 Tick Count bestemmer, hvor ofte en Ticker skal optræde, inden blokken KICKes 1 gang.

Bytes 4+5 Reload Count angiver med hvilken værdi Tick Count skal loades efter gennemløb.

Efter at man har forsynet sin blok med værdierne, så disse er bekendte (mindst de sidste 5 bytes, af helheden), behøver man kun at forsyne HL med startadressen for blokken (i tilfælde af Ticker, skal også Tick Count flyttes til DE og Reload Count til BC. Til at slette blokken i listen, anvendes rutinen KL DEL TICKER, o.s.v., hvor HL igen skal indeholde adressen på blokken (den, der skal slettes).

Prøv selv og undersøg, hvordan operativsystemet gør det, idet gentagne processer også udføres over Event-mekanikken.

2.5. OPERATIVSYSTEMETS ROM

Læseren har ganske givet, ved opstilling af operativsystemets vektorer (kapitel 2.1), lagt mærke til, hvor lidt operativsystemets RAM adskiller sig fra hinanden hos CPC 664 og CPC 6128. Derfor har vi besluttet os for at vise CPC 6128's operativsystem særlig megen opmærksomhed. Det skal dog ikke forstås således, at CPC 664-ejere nu kan lukke bogen og sætte den op i reolen, men at de må holde øjnene åbne for, om nu også den enkelte rutine omhandler netop deres maskine, dvs. kommentarerne til ROM-listningen. Det kunne jo være, at adresserne er blevet forskudt et par bytes. Da læseren nu har erhvervet sig et godt kendskab til maskinsprog, skulle det ikke volde nogen vanskeligheder at tilpasse rutinerne til CPC 664. Er man ikke helt sikker på sig selv, kan man nøjes med at anvende vektorerne.

I det følgende findes kommentarerne til CPC 6128-operativsystemet. Kommentarerne er ikke altid lige udførlige, hvis man ikke læser dem i sammenhæng, men laver man sig en listning af det aktuelle ROM-område med DISASSEMBLEREN, der er listet i denne bogs tillæg, så vil man sammen med kommentarerne være godt rustet.

2.5.1. KERNEL (KL)

Kernal er, som navnet antyder, kernen i computeren. Den er ansvarlig for styringen af hele forløbet, hvilket vil sige interruptbehandling og de dermed forbundne events (hændelser). Det er også kernals opgave at bearbejde restarts (genopstart), at tilføje (indkoble) ROM-udvidelser og at skifte imellem forskellige hukommelseskonfigurationer.

Det er især rutinerne, der står i forbindelse med eventteknikken, der er af interesse for programmøren.

0000 ********** RST 0 RESET ENTRY

Efter opstart af systemet, begynder processoren at udføre en fuldstændig reset af systemet.

0000 U ROM disable, Mode 1, reset deler
0005 Reset Cont'd

0008 ********** RST 1 LOW JUMP

Tjener til kald af en rutine i operativsystemet eller i den underliggende RAM. Adressen på rutinen, der skal kaldes skal være angivet direkte efter RST-kommandoen.

Da området fra &0000 til &3FFF kun behøver 14 adressebits, bliver bits 14 og 15 benyttet til selektion af PROM eller RAM.

Ved sat bit 15 vælges adresseområdet &C000 til &FFFF og ved slettet bit 14 vælges operativsystemet.

0008 (0430) RST 1 LOW JUMP CONT'D
000B (042A) KL LOW PCHL CONT'D
000E Manipulation af returadresse
000F Svarer til jp (bc)

0010 ********** RST 2 SIDE CALL

Tjener til kald af en rutine i en expansions-ROM (ekstern ROM). RST 2 bruges, hvis et program, der er tilsluttet i form af en ROM-udvidelse, kræver mere end 16 K.

0010 (04C3) RST 2 LOW SIDE CALL CONT'D
0013 (04BD) KL SIDE PCHL CONT'D
0016 Manipulation af returadresse
0017 Svarer til jp (de)

0018 ********** RST 3 FAR CALL

En rutine kan kaldes et eller andet sted i ROM eller RAM. Her skal der følge en 3 bytes parameterblok efter RST-kommandoen.
De første to bytes indeholder adressen på rutinen, der kaldes. Den tredie byte angiver ROM/RAM-status.

0018 (046D) RST 3 LOW FAR CALL CONT'D
001B (045F) KL FAR PCHL CONT'D

0020 ********** RST 4 RAM LAM

RST 4 gør det muligt at udlæse RAM fra et maskinkodeprogram. Den valgte ROM-status har ingen betydning. RST 4-kommandoen erstatter LD A,(HL), hvor HL skal indeholde adressen, der skal læses i.

0020 (056C) RST 4 RAM LAM CONT'D
0023 (0467) KL FAR ICALL CONT'D

0028 ********** RST 5 FIRM JUMP

Gør det muligt at hoppe til en rutine i operativsystemet. Den tilhørende tilgangsadresse skal følge umiddelbart efter RST 5-kommandoen. Inden hop til rutinen, vælges operativsystem-ROM, som igen skal annulleres efter at rutinen forlades.

0028 (04DB) RST 5 FIRM JUMP CONT'D

0030 ********** RST 6 USER RESTART

Bytes &0030 til &0037 står til brugerens disposition. Ved opstart af systemet er default-indstillingen RST 0.

0030 RST 0 efter High Kernel Restore

0038 ********** RST 7 INTERRUPT ENTRY

Tilgang for Hardware-interrupts

0038 (03E7) RST 7 INTERRUPT ENTRY CONT'D
003B EXT INTERRUPT

0040 ********** hertil kopieres i RAM

0040 L ROM disable

0044 ********** Restore High Kernel Jumps

0044 003F
0047 til
0048 0000
0049 kopieres
004A i RAM
004C RST 0 til
004E 0030
0051 Jump
0054 Block
0057 kopieres

005C ********** KL CHOKE OFF

Reset kernel, sletning af Event Wait Queue, o.a.

005D (Løbende ROM-konfiguration)
0060 (Tilgang for løbende ROM)
0064 Slet Firmware-

0066 RAM
0069 til
006B B8CD
006C
0071 Var en ROM om?
0072 ja, hop
007C hvis hl=0
007D Default load
0080 (Løbende Exp.-ROM)
0083 (Løbende ROM-konfiguration)
0086 (Tilgang for løbende ROM)
0089 Parametre for
008C RST 3 loades
0095 FAR CALL
0096 dw B8D7

0099 ********** KL TIME PLEASE

Hvor lang tid er der gået?

009A (Timer high)
009E (Timer low)

00A3 ********** KL TIME SET

Indstilling af tid til en forud bestemt værdi.

00A4 Load akk med 0 og reset flag
00A5 (Timerflag)
00A8 (Timer high)
00AC (Timer low)

00B1 ********** Scan Events

00B1 Timer low
00B4 update
00B5 Timer
00BA Port B
00BC VSYNC?
00BD nej hop
00BF (Start Frame Fly Chain)
00C2 Highbyte til akk
00C3 er akk 0?
00C4 Hvis akk ikke 0, hop til Kick event
00C7 (Start Fast Ticker Chain)
00CA Highbyte til akk
00CB Er akk 0?
00CC Hvis akk ikke 0, hop til Kick Event
00CF Scan Sound Queues

00D2 Count for Ticker
00D9 Update Key State Map
00DC (Start Ticker Chain)
00DF Highbyte til akk
00E0 Er akk 0?
00E1 Akku 0 hop
00E2 diverse Flag for Int. Routine
00E5 Ticker Chain skal endnu
00E7 bearbejdes
00F2 (Start Int Pending Queue)
00F8 diverse Flags for Int. Routine
010A (sp save)
010E Timer low
0114 diverse Flags for Int. Routine
011D (Start Int Pending Queue)
0120 Highbyte til akk
0121 Er akk 0?
0122 Akku 0 hop
0127 (Start Int Pending Queue)
0132 diverse Flags for Int Routine
0135 Ticker Queue pending?
0137 nej hop
013D Ticker Chain bearbejdes
0142 diverse Flags for Int Routine
0145 Highbyte til akk
0146 skal der fortsat ske bearbejdning?
0147 ja hop
0149 Slet flag
014E sp reloades

0153 ********** Kick Event

0158 KL EVENT
015D KL EVENT
0161 Kick Event

0163 ********** KL NEW FRAME FLY

Opret Eventblock og tilføj den.

0166 KL INIT EVENT

016A ********** KL ADD FRAME FLY

Eventblock tilføjes.

016A Start Frame Fly Chain
016D Add Event

0170 ********** KL DEL FRAME FLY

Disable Eventblock.

0170 Start Frame Fly Chain
0173 Delete Event

0176 ********** KL NEW FAST TICKER

Eventblock oprettes og tilføjes (sammenlign KL NEW FRAME FLY).

0179 KL INIT EVENT

017D ********** KL ADD FAST TICKER

Eventblock tilføjes (sammenlign KL ADD FRAME FLY).

017D Start Fast Ticker Chain
0180 Add Event

0183 ********** KL DEL FAST TICKER

Disable Eventblock (sammnelign KL DEL FRAME FLY).

0183 Start Fast Ticker Chain
0186 Delete Event

0189 ********** Ticker Chain bearbejdning.

0189 (Start Ticker Chain)
018C Highbyte til akk.
018D er akk 0?
018E Akku 0 hop
01A4 KL EVENT

01B3 ********** KL ADD TICKER

Tilføj Tickerblock.

01BF Start Ticker Chain
01C2 Add Event

01C5 ********** KL DEL TICKER

Disable Tickerblock.

01C5 Start Ticker Chain
01C8 Delete Event

01D2 ********** KL INIT EVENT

Opret Eventblock.

01E2 ********** KL EVENT

Eventblock "kick".

01E7 Event Cnt >127/<0
01EB Event Cnt >0&<127
01F1 Sync Event tilføjes.

0219 ********** KL DO SYNC

Udfør Eventrutine.

021F (0467) KL FAR INCALL CONT'D

0227 ********** KL SYNC RESET

Sync Pending Queue slettes.

022E ********** Sync Event tilføjes.

022F Prioritet til b.
0230 Kommando for udførelse.
0236 Adresse på den næste
0237 Event Block
0238 flyttes til DE
0240 Løbende prioritet > den fundne.
0241 Prioritet?
0242 Nej hop.

0255 ********** KL NEXT SYNC

Klar til den næste!

0256 (Start Sync Pending Queue)
0259 Highbyte til akk.
025A Er akk 0?
025B Akku 0 hop.
0263 (Prioritet løbende Event)
026B (Prioritet løbende Event)
026E (Start Sync Pending Queue)

0276 ********** KL DONE SYNC

Eventroutine færdigudført.

0276 (Prioritet løbende Event)
027E Sync Event tilføjes.

0284 ********** KL DEL SYNCHRONOUS

Slet bestem blok fra Pending Queue.

0284 KL DISARM EVENT
0287 Start Sync Pending Queue
028A Delete Event

028D ********** KL DISARM EVENT

Eventblock låses (Tæller negativ).

0294 ********** KL EVENT DISABLE

Standsning af de normale samtidige hændelser.
Busy (prioriterede) samtidige Events hindres ikke.

0294 Prioritet for løbende Event.

029A ********** KL EVENT ENABLE

Tillad normale samtidige hændelser.

029A Prioritet for løbende Event.

02A0 ********** KL LOG EXT

Tilføj residente udvidelser.

02B1 ********** KL FIND COMMAND

Søg kommando i alle aktive hukommelsesområder.

02B1 Kommando for udførelse.
02B7 (0553) KL ROM OFF & KOFIGURATION SAVE
02DA (0524) KL PROBE ROM CONT'D
02E4 MC START PROGRAM
02FC (051F) KL ROM SELECT CONT'D
0307 Kommando for udførelse.
0323 (052D) KL ROM DESELECT CONT'D

0326 ********** KL ROM WALK

Finder og initialiserer ROM-udvidelser, således at disse ROMs er tilgængelige.

0328 KL INIT BACK

0330 ********** KL INIT BACK

ROM-udvidelser tilføjes.

0330 Løbende ROM-konfiguration
0339 (051F) KL ROM SELECT CONT'D
0351 (Løbende Exp.-ROM)
0360 KL LOG EXT
0366 (052D) KL ROM DESELECT CONT'D

0379 ********** Add Event

0388 ********** Delete Event

0397 ********** SÆT KL RAM-KONFIGURATION

Her foregår omskiftning mellem de forskellige RAM-konfigurationer i CPC 6128.

0398 Gem register.
0399 Løbende RAM-konfiguration
039E Klargøring for Gate-Array
03A0 Skift mellem RAM-konfiguration
03A3 Genetabler oprindelig registertilstand.

03A6 (0505) KL U ROM ENABLE CONT'D
03A9 (050C) KL U ROM DISABLE CONT'D
03AC (04F7) KL L ROM ENABLE CONT'D
03AF (04FE) KL L ROM DISABLE CONT'D
03B2 (0516) KL ROM RESTORE CONT'D
03B5 (051F) KL ROM SELECT CONT'D
03B8 (0543) KL CURR SELECTION CONT'D
03BB (0524) KL PROBE ROM CONT'D
03BE (052D) KL ROM DESELECT CONT'D
03C1 (0547) KL LDIR CONT'D
03C4 (054D) KL LDDR CONT'D

03C7 ********** KL POLL SYNCHRONOUS

Findes der en EVENT med højere prioritet end den kørende?

03D6 (Start Sync Pending Queue)
03E0 (Prioritet for kørende event).

03E7 ********** RST 7 INTERRUPT ENTRY CONT'D

Sammenlign med RST 7 INTERRUPT ENTRY.

03E9 KL EXT INTERRUPT ENTRY
03F4 L ROM enable

03F6 Scan Events
03FE (diverse flags for Int. Routine)
0418 Sæt oprindelig konfiguration.

041E ********** KL EXT INTERRUPT ENTRY

0423 L ROM disable

042A ********** KL LOW PCHL CONT'D

Hop i nederste ROM eller RAM.

0430 ********** RST 1 LOW JUMP CONT'D

Sammenlign med RST 1 LOW JUMP.

043C Roter akk 4 gange mod venstre.
0445 (0456) Forbered konfiguration og udfør hop.
0456 Gem hopadresse i stack.
0458 Sæt ROM konfiguration.
045E Udfør det forberedte hop.

045F ********** KL FAR PCHL CONT'D

0467 ********** KL FAR ICALL CONT'D

046D ********** RST 3 LOW FAR CALL CONT'D

Sammenlign med RST 3 LOW FAR CALL.

047C ROM# > 252?
047E ja spring
0480 Expansion-ROM
0482 indkobles.
0484 Løbende Expansions-ROM
04A2 L ROM disable
04A4 U ROM enable
04A6 (0456) Forbered konfiguration og udfør hop.
04AF Konfigurer
04B0 tidligere
04B1 ROM-Exp.
04B3 GEnopret.
04B5 (Løbende expansion-ROM)

04BD ********** KL SIDE PCHL CONT'D

04C3 ********** RST 2 LOW SIDE CALL CONT'D

Sammenlign med RST 2 LOW SIDE CALL.

04D5 (Løbende ROM-konfiguration)

04DB ********** RST 5 FIRM JUMP CONT'D

Sammenlign med RST 5 FIRM JUMP.

04E3 L ROM enable
04E5 Load hopadresse.
04EB Udfør hop.
04F0 L ROM disable

04F7 ********** KL L ROM ENABLE CONT'D

Aktiver nederste ROM.

04FA L ROM enable
04FC Hop til gennemførelse.

04FE ********** KL L ROM DISABLE CONT'D

Disble nederste ROM.

0501 L ROM disable
0503 Hop til gennemførelse.

0505 ********** KL U ROM ENABLE CONT'D

Enable øverste ROM.

0508 U ROM enable
050A Hop til udførelse.

050C ********** KL U ROM DISABLE CONT'D

Disable øvre ROM.

050F U ROM disable
0511 Gennemførelse.

0516 ********** KL ROM RESTORE CONT'D

Genetabler oprindelig ROM-konfiguration.

0517 A indeholder
0518 den oprindelige
0519 konfiguration.
051D Hop til gennemførelse.

051F ********** KL ROM SELECT CONT'D

Udvælg en bestemt øvre ROM.

051F (0505) KL U ROM ENABLE CONT'D

0524 ********** KL PROBE ROM CONT'D

Undersøg ROM.

0524 (051F) KL ROM SELECT CONT'D

052D ********** KL ROM DESELECT CONT'D

Genetabler oprindelige øvre ROM-konfiguration.

052F (0516) KL ROM RESTORE CONT'D
0535 Expansion-ROM (# i c)
0537 aktiveres.
0539 Løbende expansion-ROM

0543 ********** KL CURR SELECTION CONT'D

Hvilken øvre ROM er aktiv?

0543 Løbende expansion-ROM

0547 ********** KL LDIR CONT'D

LDIR ved blokeret ROM.

0547 (0553) KL ROM OFF & KONFIG. SAVE

054D ********** KL LDDR CONT'D

LDDR ved blokeret ROM.

054D (0553) KL ROM OFF & KONFIG. SAVE

0553 ********** KL ROM OFF & KONFIG. SAVE

0555 Manipulation af RET-adresse.
0556 Gem oprindelig konfiguration i stack.
0557 Disable
0559 ROM
055D call (hl)
0561 Genopret
0562 oprindelig
0563 konfiguration.
0568 Manipuler RET-adresse.

056C ********** RST 4 RAM LAM CONT'D

Sammenlign RST 4 RAM LAM.

056F Disable
0571 ROM.
0576 Hent byte.
0578 Sæt oprindelig konfiguration.

057D ********** KL RAM LAM (IX)

Svarer til ld a,(ix).

057F Disable
0581 ROM.
0583 Hent byte.
0586 Sæt oprindelig konfiguration.

2.5.2. MACHINE PACK (MC)

Nu kommer vi til den hardwaremæssige del af operativsystemet. Her styres de diverse interfaces og perifere kredse såsom, PIO og PSG. Denne konfiguration har den fordel, at man ved eventuelle ændringer af hardware, kun behøver at tilpasse MACHINE PACK, der kan sammenlignes med BIOS i CP/M.

0591 ********** Reset Cont'd

0592 Control
0597 Port A
059C Port C
05A1 Centronics
05A6 Port B
05AA Isoler LK4
05AC Slut tabel 60Hz
05AF 50Hz? Hop, hvis ikke.
05B1 Slut tabel 50 Hz
05B7 Load Video Register-adresse
05BC Load Video Register

05C5 ********** Tabel 60Hz

3F 28 2E 8E 26 00 19 1E
00 07 00 00 30 00 C0 00

05D5 ********** Tabel 50Hz

```
3F 28 2E 8E 1F 06 19 1B
00 07 00 00 30 00 C0 00

05E5    Koldstart
05E8    i fortsættelses-
05EB    adresse

05ED ******************************** MC BOOT PROGRAM
```

Reset'er operativsystemet og overgiver styringen til en rutine i (HL).

```
05F1    SOUND RESET
05F5    Reset
05F8    periferi.
05FA    KL CHOKE OFF
0601    KM RESET
0604    TXT RESET
0607    SCR RESET
060A    KL U ROM ENABLE CONT'D
060E    jp (hl)
0613    MC START PROGRAM
0617    Load Error.

061C ******************************** MC START PROGRAM
```

Fuldstændig initialisering af system og kald af programmet, hvis startadresse står anført i HL.

```
061C    Efter 966F mødes RET
0620    Sæt Interrupt-Mode 1
0622    Gem registerindhold
0623    Palette Pointer reset
0628    Reset af evt. tilsluttet
062B    perifert udstyr.
062D    Reset
0630    RAM-konfiguration.
0632    Floppy-Motor on/off Flip/Flop
0636    Floppy-Motor off
0638    Kopier &7f9 bytes fra
063B    startadresse &B100 til
063E    måladresse &B101.
0641    Load &B100 med indholdet i akk.
0642    Gennemfør kopiering.
0644    U ROM off & L ROM on
0647    Sreen Mode 1
0649    Reetabler oprindeligt registerindhold.
0652    Restore High Kernel Jumps
0655    JUMP RESTORE
0658    KM INITIALISE
```

065B SOUND RESET
065E SCR INITIALISE
0661 TXT INITIALISE
0664 GRA INITIALISE
0667 CAS INITIALISE
066A MC RESET PRINTER
066F jp (hl)
0674 U ROM initialisering.

0677 ********** Koldstart.

067A TXT SET CURSOR
067D Udskriv firmanavn.
0680 Udskriv meddelelser.
0683 Opstartmelding.
0686 Udskriv meldinger.

0688 ********** Opstartsmeddelelse.

0689 128K
068E Microcomputer
069D (v3)
06A4 Copyright
06B0 c1985
06B6 Amstrad
06BE Consumer
06C7 Electronics
06D3 plc
06D9 and
06DD Locomotive
06E8 Software
06F1 Ltd

06F9 LOAD ERROR-meddelelse.

06FC ********** Udskriv meddelelser.

0700 TXT OUTPUT
0703 Udskriv meddelelser.

0705 ********** LOAD ERROR-meddelelse.

0705 ***
0709 PROGRAM
0711 LOAD
0716 FAILED
071E ***

0725 Port B
0728 LK1...3 isoleres.
072A /2
072B Firmanavn.

0738 ********** Firmanavne.

0738 Arnold
073F Amstrad
0747 Orion
074D Schneider
0757 Awa
075B Solavox
0763 Saisho
076A Triumph
0772 Isp

0776 ********** MC SET MODE

Sæt skærm-mode.

0776 Mode>2?
0778 Hvis ja, retur hop.
077B Mode Bits
077D reset'es.
0780 Sæt ny
0781 mode.

0786 ********** MC CLEAR INKS

Sæt skærmkant og alle INKs til en farve.

0786 Læg indholdet af HL i stack.
0787 Load HL med &0000.
078A Seks bytes videre.

078C ********** MC SET INKS

Udskriv (sæt) alle INKs og skærmkant.

078C Læg indholdet af HL i stack.
078D Load HL med &0001
0793 Border Color.
0796 Udskriv farve (sæt farve).
079A Adresse for Ink 0
079C Udskriv farve.
07A4 Load alle farveadresser.

07AA ********** Udskriv farver.

07AA Palette Pointer
07AD Slet bits 5,6 og 7 i akk.
07AF Sæt bit 6.
07B1 Farve.

07B4 ********** MC WAIT FLYBACK

Afvent retur-scan (katode-stråle).

07B6 Port B
07BA VSYNC?
07BB Hvis ikke, vent.

07C0 ********** MC SCREEN OFFSET

Sæt skærm-offset.

07C3 Slet alle bits undtagen 4 og 5.
07C8 Slet alle bits undtagen 0 og 1.
07CE Video Contr Register 12
07D1 Skærmbillede Start Hi.
07D5 Register 13
07DC Skærmbillede Start Lo.

07E0 ********** MC RESET PRINTER

Reset indirekte forgrening for printer.

07E0 Startadresse.
07E3 måladresse.
07E6 Kopier
07E9 21 bytes.
07EE Move (hl+3) til ((hl+1)),cnt = (hl)
07F1 db 03 3 bytes
07F2 dw BDF1 måladresse.
07F4 MC WAIT PRINTER

07F7 ********** Konverter nationale tegn.

Den følgende tabel blev kopieret til RAM af MC RESET PRINTER (måladresse &B804). Den første byte i tabellen angiver tabellens længde i bytes. Så følger et antal byte-par, hvor det første angiver den interne tastaturkode, og det andet det tilordnede tegn. Ændres denne tabel i RAM, er det muligt at manipulere de tilordnede koder, hvorved man f.eks. kan lave dansk karaktersæt på tastaturet.

07F7 db 0A Antal bytes.

07F8 db A0 Intern tastaturkode.
07F9 db 5E Tilordnet tegn : ↑

```
07FA    db A1   Intern tastaturkode.
07FB    db 5C   Tilordnet tegn      : \

07FC    db A2   Intern tastaturkode.
07FD    db 7B   Tilordnet tegn      : {

07FE    db A3   Intern tastaturkode.
07FF    db 23   Tilordnet tegn      : #

0800    db A6   Intern tastaturkode.
0801    db 40   Tilordnet tegn      : @

0802    db AB   Intern tastaturkode.
0803    db 7C   Tilordnet tegn      : |

0804    db AC   Intern tastaturkode.
0805    db 7D   Tilordnet tegn      : }

0806    db AD   Intern tastaturkode.
0807    db 7E   Tilordnet tegn      : ~

0808    db AE   Intern tastaturkode.
0809    db 5D   Tilordnet tegn      : ]

080A    db AF   Intern tastaturkode.
080B    db 5B   Tilordnet tegn      : [
```

080C ********** MC TEGNTILORDNING

Her sker manipulation af karaktererne (nationale).

```
080C    hl: Startadresse for den nye tegntabel (RAM).
0812    Konvertering af nationale tegn (RAM).
0817    KL LDIR CONT'D
```

081B ********** MC PRINT CHAR

Udskriver tegnet i A på Centronics-porten (Den parallelle printerport). Efter returnering fra rutinen er carry sat, hvis tegnet blev sat som det skulle.

```
0826    Nationalt tegn?
0828    Hop, hvis ikke.
082F    MC WAIT PRINTER
```

0835 ********** MC WAIT PRINTER

Send et tegn til printeren. Hvis denne ikke er parat til at modtage, så vent en periode.

```
0838    MC BUSY PRINTER
083B    MC SEND PRINTER

0844 ********************************** MC SEND PRINTER
```

Sender et tegn til printeren, der ikke må være BUSY.

```
0847    Byte uden strobe
0849    til printer.
084E    Strobe On.
0853    Strobe Off.

0858 ********************************** MC BUSY PRINTER
```

Undersøg om printeren er BUSY.

```
085A    Port B
085E    Printer Busy
085F    Til carry.

0863 ******************************** MC SOUND REGISTER
```

Forsyn controlleren med data. MC SOUND REGISTER er af særlig interesse for musikfans. Uden at skulle plages med den relativt komplicerede overførsel til PSG, behøver man blot at give akk det ønskede registernummer og flytte data- byten til C.

```
0864    Port A
0866    Sound Register#
0868    Port C
086A    Sound Chip
086C    Ved input
086E    & Strobe On
0872    Strobe Off
0874    Port A
0876    Sound data
0878    Port C
087D    Indlæg
087F    data.

0883 ************************************** Scan Keyboard

0883    Port A
0886    Sound Register 14 (Keyboard X Input)
0888    Port C
0891    Strobe On
0893    Strobe Off
0896    Port A&B = Input
0898    Control
089D    Port C
```

```
089F    Keyboard Y Output und X Input
08A1    Port A
08A3    Data (Keyboard X Input) til akk.
08AC    Keyboard Y+1
08B0    alle Y-ledninger bearbejdet?
08B2    Nej, næste ledning.
08B5    Port A Output
08B7    Control
08BA    Port C
```

2.5.3. JUMP RESTORE (JRE)

Denne PACK har, som eneste opgave, at sætte MAIN-JUMP-adresserne til deres default-værdier. Her foranstilles FIRM JUMPS et RST 1 og ARITHMETIK JUMPS et RST 5.

Hvis man efter heftig programmering, får den opfattelse, at man har flyttet alt for mange vektorer, kan man så at sige "trække i nødbremsen" ved at hoppe til JUMP RESTORE. Det er iøvrigt altid tilrådeligt, at man kalder denne rutine efter at have erstattet flere rutiner i operativsystemet, med sine egne.

```
08BD ************************************ JUMP RESTORE

08BD    Main Jump Adress
08C0    Pointer på vektorområde i RAM.
08C3    b: antal vektorer  c: Code RST 1
08C6    Kopier vektortabel.
08C9    b: Antal vektorer  c: Code RST 5
08CD    Gem koden RST.
08CE    pointer+1 (RAM)
08CF    en byte fra ROM til RAM.
08D1    bc til værdi inden LDI.
08D2    Komplementer akk.
08D3    Forskyd bit 5
08D4    til bit 7 og
08D5    isoler den.
08D7    Hent bits 0-6 fra adresse High-Byte.
08D8    Gem High-Byte.
08D9    Pointer+1 (RAM)
08DA    Pointer+1 (ROM)
08DB    Fortsæt så længe, det er påkrævet.
08DD    Retur fra underrutinen.

08DE ************************************ Main Jump Adress

08DE    dw 1B5C KM INITIALISE
08E0    dw 1B98 KM RESET
08E2    dw 1BBF KM WAIT CHAR
```

```
08E4    dw 1BC5 KM READ CHAR
08E6    dw 1BFA KM CHAR RETURN
08E8    dw 1C46 KM SET EXPAND
08EA    dw 1CB3 KM GET EXPAND
08EC    dw 1C04 KM EXPAND BUFFER
08EE    dw 1CDB KM WAIT KEY
08F0    dw 1CE1 KM READ KEY
08F2    dw 1E45 KM TEST KEY
08F4    dw 1D38 KM GET STATE
08F6    dw 1DE5 KM GET JOYSTICK
08F8    dw 1ED8 KM SET TRANSLATE
08FA    dw 1EC4 KM GET TRANSLATE
08FC    dw 1EDD KM SET SHIFT
08FE    dw 1EC9 KM GET SHIFT
0900    dw 1EE2 KM SET CONTROL
0902    dw 1ECE KM GET CONTROL
0904    dw 1E34 KM SET REPEAT
0906    dw 1E2F KM GET REPEAT
0908    dw 1DF6 KM SET DELAY
090A    dw 1DF2 KM GET DELAY
090C    dw 1DFA KM ARM BREAK
090E    dw 1E0B KM DISARM BREAK
0910    dw 1E19 KM BREAK EVENT

0912    dw 1074 TXT INITIALISE
0914    dw 1984 TXT RESET
0916    dw 1459 TXT VDU ENABLE
0918    dw 1452 TXT VDU DISABLE
091A    dw 13FE TXT OUTPUT
091C    dw 1335 TXT WR CHAR
091E    dw 13AC TXT RD CHAR
0920    dw 13A8 TXT SET GRAPHIC
0922    dw 1208 TXT WIN ENABLE
0924    dw 1252 TXT GET WINDOW
0926    dw 154F TXT CLEAR WINDOW
0928    dw 115A TXT SET COLUMN
092A    dw 1165 TXT SET ROW
092C    dw 1170 TXT SET CURSOR
092E    dw 117C TXT GET CURSOR
0930    dw 1286 TXT CUR ENABLE
0932    dw 1297 TXT CUR DISABLE
0934    dw 1276 TXT CUR ON
0936    dw 127E TXT CUR OFF
0938    dw 11CA TXT VALIDATE
093A    dw 1265 TXT PLACE/REMOVE CURSOR
093C    dw 1265 TXT PLACE/REMOVE CURSOR
093E    dw 12A6 TXT SET PEN
0940    dw 12BA TXT GET PEN
0942    dw 12AB TXT SET PAPER
```

```
0944    dw 12C0 TXT GET PAPER
0946    dw 12C6 TXT INVERSE
0948    dw 137B TXT SET BACK
094A    dw 1388 TXT GET BACK
094C    dw 12D4 TXT GET MATRIX
094E    dw 12F2 TXT SET MATRIX
0950    dw 12FE TXT SET M TABLE
0952    dw 132B TXT GET M TABLE
0954    dw 14D4 TXT GET CONTROLS
0956    dw 10E4 TXT STR SELECT
0958    dw 1103 TXT SWAP STREAMS

095A    dw 15A8 GRA INITIALISE
095C    dw 15D7 GRA RESET
095E    dw 15FE GRA MOVE ABSOLUTE
0960    dw 15FB GRA MOVE RELATIVE
0962    dw 1606 GRA ASK CURSOR
0964    dw 160E GRA SET ORIGIN
0966    dw 161C GRA GET ORIGIN
0968    dw 16A5 GRA WIN WIDTH
096A    dw 16EA GRA WIN HEIGHT
096C    dw 1717 GRA GET W WIDTH
096E    dw 172D GRA GET W HEIGHT
0970    dw 1736 GRA CLEAR WINDOW
0972    dw 1767 GRA SET PEN
0974    dw 1775 GRA GET PEN
0976    dw 176E GRA SET PAPER
0978    dw 177A GRA GET PAPER
097A    dw 1783 GRA PLOT ABSOLUTE
097C    dw 1780 GRA PLOT RELATIVE
097E    dw 1797 GRA TEST ABSOLUTE
0980    dw 1794 GRA TEST RELATIVE
0982    dw 17A9 GRA LINE ABSOLUTE
0984    dw 17A6 GRA LINE RELATIVE
0986    dw 1940 GRA WR CHAR

0988    dw 0ABF SCR INITIALISE
098A    dw 0AD0 SCR RESET
098C    dw 0B37 SCR SET OFFSET
098E    dw 0B3C SCR SET BASE
0990    dw 0B56 SCR GET LOCATION
0992    dw 0AE9 SCR SET MODE
0994    dw 0B0C SCR GET MODE
0996    dw 0B17 SCR MODE CLEAR
0998    dw 0B5D SCR CHAR LIMITS
099A    dw 0B6A SCR CHAR POSITION
099C    dw 0BAF SCR DOT POSITION
099E    dw 0C05 SCR NEXT BYTE
09A0    dw 0C11 SCR PREV BYTE
```

```
09A2    dw 0C1F SCR NEXT LINE
09A4    dw 0C39 SCR PREV LINE
09A6    dw 0C8E SCR INK ENCODE
09A8    dw 0CA7 SCR INK DECODE
09AA    dw 0CF2 SCR SET INK
09AC    dw 0D1A SCR GET INK
09AE    dw 0CF7 SCR SET BORDER
09B0    dw 0D1F SCR GET BORDER
09B2    dw 0CEA SCR SET FLASHING
09B4    dw 0CEE SCR GET FLASHING
09B6    dw 0DB9 SCR FILL BOX
09B8    dw 0DBD SCR FLOOD BOX
09BA    dw 0DE5 SCR CHAR INVERT
09BC    dw 0E00 SCR HW ROLL
09BE    dw 0E44 SCR SW ROLL
09C0    dw 0EF9 SCR UNPACK
09C2    dw 0F2A SCR REPACK
09C4    dw 0C55 SCR ACCESS
09C6    dw 0C74 SCR PIXELS
09C8    dw 0F93 SCR HORIZONTAL
09CA    dw 0F9B SCR VERTICAL

09CC    dw 24BC CAS INITIALISE
09CE    dw 24CE CAS SET SPEED
09D0    dw 24E1 CAS NOISY
09D2    dw 2BBB CAS START MOTOR
09D4    dw 2BBF CAS STOP MOTOR
09D6    dw 2BC1 CAS RESTORE MOTOR
09D8    dw 24E5 CAS IN OPEN
09DA    dw 2550 CAS IN CLOSE
09DC    dw 2557 CAS IN ABANDON
09DE    dw 25A0 CAS IN CHAR
09E0    dw 2618 CAS IN DIRECT
09E2    dw 2607 CAS RETURN
09E4    dw 2603 CAS TEST EOF
09E6    dw 24FE CAS OUT OPEN
09E8    dw 257F CAS OUT CLOSE
09EA    dw 2599 CAS OUT ABANDON
09EC    dw 25C6 CAS OUT CHAR
09EE    dw 2653 CAS OUT DIRECT
09F0    dw 2692 CAS CATALOG
09F2    dw 29AF CAS WRITE
09F4    dw 29A6 CAS READ
09F6    dw 29C1 CAS CHECK

09F8    dw 1FE9 SOUND RESET
09FA    dw 2114 SOUND QUEUE
09FC    dw 21CE SOUND CHECK
09FE    dw 21EB SOUND ARM EVENT
```

```
0A00    dw 21AC SOUND RELEASE
0A02    dw 2050 SOUND HOLD
0A04    dw 206B SOUND CONTINUE
0A06    dw 2495 SOUND AMPL ENVELOPE
0A08    dw 249A SOUND TONE ENVELOPE
0A0A    dw 24A6 SOUND A ADRESS
0A0C    dw 24AB SOUND T ADRESS

0A0E    dw 005C KL CHOKE OFF
0A10    dw 0326 KL ROM WALK
0A12    dw 0330 KL INIT BACK
0A14    dw 02A0 KL LOG EXT
0A16    dw 02B1 KL FIND COMMAND
0A18    dw 0163 KL NEW FRAME FLY
0A1A    dw 016A KL ADD FRAME FLY
0A1C    dw 0170 KL DEL FRAME FLY
0A1E    dw 0176 KL NEW FAST TICKER
0A20    dw 017D KL ADD FAST TICKER
0A22    dw 0183 KL DEL FAST TICKER
0A24    dw 01B3 KL ADD TICKER
0A26    dw 01C5 KL DEL TICKER
0A28    dw 01D2 KL INIT EVENT
0A2A    dw 01E2 KL EVENT
0A2C    dw 0227 KL SYNC RESET
0A2E    dw 0284 KL DELETE SYNCHRONOUS
0A30    dw 0255 KL NEXT SYNC
0A32    dw 0219 KL DO SYNC
0A34    dw 0276 KL DONE SYNC
0A36    dw 0294 KL EVENT DISABLE
0A38    dw 029A KL EVENT ENABLE
0A3A    dw 028D KL DISARM EVENT
0A3C    dw 0099 KL TIME PLEASE
0A3E    dw 00A3 KL TIME SET

0A40    dw 05ED MC BOOT PROGRAM
0A42    dw 061C MC START PROGRAM
0A44    dw 07B4 MC WAIT FLYBACK
0A46    dw 0776 MC SET MODE
0A48    dw 07C0 MC SCREEN OFFSET
0A4A    dw 0786 MC CLEAR INKS
0A4C    dw 078C MC SET INKS
0A4E    dw 07E0 MC RESET PRINTER
0A50    dw 081B MC PRINT CHAR
0A52    dw 0858 MC BUSY PRINTER
0A54    dw 0844 MC SEND PRINTER
0A56    dw 0863 MC SOUND REGISTER

0A58    dw 08BD JUMP RESTORE
```

```
0A5A    dw 1D3C KM SET STATE
0A5C    dw 1BFE KM TØM BUFFER
0A5E    dw 1460 TXT LFD. CURSOR FLAG TIL AKK
0A60    dw 15EC GRA NN
0A62    dw 19D5 GRA GEM PARAMETRE
0A64    dw 17B0 GRA GEM MASK PARAMETRE
0A66    dw 17AC GRA GEM MASK PARAMETRE
0A68    dw 1624 GRA KOORD. KONVERTERES
0A6A    dw 19D9 GRA FILL
0A6C    dw 0B45 SCR ÆNDRING AF SCREEN START
0A6E    dw 080C MC TEGNTILORDNING
0A70    dw 0397 KL SÆT RAM-KONFIGURATION

0A72 ********************************** BASIC Jump Adr.

0A72    dw 2C02 EDIT

0A74    dw 2F91 FLO KOPIER VARIAB. FRA (DE) TIL (HL)
0A76    dw 2F9F FLO INTEGER TIL FLOATING POINT
0A78    dw 2FC8 FLO 4-BYTE-VÆRDI TIL FLO
0A7A    dw 2FD9 FLO FLO TIL INT
0A7C    dw 3001 FLO FLO TIL INT
0A7E    dw 3014 FLO FIX
0A80    dw 3055 FLO INT
0A82    dw 305F FLO
0A84    dw 30C6 FLO MULTIPLICER TAL MED 10↑A
0A86    dw 34A2 FLO ADDITION
0A88    dw 3159 FLO RND
0A8A    dw 349E FLO SUBTRAKTION
0A8C    dw 3577 FLO MULTIPLIKATION
0A8E    dw 3604 FLO DIVISION
0A90    dw 3188 FLO HENT SIDSTE RND-VÆRDI
0A92    dw 36DF FLO SAMMENLIGN
0A94    dw 3731 FLO SKIFT FORTEGN
0A96    dw 3727 FLO SGN
0A98    dw 3345 FLO DEG/RAD
0A9A    dw 2F73 FLO PI
0A9C    dw 32AC FLO SQR
0A9E    dw 32AF FLO SÆT POTENS
0AA0    dw 31B6 FLO LOG
0AA2    dw 31B1 FLO LOG10
0AA4    dw 322F FLO EXP
0AA6    dw 3353 FLO SIN
0AA8    dw 3349 FLO COS
0AAA    dw 33C8 FLO TAN
0AAC    dw 33D8 FLO ATN
0AAE    dw 2FD1 FLO 4-BYTE-VÆRDI TIL FLO
0AB0    dw 3136 FLO RND INIT
0AB2    dw 3143 FLO SET RND SEED

0AB4 ********************** Move (hl+3) til ((hl+1)),cnt=(hl)
```

2.5.4. SCREEN PACK (SCR)

SCREEN PACK er underordnet TEXT- og GRAPHICS PACK. Den er egentlig ansvarlig for begge disse PACKs, og dermed også for den umiddelbare håndtering af skærmbilledet.

0ABF ********** SCR INITIALIZE

Fuldstændig initialisering af SCREEN PACK.

0ABF Default farver
0AC2 MC CLEAR INKS
0AC7 (High Byte Screen Start)
0ACA SCR RESET

0AD0 ********** SCR RESET

Reset af SCREEN PACK.

0AD1 SCR ACCESS
0AD4 Restore SCR Indirections
0AD7 Move (hl+3) til ((hl+1)),cnt=(hl)
0ADA Reset farver
0ADD db 09 9 Bytes
0ADE dw BDE5 måladresse
0AE0 SCR READ
0AE3 SCR WRITE
0AE6 SCR CLEAR

0AE9 ********** SCR SET MODE

Sæt skærm til en ny MODE.

0AFF SCR CLEAR

0B0C ********** SCR GET MODE

Hent den ønskede skærm-mode.

0B0C (curr. Screen Mode)

0B17 ********** SCR CLEAR

Slet skærmbilledet.

0B1D SCR SET OFFSET
0B25 hl=Basis Adresse
0B26 de=Basis Adresse+1

0B28 16k
0B2C Slet skærmbilledet.

0B31 (curr. Screen Mode)
0B34 MC SET MODE

0B37 ********** SCR SET OFFSET

Sæt startadresse for det første tegn relativt til basisadressen for Video-RAM.

0B37 (High Byte Screen Start)

0B3C ********** SCR SET BASE

Video-RAMs basisadresse.

0B3C (Position indenfor en linie).
0B42 MC SCREEN OFFSET

0B45 ********** SCR ÆNDRING AF SCREEN START.

0B47 (High Byte Screen Start)
0B51 (Position indenfor en linie).

0B56 ********** SCR GET LOCATION

Løbende Skærmstart? (Basis+Offset)

0B56 (Position indenfor en linie).
0B59 (High Byte Screen Start)

0B5D ********** SCR CHAR LIMITS

Hent maksimalt linie- og kolonnetal for skærmen (afhængig af mode).

0B5D SCR GET MODE

0B6A ********** SCR CHAR POSTION

Konverter fysiske koordinater til en skærmposition.

0B6B SCR GET MODE
0B93 (High Byte Screen Start)
0BA6 SCR CHAR POSITION

0BAF ********** SCR DOT POSITION

Hent skærmposition for en enkelt pixel.

0BED (High Byte Screen Start)

0BF6 SCR GET MODE

0C05 ********** SCR NEXT BYTE

Lægger skærmadressen for den næste byteposition i HL, hvis man inden tilgang har flyttet den oprindelige adresse til HL. Lige så overflødigt, det lyder, lige så praktisk er det at gøre det. På grund af den for grafik- mode opbyggede skærm, er det ikke så enkelt en sag at finde bytepositionen. Det hænger iøvrigt også sammen med mode. Pas derfor på, hvis den næste position ikke længere ligger indenfor skærmbilledet (den aktuelle mode), så er den fundne adresse ikke troværdig. Den ligger i en ubenyttet del af video-RAM.

0C11 ********** SCR PREV BYTE

Lægger skærmadressen for den forrige byteposition i HL, hvis man inden tilgang har sørget for at lægge den tidligere adresse i HL. Sammenlign med SCR NEXT BYTE.

0C1F ********** SCR NEXT LINE

Arbejder analogt med SCR NEXT BYTE, dog er skærmadressen regnet en linie frem. Også her gælder det, at adressen er ugyldig ved ændring af mode.

0C39 ********** SCR PREV LINE

Arbejder analog til SCR PREV BYTE, kun er skærmadressen regnet en linie bagud. Sammenlign med SCR NEXT og SCR PREV BYTE.

0C55 ********** SCR ACCESS

Sæt styretegn synligt/usynligt.

0C57 SCR PIXELS (FORCE MODE)
0C5E Low Byte XOR Mode
0C62 Low Byte AND Mode
0C66 Low Byte OR Mode
0C68 jp
0C6A (Write Indirection)

0C71 ********** SCR WRITE

0C71 Write Indirection

0C74 ********** SCR PIXELS (FORCE Mode)

Sæt et punkt på skærmen.

0C7A ********** XOR Mode

0C7F ********** AND Mode

0C85 ********** OR Mode

0C8A ********** SCR READ

0C8E ********** SCR INK ENCODE

Kodning af en INK, således at alle punkter sættes til farven.

0CA7 ********** SCR INK DECODE

Dekodning af INK.

0CC9 SCR GET MODE

0CD8 ********** Reset farver

0CD8 Default farver
0CDB Farvehukommelse 2. farver
0CE4 (Flag for løbende farve)

0CEA ********** SCR SET FLASHING

Sæt BLINK-tider til alle INKs og skærmkant.

0CEA (Flash Periods)

0CEE ********** SCR GET FLASHING

Find blinktider (INKs og kant).

0CEE (Flash Periods)

0CF2 ********** SCR SET INK

Tilordning af begge farver, der bruges til opbygning af en INK.

0CF5 Set Colour

0CF7 ********** SCR SET BORDER

Tilordning af begge farver, der bruges til opbygning af en skærmkant.

0CF8 ********** Set Colour

0CFA Hent farvematrix.
0CFF Hent farvematrix.
0D04 Hent INK-adresse

0D10 ****************************** Hent farvematrix.

0D1A ************************************ SCR GET INK

Hent begge farver, der bruges til opbygning af en INK.

0D1D Get Colour

0D1F ********************************** SCR GET BORDER

Hent begge farver, der bruges til opbygning af en skærmkant.

0D20 *** Get Colour

0D20 Hent Ink sdresse.
0D2C Farvematrix.

0D35 ********************************** Hent Ink adresse.

0D38 Farvehukommelse 1. farver.
0D42 Event Block: Set Inks
0D46 KL DEL FRAME FLY
0D49 Flash Inks
0D4C Set Inks on Frame Fly
0D52 KL NEW FRAME FLY
0D55 Event Block: Set Inks
0D58 KL DEL FRAME FLY
0D5B Hent parametre for løbende farvesæt.
0D5E MC CLEAR INKS

0D61 ******************************** Set Inks on Frame Fly

0D61 curr. Flash Period
0D65 Flash Inks
0D6B Hent parametre for løbende farvesæt.
0D6E MC SET INKS

0D73 ** Flash Inks

0D73 Hent parametre for løbende farvesæt.
0D76 (curr. Flash Period)
0D79 MC SET INKS
0D7C Flag for løbende farvesæt.

0D87 ************************** Hent parametre for løbende farvesæt.

0D87 Farvehukommelse 1. farver.
0D8A (Flag for løbende farvesæt.
0D8E (Flash Period 1. Colour)

0D92 Farvehukommelse for 2. farver.
0D95 (Flash Periods)

0D99 ********** Farvematrix.

0D99 14 04 15 1C 18 1D 0C 05
0DA1 0D 16 06 17 1E 00 1F 0E
0DA9 07 0F 12 02 13 1A 19 1B
0DB1 0A 03 0B 01 08 09 10 11

0DB9 ********** SCR FILL BOX

Udfyld aktuelt vindue med en farve. (Positioner er uafhængige af skærmadresser og mode).

0DBD ********** SCR FLOOD BOX

Udfyld aktuelt vindue med en farve. (Positioner er uafhængige af skærmadresser og mode).

0DC6 SCR NEXT BYTE
0DDE SCR NEXT LINE
0DE2 SCR FLOOD BOX

0DE5 ********** SCR CHAR INVERT

Byt farve på tegns for- og baggrund.

0DE8 SCR CHAR POSITION
0DF2 SCR NEXT BYTE

0DF8 ********** Adresser farvehukommelse.

0DF9 SCR NEXT LINE

0E00 ********** SCR HW ROLL

Forskyder skærmen (hardwaremæssigt) en linie nedefter, hvis B=0, og en linie opefter, hvis B <> 0. Værdien for farven, den nye (tomme) linie, skal være anført i akk.

0E0B MC WAIT FLYBACK
0E32 (High Byte Screen Start)
0E3A SCR FLOOD BOX
0E41 SCR SET OFFSET

0E44 ********** SCR SW ROLL

Forskyder et skærmområde softwaremæssigt. A og B skal have status som ved SCR HW ROLL. Desuden, skal H indeholde kolonnenummeret for den venstre kant i det

område, der skal forskydes, hvor 1 = øverste linie, D = højre kolonne og E = nederste linie i området.

Bemærk at kolonne og linie 0, svarer til øverste venstre hjørne i skærmen. Det er op til programmøren at sørge for, at de overflyttede parametre holdes indenfor et område i Video-RAM.

```
0E4F    SCR CHAR POSITION
0E5A    MC WAIT FLYBACK
0E64    SCR NEXT LINE
0E69    SCR NEXT LINE
0E76    SCR FLOOD BOX
0E8B    SCR CHAR POSITION
0E8F    SCR CHAR POSITION
0E93    MC WAIT FLYBACK
0E96    SCR PREV LINE
0E9B    SCR PREV LINE
0EE1    SCR NEXT BYTE
0EE5    SCR NEXT BYTE
```

0EF9 ************************************ SCR UNPACK

Forstørrelse af tegnmatrix for mode 0/1.

```
0EF9    SCR GET MODE
```

0F2A ************************************ SCR REPACK

Sæt tegnmatrix til originalform.

```
0F2B    SCR CHAR POSITION
0F2E    SCR GET MODE
0F3C    SCR NEXT LINE
0F48    SCR NEXT BYTE
0F53    SCR NEXT LINE
0F82    SCR NEXT BYTE
0F8C    SCR NEXT LINE
```

0F93 ********************************** SCR HORIZONTAL

Træk horisontal linie.

0F9B ********************************** SCR VERTICAL

Træk lodret linie.

```
0FA5    (GRA Pen)
0FA9    (GRA Pen)
0FAE    (GRA Pen)
```

```
0FB1    (GRA Pen)
0FB8    Load akk med &FF
0FF3    (GRA Paper)
0FFF    (GRA Pen)
100A    SCR NEXT BYTE
101C    (GRA Pen)
1027    (GRA Paper)
102C    SCR WIRTE
1030    SCR PREV LINE
1049    SCR DOT POSITION

1052 ****************************************** Default farver.

1052    04  04  0A  13  0C  0B  14  15
105A    0D  06  1E  1F  07  12  19  04
1062    17  04  04  0A  13  0C  0B  14
106A    15  0D  06  1E  1F  07  12  19
1072    0A  07
```

2.5.5. TEXT SCREEN (TXT)

Denne PACK er, som navnet siger, ansvarlig for forvaltningen af tekst.
Herunder hører tillige organisationen af vinduer.

Til håndtering af cursoren, skal der siges følgende:

De i cursor-rutiner nødvendige eller leverede koordinater, skal forstås som logiske angivelser. D.v.s., de angiver løbende (de aktuelle) vinduer. Koordinaterne 1,1 er her øverste venstre hjørne i et vindue. Ønsker man eksempelvis at placere cursoren udenfor vinduet med TXT SET CURSOR, sættes den automatisk på den næste mulige position indenfor vinduet, hvis cursoren er aktiv, eller der skal udskrives et tegn på positionen. Herved ændres også den løbende position (den, der findes med TXT GET CURSOR). Er cursoren ikke aktiv, accepteres den nye position, indtil der skal udskrives et tegn eller cursoren aktiveres.

```
1074 ************************************ TXT INITIALIZE
```

Fuldstændig initialisering af TEXT-PACK.

```
1074    TXT RESET
107E    TXT Sæt default parametre.
1081    Reset parametre (alle vinduer).

1084 ****************************************** TXT RESET
```

Reset af TEXT-PACK.

1084 Restore TXT Indirections
1087 Move (hl+3) til ((hl+1)), cnt=(hl)
108D db 0F 15 Bytes
108E dw BDCD måladresse.
1090 TXT DRAW/UNDRAW CURSOR
1093 TXT DRAW/UNDRAW CURSOR
1096 TXT WRITE CHAR
1099 TXT UNWRITE CHAR
109C TXT OUT ACTION

109F ********** Reset parametre (alle vinduer).

10A1 Start parametre for vindue 0.
10A4 Løbende Cursor Position (Row, Col)
10AF (Løbende skærmvidnue).
10B3 (Løbende skærmvindue).
10BB TXT STR SELECT
10BE TXT DRAW/UNDRAW CURSOR
10C1 TXT GET PAPER
10C4 (TXT løbende Paper).
10C7 TXT GET PEN
10CA (TXT løbende Pen).
10D6 TXT STR SELECT
10DA (TXT Løbende Pen).
10DD Sæt default parametre.

10E4 ********** TXT STR SELECT

Vælg tekstvindue.

10E6 Løbende skærmvindue.
10F1 Flyt adresse for vindues parametre til DE.
10F4 ldir cnt=15
10F8 Flyt adresse for vindues parametre til DE.
10FC ldir cnt=15

1103 ********** TXT SWAP STREAMS

Parametrene (farver, vindueskanter o.s.v.) for to vinduer ombyttes.

1103 (Løbende skærmvindue).
1108 TXT STR SELECT
110C (Løbende skærmvindue).
110F Flyt vindues parametre til DE.
1114 Flyt vindues parametre til DE.
1118 ldir cnt=15
111C TXT STR SELECT

111E ********** ldir cnt=15

1126 ********** Flyt adresse for vindues parametre til DE.

1135 Løbende cursor position (Row, Col)

1139 ********** Sæt default parametre.

113C (Løbende cursor flag).
1140 TXT SET PAPER
1144 TXT SET PEN
1148 TXT SET GRAPHIC
114B TXT SET BACK
1154 TXT WIN ENABLE
1157 TXT VDU ENABLE

115A ********** TXT SET COLUMN

Sæt cursors horisontale position.

115B Løbende vindue, venstre.
115F (Løbende cursor pos. (Row, Col)

1165 ********** TXT SET ROW

Cursors lodrette position.

1166 Løbende vindue, øverst.
116A (Løbende cursor position (Row,Col).

1170 ********** TXT SET CURSOR

Positioner cursor.

1170 Løbende vindue, øverst, venstre + HL
1173 TXT DRAW/UNDRAW CURSOR
1176 (Løbende cursorposition (Row,Col)).
1179 TXT DRAW/UNDRAW CURSOR

117C ********** TXT GET CURSOR

Aflæsning af den aktuelle cursorposition.

117C (Løbende cursorposition (Row,COL)).
117F Løbende vindue, øverst, venstre - HL
1182 (Løbende Roll Count)

1186 ********** Løbende vindue øverst, venstre + HL.

1186 (Løbende vindue, øverst).
118C (Løbende vindue, venstre).

1193 ********** Løbende vindue øverst, venstre - HL.

1193 (Løbende vindue, øverst).
119B (Løbende vindue, venstre).

11A4 ********** Move Cursor

11A4 TXT DRAW/UNDRAW CURSOR
11A7 (Løbende cursorposition (Row,Col)).
11AA ER HL indenfor vindues grænser?
11AD (Løbende cursorposition (Row,Col)).
11B2 Løbende Roll Count.
11BA TXT GET WINDOW
11BD (TXT Løbende Paper)
11C1 SCR SW ROLL
11C5 SCR HW ROLL

11CA ********** TXT VALIDATE

Er cursoren inden for tekstvinduets grænser?

11CA Løbende vindue, øverst, venstre + HL.
11CD Er HL inden for vinduets grænser?
11D1 Løbende vindue, øverst, venstre - HL.

11D6 ********** HL inden for vinduets grænser.

11D6 (Løbende vindue, højre).
11DD (Løbende vindue, venstre).
11E2 (Løbende vindue, venstre).
11E7 (Løbende vindue, højre).
11EF (Løbende vindue, øverst).
11F7 (Løbende vindue, nederst).

1208 ********** TXT WIN ENABLE

Bestem størrelse på løbende tekstvindue.

1208 SCR CHAR LIMITS
1229 (Løbende vindue, øverst).
122C (Løbende vindue, nederst).
123A (Vindues flag (0 = hele skærmen)).

1252 ********** TXT GET WINDOW

Hvilken størrelse har det aktuelle vindue?

1252 (Løbende vindue, øverst).
1255 (Løbende vindue, nederst).

1259 (Vinduets flag (0 = hele skærmen)).

125F ********** TXT DRAW/UNDRAW CURSOR

Sæt/slet cursor.

125F (Løbende cursor flag).

1265 ********** TXT PLACE/REMOVE CURSOR

Sæt cursor på skærm/fjern cursor fra skærm.

126B (TXT løbende Pen).
126F SCR CHAR INVERT

1276 ********** TXT CUR ON

Tillad cursor (Operativsystem).

1279 Cur Enable Cont'd

127E ********** TXT CUR OFF

Lås cursor (Operativsystem, højere prioritet end TXT CUR ENABLE og TXT CUR DISABLE.

1281 Cur Disable Cont'd

1286 ********** TXT CUR ENABLE

Tillad cursor (Brugerprogram).

1288 ********** Cur Enable Cont'd

1289 TXT DRAW/UNDRAW CURSOR
128E Løbende cursor flag.
1294 TXT DRAW/UNDRAW CURSOR

1297 ********** TXT CUR DISABLE

Lås cursor (Brugerprogram).

1299 ********** Cur Disable Cont'd

129A TXT DRAW/UNDRAW CURSOR
129F Løbende cursor flag.

12A6 ********** TXT SET PEN

Sæt forgrundsfarve.

12A6 TXT Løbende Pen.

12AB ********** TXT SET PAPER

Sæt baggrundsfarve.

12AB TXT Løbende Paper.
12AF TXT DRAW/UNDRAW CURSOR
12B3 SCR INK ENCODE
12B7 TXT DRAW/UNDRAW CURSOR

12BA ********** TXT GET PEN

Hvilken forgrundsfarve er sat?

12BA (TXT Løbende Pen).
12BD SCR INK DECODE

12C0 ********** TXT GET PAPER

Hvilken baggrundsfarve er sat?

12C0 (TXT løbende Paper).
12C3 SCR INK DECODE

12C6 ********** TXT INVERSE

Aktuel for- og baggrundsfarve ombyttes.

12C6 TXT DRAW/UNDRAW CURSOR
12C9 (TXT løbende Pen).
12CF (TXT løbende Pen).

12D4 ********** TXT GET MATRIX

Hent adresse for pixelmønster til et tegn.

12D6 TXT GET M TABLE

12F2 ********** TXT SET MATRIX

Sæt adresse for et bestemt tegns bitmønster (brugerdefineret).

12F3 TXT GET MATRIX

12FE ********** TXT SET M TABLE

Sæt startadresse og første tegn for et, af brugeren defineret, bitmønster.

130A TXT GET MATRIX
131E TXT GET M TABLE
1321 (1. tegn User Matrix).
1326 (Adr. User Matrix)

132B ********** TXT GET M TABLE

Startadresse og første tegn for et bitmønster defineret af brugeren.

132B (1. tegn User Matrix)
1331 (Adr. User Matrix)

1335 ********** TXT WR CHAR

Opbyg og vis tegn.

1336 (Løbende cursor flag)
133C move Cursor
1340 (Løbende cursor position (Row, Col))
1345 TXT WRITE CHAR
1348 TXT DRAW/UNDRAW CURSOR

134B ********** TXT WRITE CHAR

Skriv et tegn på skærmen.

134C TXT GET MATRIX
1353 SCR UNPACK
1358 SCR CHAR POSITION
1366 SCR NEXT BYTE
136F SCR NEXT LINE
1377 (Løbende Background Mode).

137B ********** TXT SET BACK

Transparentmode On/Off.

1384 (Løbende Background Mode)

1388 ********** TXT GET BACK

Hvilken transparentmode?

1388 (Løbende Background Mode)
1392 (TXT løbende Pen)
13A0 (TXT løbende Pen)
13A5 SCR PIXELS

13A8 ********** TXT SET GRAPHIC

Udlæsning af styretegn On/Off.

13A8 (GRA Char WR Mode (0=disable))

13AC ********** TXT RD CHAR

Læs et tegn fra skærmen.

13AF move Cursor
13B2 TXT UNWRITE CHAR
13B6 TXT DRAW/UNDRAW CURSOR

13BE ********** TXT UNWRITE CHAR

Læs et tegn fra skærmen.

13BE (TXT løbende Pen).
13C6 SCR REPACK
13DE SCR REPACK
13E4 TXT GET MATRIX

13FE ********** TXT OUTPUT

(Styre-)tegn udskrives eller udføres.

Flytter tegnet i akk til løbende skærmvindue, henholdsvis udfører det, hvis det drejer sig om et styretegn.

Bemærk at denne rutine gør brug af Indirection TXT OUT ACTION. Har man flyttet denne (pointer), så vil TXT OUTPUT også bruge den egendefinerede rutine, altså ikke ROM-rutinen.

1402 TXT OUT ACTION

140A ********** TXT OUT ACTION

Udskrivning af et tegn på skærmen eller udførelse af et styretegn.

140B (GRA Char WR Mode (0=disable)
1410 GRA WR CHAR
1413 Tegntæller Control Buffer
1418 Control Buffer fyldt op?
141A ja, så hop.
141C Control Buffer tømt?
141D nej, så hop
1420 Styretegn?
1422 nej, så TXT WR CHAR

```
1425    Tæller+1
142C    (Start Control Buffer)
1430    Hoptabel for styretegn.
1436    Antal, der kræves.
1439    Styreparameter nået?
143A    Nej, så hop
1446    Start Control Buffer
144A    call (de)
144E    (Tegntæller Control Buffer).

1452 ********************************** TXT VDU DISABLE
```

Disable tegnudlæsning.

```
1454    Cur Disable Cont'd

1459 ********************************** TXT VDU ENABLE
```

Der kan skrives på skærmen igen. (Se TXT VDU DISABLE).

```
145B    Cur Enable Cont'd

1460 ******************* LØBENDE CURSOR FLAG TIL AKK

1460    (løbende cursor flag)

1464 ****************** Kopier default styretegnshop.

1465    (Tegntæller Control Buffer)
1468    Default styretegnshop.
146B    Hoptabel for styretegn.
146E    Antal bytes
1471    Kopier

1474 ************************** Default styretegnshop.

1474    db 80
1475    dw 1513   00

1477    db 81
1478    dw 1335   01  TXT WR CHAR

147A    db 80
147B    dw 1297   02  TXT CUR DISABLE

147D    db 80
147E    dw 1286   03  TXT CUR ENABLE
```

```
1480    db 81
1481    dw 0AE9 04  SCR SET MODE

1483    db 81
1484    dw 1940  05  GRA WR CHAR

1486    db 00
1487    dw 1459  06  TXT VDU ENABLE

1489    db 80
148A    dw 14E1  07  BELL

148C    db 80
148D    dw 1519  08  CRSR Left

148F    dw 80
1490    dw 151E  09  CRSR Right

1492    db 80
1493    dw 1523  0A CRSR Down

1495    db 80
1496    dw 1528  0B CRSR Up

1498    db 80
1499    dw 154F  0C  TXT CLEAR WINDOW

149B    db 80
149C    dw 153F  0D CRSR til start på linien.

149E    db 81
149F    dw 12AB 0E  TXT SET PAPER

14A1    db 81
14A2    dw 12A6  0F  TXT SET PEN

14A4    db 80
14A5    dw 155E  10  Slet tegn på aktuel cursorposition.

14A7    db 80
14A8    dw 1599  11  Slet linie indtil cursorposition.

14AA    db 80
14AB    dw 158F  12  Slet linie fra cursorposition.

14AD    db 80
14AE    dw 1578  13  Slet vindue indtil cursorposition.
```

```
14B0    db 80
14B1    dw 1565  14  Slet vindue fra cursorposition.

14B3    db 80
14B4    dw 1452  15  TXT VDU DISABLE

14B6    db 81
14B7    dw 14EC  16  Transparentmode On/Off

14B9    db 81
14BA    dw 0C55  17  SCR ACCESS

14BC    db 80
14BD    dw 12C6  18  TXT INVERSE

14BF    db 89
14C0    dw 150D  19  SYMBOL-kommando.

14C2    db 84
14C3    dw 1501  1A  Definer vindue.

14C5    db 00
14C6    dw 14EB  1B  Ingen effekt.

14C8    db 83
14C9    dw 14F1  1C  INK-kommando.

14CB    db 82
14CC    dw 14FA  1D  BORDER-kommando.

14CE    db 80
14CF    dw 1539  1E  CRSR Home

14D1    db 82
14D2    dw 1547  1F  LOCATE-kommando.
```

14D4 ********** TXT GET CONTROLS

Hent adresse for styretegns hoptabel.

14D4 Hoptabel for styretegn.

14E1 ********** BELL

14E6 SOUND QUEUE

14EC ********** Transparentmode On/Off

14EE TXT SET BACK

14F1 ********** INK-kommando.

14F7 SCR SET INK

14FA ********** BORDER-kommando.

14FE SCR SET BORDER

1501 ********** Definer vindue.

150A TXT WIN ENABLE

150D ********** SYMBOL-kommando.

1510 TXT SET MATRIX

1513 Move Cursor
1516 TXT DRAW/UNDRAW CURSOR

1519 ********** CRSR Left

151E ********** CRSR Right

1523 ********** CRSR Down

1528 ********** CRSR Up

152C Move Cursor

1539 ********** CRSR Home

153F ********** CRSR til start på linien.

153F Move Cursor
1542 (Løbende vindue, venstre).

1547 ********** LOCATE-kommando.

154C TXT SET CURSOR

154F ********** TXT CLEAR WINDOW

Slet løbende tekst-vindue.

154F TXT DRAW/UNDRAW CURSOR
1552 (Løbende vindue, øverst)
1555 (Løbende Cursor Pos. (Row, Col)
1558 (Løbende vindue, nederst).

155E ********** Slet tegn på CRSR-Pos.

155E Move Cursor

1565 ********** Slet vindue fra CRSR-position.

1565 12 Slet linier fra CRSR-position.
1568 (Løbende vindue, øverst).
156B (Løbende vindue, nederst).
156F (Løbende Cursor Pos. (Row, Col))

1578 ********** Slet vindue indtil CRSR-position.

1578 11 Slet linier til CRSR-position.
157B (Løbende vindue, øverst).
157E (Løbende vindue, højre).
1582 (Løbende Cursor Pos. (Row, Col)
1589 (TXT løbende Paper)
158C SCR FILL BOX

158F ********** Slet linie fra CRSR-position.

158F Move Cursor
1593 (Løbende vindue, højre).

1599 ********** Slet linie til CRSR-position).

1599 Move Cursor
159E (Løbende vindue, venstre).
15A5 TXT DRAW/UNDRAW CURSOR

2.5.6. GRAPHICS SCREEN (GRA)

Denne PACK har udelukkende til opgave at styre grafik-vinduet.

Der skal bemærkes følgende til koordinatangivelserne, der forlanges af forskellige rutiner:

Koordinaterne fortolkes i 3 trin. Brugertrinet beror på positionen, givet ved kommandoen ORIGIN. Denne omregnes til en position relativ til skærmoprindelsen (nederst til venstre). Begge disse trin er uafhængige af den aktuelle MODE. Det sidste trin er punktets fysiske adresse, der igen er afhængig af den aktuelle skærmmode.

De 3 trin foranstilles et 4. trin, hvis et relativt koordinatsæt skal konverteres til en absolut position, der er relativ til ORIGIN.

15A8 ********** GRA INITIALIZE

Fuldstændig initialisering af GRAPHICS PACK.

15A8 GRA RESET
15AB Pen 1,Paper 0
15AF GRA SET PAPER
15B3 GRA SET PAPER
15B6 Sæt Origin til 0,0
15BB GRA SET ORIGIN
15C6 GRA WIN WIDTH
15CB GRA WIN HEIGHT
15CE GRA GET PAPER
15D2 GRA GET PEN

15D7 ********** GRA RESET

Reset af GRAPHICS PACK.

15DA Restore GRA Indirections
15DD Move (hl+3) til ((hl+1)), cnt=(hl)
15E0 db 09 9 Bytes
15E1 dw BDDC måladresse.
15E3 GRA PLOT
15E6 GRA TEST
15E9 GRA LINE

15EC ********** NN

15ED SCR ACCESS
15F1 GRA FILL

15FB ********** GRA MOVE RELATIVE

Bevægelse relativ til øjeblikkelig position.

15FB Add løbende koordinat + relativ koordinat.

15FE ********** GRA MOVE ABSOLUTE

Bevægelse til en absolut position.

15FE (Løbende X Koord.)
1602 (Løbende Y Koord.)

1606 ********** GRA ASK CURSOR

Hvor befinder grafik-cursoren sig?

1606 (Løbende X Koord.)
160A (Løbende Y Koord.)

160E ********** GRA SET ORIGIN

Sæt udgangspunkt for bruger-koordinater.

160E (X Origin)
1612 (Y Origin)
161A GRA MOVE ABSOLUTE

161C ********** GRA GET ORIGIN

Hent udgangspunkt for bruger-koordinater.

161C (X Origin)
1620 (Y Origin)

1624 ********** Hent den fysiske startposition.

1624 GRA ASK CURSOR

1627 ********** Hent fysisk målposition + placering af cursor.

1627 GRA MOVE ABSOLUTE

162A ********** Konverter grafik-koordinater.

162B SCR GET MODE
1640 (X Origin)
1655 (Y Origin)

165D ********** Add løbende koordinater + relative koordinater.

165E (Løbende X Koord.)
1664 (Løbende Y Koord.)
166A (X Koord. GRA vindue, venstre).
1673 (X Koord. GRA vindue, højre).
1680 (Y Koord. GRA vindue, øverst).
1689 (Y Koord. GRA vindue, nederst).
1694 Hent fysisk målposition + cursorplacering.

16A5 ********** GRA WIN WIDTH

Sæt venstre og højre begrænsning af grafikvinduet.

16BE SCR Get Mode
16C9 (X Koord. GRA vindue, venstre).

16CD (X Koord. GRA vindue, højre).

16EA ********** GRA WIN HEIGHT

Sæt øverste og nederste begrænsning på grafikvinduet.

16FB (Y Koord. GRA vindue, øverst).
16FF (Y Koord. GRA vindue, nederst).

1717 ********** GRA GET W WIDTH

Hent venstre og højre begrænsning på grafikvinduet.

1717 (X Koord. GRA vindue, venstre).
171B (X Koord. GRA vindue, højre).
171E SCR GET MODE

172D ********** GRA GET W HEIGHT

Hent øverste og nederste begrænsning på grafikvinduet.

172D (Y Koord. GRA vindue, øverst).
1731 (Y Koord. GRA vindue, nederst).

1736 ********** GRA CLEAR WINDOW

Slet grafikvindue.

1736 GRA GET W WIDTH
1746 (Y Koord. GRA vindue, nederst).
174A (Y Koord. GRA vindue, øverst).
1753 (X Koord. GRA vindue, venstre)
1759 SCR DOT POSITION
175D (GRA Paper)
1761 SCR FLOOD BOX

1767 ********** GRA SET PEN

Sæt skrivefarve.

1767 SCR INK ENCODE
176A (GRA Pen)

176E ********** GRA SET PAPER

Sæt baggrundsfarve.

176E SCR INK ENCODE
1771 (GRA Paper)

1775 ********** GRA GET PEN

Hvilken skriftfarve?

1775 (GRA Pen)

177A ********** GRA GET PAPER

Hvilken baggrundsfarve?

177A (GRA Paper)
177D (SCR INK DECODE)

1780 ********** GRA PLOT RELATIVE

Sæt grafikpunkt relativ til aktuel cursorposition.

1780 Add løbende koordinat + relativ koordinat.

1783 ********** GRA PLOT ABSOLUTE

Sæt grafikpunkt (absolut).

1783 GRA PLOT

1786 ********** GRA PLOT

Sæt et punkt på skærmen.

178A SCR DOR POSITION
178D (GRA Pen)
1791 SCR WRITE

1794 ********** GRA TEST RELATIVE

Er det satte punkt relativt til løbende cursor?

1794 Add. Løbende koordinat + relativ koordinat.

1797 ********** GRA TEST ABSOLUTE

Er det satte punkt absolut?

1797 GRA TEST

179A ********** GRA TEST

Find INK for den øjeblikkelige grafikposition.

179D GRA GET PAPER
17A0 SCR DOT POSITION
17A3 SCR READ

17A6 ********** GRA LINE RELATIVE

Sæt en linie fra aktuelle position til relative position.

17A6 Add. løbende koordinat + relativ koordinat.

17A9 ********** GRA LINE ABSOLUTE

Sæt en linie fra aktuelle position til absolutte position.

17A9 GRA LINE

17AC ********** GEM GRA MASK PARAMETRE.

Gem parameter fra BASIC-kommandoen MASK.

17B0 ********** GEM GRA MASK PARAMETRE

Gem parameter fra BASIC-kommandoen MASK.

17B4 ********** GRA LINE

Tegn en linie.

17B9 Hent fysisk målposition og sæt cursor.
17BD (Regnebuffer X Koord.)
17CC (Regnebuffer Y Koord.)
188C Hent fysisk startposition.
188F (Regnebuffer X Koord.)
1893 (Regnebuffer Y Koord.)
18A2 (Regnebuffer Y Koord.)
18AD (Regnebuffer Y Koord.)
18B2 (Regnebuffer Y Koord.)
18B9 (Y Koord. GRA vindue, øverst)
18C3 (Y Koord. GRA vindue, nederst)
18C8 (Regnebuffer X Koord.)
18DA (Regnebuffer X Koord.)
18E6 (Regnebuffer X Koord.)
18EF (Regnebuffer X Koord.)
18FA (Regnebuffer X Koord.)
18FF (Regnebuffer X Koord.)
1906 (X Koord. GRA vindue, højre)
1910 (X Koord. GRA vindue, venstre)
1915 (Regnebuffer Y Koord.)
1928 (Regnebuffer Y Koord.)
1934 (Regnebuffer Y Koord.)

1940 ********** GRA WR CHAR

Skriv et tegn på den aktuelle grafik-cursor position.

1942 TXT GET MATRIX
1948 Hent fysiske startposition
1962 SCR DOR POSITION
1973 SCR NEXT BYTE
197B SCR NEXT LINE
1985 GRA ASK CURSOR
1989 SCR GET MODE
1998 GRA MOVE ABSOLUTE
19AC SCR DOT POSITION
19C4 (GRA Pen)
19CE (GRA Paper)
19D2 SCR WRITE

19D5 ********** GEM GRA PARAMETRE

19D9 ********** GRA FILL

19D9 (Regnebuffer X Koord.)
19DF (Regnebuffer Y Koord.)
19E3 SCR INK ENCODE
19E9 Hent fysisk startposition
1A19 (Regnebuffer X Koord.)
1A25 (Regnebuffer Y Koord.)
1A2C (Regnebuffer Y Koord.)
1A44 (Regnebuffer X Koord.)
1A9F (Regnebuffer Y Koord.)
1AA9 (Regnebuffer Y Koord.)
1AC1 (Regnebuffer X Koord.)
1AE8 (GRA Y Koord. GRA vindue, øverst)
1B10 SCR PREV LINE
1B18 (Y Koord. GRA vindue, nederst)
1B25 SCR NEXT LINE
1B35 (GRA Pen)
1B45 SCR DOT POSITION
1B51 SCR DOT POSITION
1B56 SCR DOT POSITION

2.5.7. KEYBOARD MANAGER (KM)

Denne PACK varetager overvågning af tastaturet og konverteringen til brugbare tegnkoder. Ved udførelse af denne opgave, d.v.s. den cykliske scanning af tastaturet, anvendes EVENT-proceduren.

1B5C ********** KM INITIALIZE

Fuldstændig initialisering af tastaturforvaltningen. Tilstanden fra før kald af KM INITIALIZE slettes.

1B5F KM SET DELAY
1B68 (Shift Lock State)
1B80 Key Translation Table
1B8A Key State Map
1B8D under Scan aktiverede taster.

1B98 ********** KM RESET

Tastaturforvaltningen bringes til udgangsstatus. Den indirekte hoptabel og tastaturbufferen neutraliseres.

1BA4 Exp Buffer Cont'd
1BA7 Restore KM Indirection
1BAA Move (hl+3) til ((hl+1)), cnt=(hl)
1BB0 KM DISARM BREAK
1BB3 db 03 3 bytes
1BB4 dw BDEE måladresse
1BB6 Test Break

1BBF ********** KM WAIT CHAR

Henter et tegn fra input-bufferen, henholdsvis expansion-string eller Put-Back- buffer. Hvis der ikke er et tegn at hente, returnerer rutinen ikke, men venter.

Når et tegn dukker op, indeholdes det i akk.

1BBF KM READ CHAR
1BC2 KM WAIT CHAR

1BC5 ********** KM READ CHAR

Henter ligeledes et tegn (sammenlign med KM WAIT CHAR), hvis der er et tegn. Hvis carry efter returnering er sat, var der ikke noget tegn.

1BC6 Put Back Buffer
1BC9 Hent tegn
1BCA Slet buffer
1BCC Var der et tegn?
1BCD Hvis ja, hop
1BCF (Exp. String Pointer)
1BD2 Highbyte til akk.
1BD3 Exp. String til stede?
1BD4 Hvis ja, hop
1BD6 KM READ KEY

1BD9 Hop, hvis intet tegn
1BDB Er tegnet < 128?
1BDD Hvis < 128, hop
1BE8 KM GET EXPAND
1BF0 (Exp. String Pointer)

1BF8 Akk=&FF

1BFA ********** KM CHAR RETURN

Returner et tegn til tastaturbufferen for næste tilgang (KM READ CHAR eller KM WAIT CHAR).

1BFA (Put Back Buffer)

1BFE KM READ CHAR

1C04 ********** KM EXP BUFFER

Tilskriv hukommelse for udvidelsesstreng (Adresse, længde).
Initialiser buffer.

1C04 Exp Buffer Cont'd

1C0A ********** Exp Buffer Cont'd

1C13 (Pointer End Exp Buffer)
1C17 (Pointer Start Exp Buffer)
1C1A ASCII
1C1D 0
1C1F til
1C20 9
1C21 til
1C22 Expansion
1C23 Buffer
1C25 Restore
1C26 Default Exp String
1C35 (Pointer for fri Exp Buffer)

1C3C ********** Default Exp String

```
1C3C   01  2E 01  0D 05  52  55  4E  .....RUN
1C44   22  0D                       ".
```

1C46 ********** KM SET EXPAND

Opbyg udvidelsesstreng.

1C47 Adresse for Exp String til DE
1C4A Hop, hvis token er invalid
1C4E Ryd Exp Buffer

1C6A ********** Ryd Exp Buffer

1C79 Plads til ny Exp String?
1C85 (Pointer for fri Exp Buffer)
1C8A (Pointer End Exp Buffer)
1C93 Platz til ny Exp String?
1C96 (Pointer for fri Exp Buffer)
1CA1 (Pointer for fri Exp Buffer)

1CA7 ********** Plads til ny Exp String?

1CA7 (Pointer for fri Exp Buffer).

1CB3 ********** KM GET EXPAND

Hent tegn fra udvidelsesstreng. Med start ved 0 er tegnene i tabellen nummereret fortløbende.

1CB3 Adresse på Exp String til DE

1CC3 ********** Adresse på Exp String til DE.

1CC3 Er token inden for
1CC5 validt område?
1CC7 Retur, hvis ikke.
1CC9 (Pointer Start Exp Buffer)
1CD0 Udvid HL med længden
1CD1 af Expansion
1CD2 String

1CDB ********** KM WAIT KEY

Afventer næste tryk på tastaturet, hvis der ikke umiddelbart er et tegn til rådighed. Tester kun på input-buffer, ikke Expansion String og Put Back Buffer. (Sammenlign med KM WAIT CHAR).

1CDB KM READ KEY
1CDE KM WAIT KEY

1CE1 ********** KM READ KEY

Henter tastenummer, hvis en tast trykkes. Venter ikke, hvis der ikke umiddelbart er et tegn til rådighed. Der tages ikke hensyn til Expansion-String og Put Back Buffer.

1CFB Caps Lock State
1D12 Shift Lock State
1D17 caps lock?
1D1A Hvis ikke, hop.
1D1D toggle caps lock
1D27 KM GET CONTROLS
1D2B (Shift Lock State)
1D32 KM GET SHIFT
1D35 KM GET TRANSLATE

1D38 ********** KM GET STATE

Undersøg om CAPS-LOCK- og SHIFT-LOCK-tasterne er aktiveret.

1D38 (Shift Lock State)

1D3C ********** Set State

1D3C (Shift Lock State)

1D40 ********** KM UPDATE KEY STATE MAP

1D40 Mulitihit Kontr. zu B63F
1D43 Under Scan aktiverede taster.
1D46 Scan Keyboard
1D4C Isoler SHIFT/CTRL
1D4F Key 16...23
1D54 Mulitihit Kontr. zu B63F
1D57 Key State Map
1D74 Test Break
1D86 Key State Map
1D8B (Adresse på repeat tabellen)
1D9E (KM Delay)

1DB8 ********** KM TEST BREAK

1DC1 KM BREAK EVENT
1DCE KM BREAK EVENT

1DE5 ********** KM GET JOYSTICK

Joysticks tilstand på tidspunktet for aftastning, findes ved hjælp af Key State Map.

1DE5 (Joystick 1)
1DEB (Joystick 0)

1DF2 ********** KM GET DELAY

Parameter for repetitionsfrekvens og hastighed.

1DF2 (KM Delay)

1DF6 ********** KM SET DELAY

Sæt tastrepetitionsfrekvens og hastighed.

1DF6 (Km Delay)

1DFA ********** KM ARM BREAK

Enable Break-tast.

1DFA KM DISARM BREAK
1DFD Break Event Block
1E02 KL INIT EVENT

1E0B ********** KM DISARM BREAK

Disable Break-tast.

1E13 KL DEL SYNCHRONOUS

1E19 ********** KM BREAK EVENT

Udfør rutiner ved betjening af Break-tast.

1E24 KL EVENT

1E2F ********** KM GET REPEAT

Test om tasten har sat repetitionsfunktion.

1E2F (Adresse på repeat tabellen)
1E32 Z Sæt tilhørende Key Bit.

1E34 ********** KM SET REPEAT

Via en tilføjelse i repetitionstabellen bestemmes det, om en tast har repetitions-funktion. Tastnummeret befinder sig herved i akk. Skal tasten repeteres, skal B indeholde &FF. Indeholder B derimod &00, annulleres repetitionsfunktionen for den pågældende tast.

1E34 Key > 80?
1E36 ja, så invalid.
1E37 (Adresse på repeat tabellen)
1E3A Hent Key# tilsvarende bit.

1E45 ********** KM TEST KEY

Med Key State Maps's tilstand testes der for betjening af tast eller joystick.

```
1E46    (Key 16...23)
1E49    Isoler SHIFT/CTRL
1E4D    Key State Map
1E50    Hent Key# svarende til bit.
1E53    Mask Key Bit

1E55 ***************************** Hent Key# svarende til bit.

1E57    Key#
1E59    /8
1E5F    Adresser Key Map
1E62    Masker bit
1E65    Load
1E67    bit svarende
1E68    til
1E69    tasten

1E6D ***************************************** Bit Masken

1E6D    01  02  04  08  10  20  40  80

1EC4 ******************************* KM GET TRANSLATE
```

Hent tilføjelse fra første del af tastaturtabellen (Key State Map).

```
1EC4    (Adresse Key Transl. Table)
1EC7    Get Key Table

1EC9 ************************************ KM GET SHIFT
```

Hent tilføjelse i anden del af tastaturtabellen.

```
1EC9    (Adresse Key SHIFT Table)
1ECC    Get Key Table

1ECE ********************************** KM GET CONTROL
```

Hent tilføjelse i tredie del af tastaturtabellen.

```
1ECE    (Adresse Key CTRL Table)

1ED1 ************************************* Get Key Table

1ED8 ******************************* KM SET TRANSLATE
```

Tag indhold i første del af tastaturtabellen.

```
1ED8    (Adresse Key Transl. Table)
1EDB    Set Key Table

1EDD ************************************ KM SET SHIFT

Tag indhold i anden del af tastaturtabellen.

1EDD    (Adresse Key SHIFT Table)
1EE0    Set Key Table

1EE2 ********************************** KM SET CONTROL

Tag indhold i tredie del af tastaturtabellen.

1EE2    (Adresse Key CTRL Table)

1EE5 ************************************** Set Key Table

1EEF ********************************** Key Translation Table

1EEF    F0 F3 F1 89 86 83 8B 8A
1EF7    F2 E0 87 88 85 81 82 80
1EFF    10 5B 0D 5D 84 FF 5C FF
1F07    5E 2D 40 70 3B 3A 2F 2E
1F0F    30 39 6F 69 6C 6B 6D 2C
1F17    38 37 75 79 68 6A 6E 20
1F1F    36 35 72 74 67 66 62 76
1F27    34 33 65 77 73 64 63 78
1F2F    31 32 FC 71 09 61 FD 7A
1F37    0B 0A 08 09 58 5A FF 7F

1F3F ************************************ Key SHIFT Table

1F3F    F4 F7 F5 89 86 83 8B 8A
1F47    F6 E0 87 88 85 81 82 80
1F4F    10 7B 0D 7D 84 FF 60 FF
1F57    A3 3D 7C 50 2B 2A 3F 3E
1F5F    5F 29 4F 49 4C 4B 4D 3C
1F67    28 27 55 59 48 4A 4E 20
1F6F    26 25 52 54 47 46 42 56
1F77    24 23 45 57 53 44 43 58
1F7F    21 22 FC 51 09 41 FD 5A
1F87    0B 0A 08 09 58 5A FF 7F

1F8F ************************************ Key CTRL Table

1F8F    F8 FB F9 89 86 83 8C 8A
1F97    FA E0 87 88 85 81 82 80
1F9F    10 1B 0D 1D 84 FF 1C FF
```

```
1FA7   1E FF 00 10 FF FF FF FF
1FAF   1F FF 0F 09 0C 0B 0D FF
1FB7   FF FF 15 19 08 0A 0E FF
1FBF   FF FF 12 14 07 06 02 16
1FC7   FF FF 05 17 13 04 03 18
1FCF   FF 7E FC 11 E1 01 FE 1A
1FD7   FF FF FF FF FF FF FF 7F
1FDF   07 03 4B FF FF FF FF FF
1FE7   AB 8F
```

2.5.8. SOUND MANAGER (SOUND)

Der kan ikke siges meget om denne PACK, selvom den er omfangsrig. Den egentlige tonefrembringelse optager ikke meget plads. Det er opbudet af de talrige venteløkker, der fylder. Hertil tæller også realiseringen af TONE ENVELOPE, som den programmerbare Sound Generator (PSG) ikke kan styre alene.

Hvis man ønsker, at CPC skal kunne frembringe naturtro fløjtelyde, så anbefales det, at man programmerer PSG direkte, da rutinerne i SOUND MANAGER i højere grad er baseret på udnyttelse fra BASIC-kommandoer. Selvom CPC kan lyde af meget i BASIC, så vil man få problemer med frembringelse af fyldigt slagtøj. Her er man nødt til at ty til maskinkodeprogrammering. Først her gives der mulighed for opbygning af komplekse klangstrukturer i hurtig udførelse.

1FE9 ************************************ SOUND RESET

Reset af hele SOUND MANAGER. Sletning af alle ventekøer.

```
1FF3   Sound Event
1FF8   KL INIT EVENT
2000   SOUND Params Kanal A
```

2050 ************************************** SOUND HOLD

Standser al lyd, men kan fortsættes med SOUND CONTINUE.

```
2050   Løbende SOUND aktivitet.
2058   Kanaler aktive?
2059   Hvis ikke, returhop
205C   Sæt volumen
205E   for alle kanaler
2060   til 0
2063   MC SOUND REGISTER
```

206B ********************************** SOUND CONTINUE

Standsede toner (SOUND HOLD) fortsættes.

206B (gammel SOUND
206E Akt. (efter HOLD))
206F Kanal aktiv?
2070 Hvis ikke, returhop
2076 Sæt tidligere
2079 lydstyrke
207A for alle
207D kanaler igen

208B ********** Sound Event

209D Kanal aktiv?
209F Nej, så næste

20D7 ********** Scan Sound Queues

20D7 Løbende SOUND aktivitet
2111 KL EVENT

2114 ********** SOUND QUEUE

Flyt tone til kø.

2114 SOUND CONTINUE

21AC ********** SOUND RELEASE

Frigiv toner.

21AD SOUND CONTINUE

21CE ********** SOUND CHECK

Er der plads i ventekøen?

21EB ********** SOUND ARM EVENT

2206 KL EVENT
2258 Løbende SOUND aktivitet
227D KL EVENT
2296 SOUND Params Kanal A
229E SOUND Params Kanal B
22A6 SOUND Params Kanal C
22B8 SOUND Params Kanal C
22C0 SOUND Params Kanal B
22F3 Load støjgenerator.
22F5 MC SOUND REGISTER
2303 Volumen for indhyldningskurver
2342 Sæt volumen

237D for indhyldningskurve.
237F MC SOUND REGISTER
2383 Indhyldningskurve-længde Lo
2385 MC SOUND REGISTER
2389 Indhyldningskurve-længde Hi
238B MC SOUND REGISTER
2390 Sæt volumen.

23DB ********** Sæt volumen.

23E2 Volumen.
23E4 MC SOUND REGISTER
23EF Løbende SOUND-aktivitet
2403 Kanal-styreregister.
2405 MC SOUND REGISTER
240C SOUND T ADRESSE
2486 Tonehøjde Lo
2489 MC SOUND REGISTER
248F Tonehøjde Hi
2492 MC SOUND REGISTER

2495 ********** SOUND AMPL ENVELOPE

Opbyg volumen for indhyldningskurve (15 forskellige amplituder).

2495 Volumen for indhyldningskurve
2498 Kopier indhyldningskurve.

249A ********** SOUND TONE ENVELOPE

Opbyg toneindhyldningskurve (15 forskellige tone-indhyldningskurver).

249A Toneindhyldningskurver.

249D ********** Kopier indhyldningskurve.

249E Hent adresse for indhyldningskurve.

24A6 ********** SOUND A ADRESS

Hent adressen for en volumenindhyldningskurve.

24A6 Volumen for indhyldningskurve
24A9 Hent adresse for kurve.

24AB ********** SOUND T ADRESS

Hent adressen på en tone-indhyldningskurve.

24AB Tone-indhyldningskurver.

24AE ********** Hent adresse på en kurve.

2.5.9. CASSETTE MANAGER (CAS)

Selvom læserens computer har et indbygget diskettedrev, så vil vi ikke forsømme at præsentere CASSETTE MANAGER, idet den indeholder nogle rutiner, man bør kende.

24BC ********** CAS INITIALIZE

Fuldstændig initialisering af CASSETTE PACK.

24BC CAS IN ABANDON
24C3 CAS NOISY

24CE ********** CAS SET SPEED

Sæt skrivehastighed.

24D9 (Cass. Speed)

24E1 ********** CAS NOISY

Kassettemeldinger On/Off. Ved spærring af meddelelser fra kassettestationen udelukkes fejlmeddelelser.

24E1 (Cass. Message Flag)

24E5 ********** CAS IN OPEN

Åbner en input-fil. B skal indeholde filnavnets længde, HL filnavnets start adresse og DE skal indeholde startadressen for et 2K stort RAM-område, der skal tjene som input-buffer.

Efter returnering indeholder HL fileheaderens startadresse. A, BC og DE indeholder de fra headeren indeholdte værdier, hvis startadresse som ligger i HL.

Flagene carry og zero giver oplysninger om procedurens resultat:

Carry = 1 , zero = 0	Viser at alt er foregået korrekt.
Carry = 0 , zero = 0	Viser at en anden fil allerede var åben.
Carry = 0 , zero = 1	Viser at ESC-tasten blev aktiveret.

24E5 Input Buffer Status
24E9 Cass. Open
24ED Læs File Header

24FE ********** CAS OUT OPEN

En outputfil åbnes. Overførselsparametrene og betydningen af flagene svarer til CASS IN OPEN, med den undtagelse at DE nu skal indeholde output-bufferens startadresse.

24FE Output Buffer Status

2502 ********** Cass. Open

2550 ********** CAS IN CLOSE

Korrekt lukning af input-filen.

2550 (Input Buffer Status)

2557 ********** CAS IN ABANDON

Øjeblikkelig standsning af læsning og lukning af input-fil (i tilfælde af en fejl).

2557 Input Buffer Status

257F ********** CAS OUT CLOSE

Korrekt lukning af output-filen.

257F (Output Buffer Status)

2599 ********** CAS OUT ABANDON

Øjeblikkelig lukning af output-fil og markering af device som lukket. Endnu ikke skrevne data slettes i bufferen.

2599 Output Buffer Status

25A0 ********** CAS IN CHAR

Henter et tegn fra input-bufferen og flytter det til akk. Var det sidste tegn i bufferen, læses der automatisk en ny blok fra kassetten til bufferen.

Carry = 0, zero = 0

betyder, at filens slutning er nået (EOF) eller at filen ikke var åben. Alle andre kombinationer svarer til CAS IN OPEN.

25A5 Check Input Buffer Status
25B0 Læs File Header
25BC (Pointer Input Buffer)
25BF ld a,(hl)
25C1 (Pointer Input Buffer)

25C6 ********** CAS OUT CHAR

Skriver tegnet i akk i output-buffer. Er bufferen fyldt op, skrives den automatisk til tape.

Flagenes betydning svarer til CAS IN CHAR og CAS IN OPEN.

25CA Output Buffer Status
25CF Check Buffer Status
25EA (Pointer Output Buffer)
25EF (Pointer Output Buffer)

25F6 ********** Check Input Buffer Status

25F6 Input Buffer Status

25F9 ********** Check Buffer Status

2603 ********** CAS TEST EOF

Test for filens afslutning.

2603 CAS IN CHAR

2607 ********** CAS RETURN

Det sidst læste tegn flyttes tilbage til bufferen.

260F (Pointer Input Buffer)
2613 (Pointer Input Buffer)

2618 ********** CAS IN DIRECT

Hele input-filen flyttes til hukommelsen; ingen tegnvis læsning.

261B (Check Input Buffer Status)
2631 Læs File Header
263C (Adr. Start Input Buffer)
2647 KL LDIR CONT'D
2650 KL LDDR CONT'D

2653 ********** CAS OUT DIRECT

Skriv defineret hukommelsesområde på tape (ikke via buffer).

2656 Output Buffer Status
265B Check Buffer Status
266E (Adr. Start Output Buffer)
2685 (Adr. Start Output Buffer)

2692 ********** CAS CATALOG

Udskrivning af katalog fra kassette til skærm.

2692 Input Buffer Status
269C (Adr. Start Input Buffer)
26A1 CAS NOISY
26A9 CAS IN ABANDON

26AC ********** Læsning af File Header

26C3 CAS READ
26E0 (Input Buffer Status)
26EF (Adr. Start Input Buffer)
26F2 (Pointer Input Buffer)
26F7 CAS READ
271B Input Buffer Status
2743 (File Header Input)
274E File Header Input
2760 File Header Input
277B CAS OUT CLOSE
2781 CAS MOTOR STOP
2790 File Header Output
279E (Adr. Start Output Buffer)
27A1 (Pointer Output Buffer)
27A8 File Header Output
27B0 CAS WRITE
27BC CAS WRITE
27D9 Output Buffer Status
27F5 CAS START MOTOR
2807 (Cass. Message Flag)
2846 TXT WR CHAR
2871 CAS Udskriv meddelelse (1 tegn).
2886 CAS Udskriv meddelelse (# i b).
288C CAS Udskriv meddelelse (1 tegn).

2891 ********** Udskriv Cass. meddelelse (# i b).

2891 TXT GET CURSOR
289D Kassette-meddelelser
28C9 Udskriv Cass. meddelelse (1 tegn).
28D0 Udskriv Cass. meddelelse (1 tegn).

28D2 (Cass. Message Flag)
28D8 Udskriv Cass. meddelelse (# i b)
28DB KM READ CHAR
28DE TXT CUR ON
28E1 KM WAIT KEY
28E4 TXT CUR OFF

28F0 ********** Udskriv Cass. meddelelse (1 tegn).

28F0 TXT OUTPUT
28F7 TXT SET COLUMN
28FE TXT GET WINDOW
2902 TXT GET CURSOR
2924 Udskriv Cass. meddelelse (1 tegn).
292F (Input Buffer Status)

2935 ********** Kassette-meddelelser.

2935 *Press*
293B *PLAY*
293F *then*
2943 *any*
2946 *key*
294B *error*
2955 *REC*
2958 *and*
295D *Read*
2963 *Write*
296A *Rewind*
2970 *tape*
2975 *Found*
297D *Loading*
2985 *Saving*
298D *ok*
2990 *block*
2996 *Unnamend*
299D *file*

29A6 ********** CAS READ

Læs en blok fra kassette. Denne rutine kaldes fra overordnede rutiner.

29A6 Motor On & åbent keyboard.

29AF ********** CAS WRITE

Skriv en blok på kassette. Kaldes som CAS READ fra overordnede rutiner.

29AF Motor On & åbent keyboard.

29C1 ********** CAS CHECK

Verify. Sammenlign blok på kassette med indhold i hukommelsen.

29C1 Motor On & åbent keyboard.
29D2 Port A=Out
29D7 Motor On
29DE CAS RESTORE MOTOR

29E3 ********** Motor On & åben keyboard.

29EA SOUND RESET
29F0 CAS START MOTOR
29F4 Sound I/O Port select
29F9 Strobe On
29FE Strobe Off
2A02 Port A=In
2A07 Keyb. Y9 (ESC) åbnes
2A0A & Sound I/O til Port A
2A3C RAM LAM (IX)
2A67 RAM LAM (IX)
2A95 Cass. Input RD DATA & Test ESC
2A9D Cass. Input RD DATA & Test ESC
2AB2 Cass. Input RD DATA & Test ESC

2B3D ********** Cass. Input RD DATA & Test ESC

2B3D Port A
2B3F Keyb. X
2B41 ESC?
2B43 Hvis ja, returhop
2B4D Port B
2B55 Input RD DATA
2B8E WR DATA Off
2B90 Cass. Output WR DATA
2B9F WR DATA On
2BA1 Cass. Output WR DATA

2BA7 ********** Cass. Output WR DATA

2BB1 Port Control
2BB3 WR DATA

2BBB ********** CAS START MOTOR

Kassettemotor On

2BBD CAS RESTORE MOTOR

2BBF ********** CAS STOP MOTOR

Stands kassettemotor.

2BC1 ********** CAS RESTORE MOTOR

Genopretter oprindelig motorstatus. Efter opstart af motor afventes konstant omdrejningstal.

2BC2 Port C
2BCE Motor On/Off
2BE9 KM TEST KEY

2.5.10. SCREEN EDITOR (EDIT)

SCREEN EDITOR bliver, i modsætning til de indtil nu behandlede bestanddele i L ROM, overhovedet ikke benyttet af operativsystemet. Strengt taget, er SCREEN EDITOR slet ikke en PACK i normal forstand. Den kan nok mere betragtes som værende i familie med aritmetik-rutinerne, der ligeledes udelukkende er til gængelige fra BASIC.

Der er ikke megen fornuft i at anvende rutinerne i SCREEN EDITOR enkeltvis. I bedste fald samlet. HL skal forsynes med startadressen for den tekst, der skal editeres. Teksten må maksimalt være af længden 255 tegn, hvilket også svarer til den maksimale længde for en BASIC-linie.

2C02 ********** EDIT

2C12 Udfør EDIT hop
2C1A Udfør EDIT hop
2C1D Pointer på input-buffer.
2C1E Tæl antallet af tegn i bufferen.
2C24 (Insert Flag)
2C2D Tegn fra tastaturet.

2C42 ********** Udfør EDIT hop.

2C49 EDIT hoptabel 1
2C4E Er der tegn i bufferen?
2C50 Hop, hvis ja
2C52 Er det en af cursortasterne?
2C54 Hop, hvis ikke.
2C56 Er der cursortast og SHFT/CTRL?
2C58 Hop, hvis ja.
2C5A EDIT hoptabel 2

2C72 ********** EDIT hoptabel 1

2C72 db 13 Antal elementer

```
2C73    dw 2D8A  Indføj tegn

2C75    db FC
2C76    dw 2CD0  ESC

2C78    db EF
2C79    dw 2CCE  Ingen effekt

2C7B    db 0D
2C7C    dw 2CF2  ENTER

2C7E    db F0
2C7F    dw 2D3C  CRSR UP (buffer)

2C81    db F1
2C82    dw 2D0A  CRSR DWN (buffer)

2C84    db F2
2C85    dw 2D34  CRSR LEFT (buffer)

2C87    db F3
2C88    dw 2D02  CRSR RGHT (buffer)

2C8A    db F8
2C8B    dw 2D4F  CTRL & CRSR UP

2C8D    db F9
2C8E    dw 2D1D  CTRL & CRSR DWN

2C90    db FA
2C91    dw 2D45  CTRL & CRSR LEFT

2C93    db FB
2C94    dw 2D14  CTRL & CRSR RGHT

2C96    db F4
2C97    dw 2E21  SHFT & CRSR UP

2C99    db F5
2C9A    dw 2E26  SHFT & CRSR DWN

2C9C    db F6
2C9D    dw 2E1C  SHFT & CRSR LEFT

2C9F    db F7
2CA0    dw 2E17  SHFT & CRSR RGHT

2CA2    db E0
2CA3    dw 2E65  COPY
```

```
2CA5    db 7F
2CA6    dw 2DC3  DEL

2CA8    db 10
2CA9    dw 2DCD  CLR

2CAB    db E1
2CAC    dw 2D81  CTRL & TAB (Flip Insert)

2CAE ********************************* EDIT hoptabel 2

2CAE    db 04       Antal elementer.

2CAF    dw 2CFE  BELL

2CB1    db F0
2CB2    dw 2CBD  CRSR UP

2CB4    db F1
2CB5    dw 2CC1  CRSR DWN

2CB7    db F2
2CB8    dw 2CC9  CRSR LEFT

2CBA    db F3
2CBB    dw 2CC5  CRSR RGHT

2CBD ***************************************** CRSR UP

2CC1 ***************************************** CRSR DWN

2CC5 ***************************************** CRSR RGHT

2CC9 ***************************************** CRSR LEFT

2CCB    TXT OUTPUT

2CD0 ************************************************ ESC

2CD0    ENTER
2CD4    *BREAK*-Melding
2CD7    ENTER
2CDA    TXT GET CURSOR
2CE0    CR
2CE2    TXT OUTPUT
2CE5    CRSR DWN

2CEA ********************************** *BREAK*-Melding
```

2CEA 2A 42 72 65 61 6B 2A 00 *BREAK*

2CF1 ********** ENTER

2CFC Sæt carry-flag.

2CFE ********** BELL

2CFE BELL

2D02 ********** CRSR RGHT (Buffer)

2D07 BELL

2D0A ********** CRSR DWN (Buffer)

2D10 BELL

2D14 ********** CTRL & CRSR RGHT

2D1D ********** CTRL & CRSR DWN

2D34 ********** CRSR LEFT (Buffer)

2D39 BELL

2D3C ********** CRSR UP (Buffer)

2D41 BELL

2D45 ********** CTRL & CRSR LEFT

2D4F ********** CTRL & CRSR UP

2D74 TXT GET WINDOW
2D7B TXT GET CURSOR

2D81 ********** CTRL & TAB (Filp Insert)

2D81 (Insert Flag)
2D85 (Insert Flag)

2D8A ********** Indføj tegn.

2D8D (Insert Flag)
2DA1 BELL

2DC3 ********** DEL

```
2DC8    BELL

2DCD ********************************************** CLR

2DCF    BELL
2E0E    TXT VALIDATE

2E17 ******************************** SHFT & CRSR RGHT

2E1C ******************************** SHFT & CRSR LEFT

2E21 ********************************** SHFT & CRSR UP

2E26 ********************************** SHFT & CRSR DWN

2E2E    TXT GET CURSOR
2E37    TXT VALIDATE
2E4A    TXT PLACE/REMOVE CURSOR
2E4F    TXT PLACE/REMOVE CURSOR
2E57    TXT GET CURSOR
2E5B    TXT SET CURSOR
2E62    TXT SET CURSOR

2E65 ********************************************** COPY

2E67    TXT GET CURSOR
2E74    TXT GET CURSOR
2E7C    TXT SET CURSOR
2E7F    TXT PLACE/REMOVE CURSOR
2E82    TXT RD CHAR
2E87    TXT SET CURSOR
2E8E    TXT VALIDATE
2E9C    Indføj tegn
2E9F    BELL
2ED3    TXT GET CURSOR
2ED9    TXT VALIDATE
2EDD    TXT OUTPUT
2EE7    TXT GET CURSOR
2EF4    TXT GET CURSOR
2EFB    TXT SET CURSOR
2F07    TXT GET CURSOR
2F0E    TXT VALIDATE
2F19    TXT VALIDATE
2F2A    TXT GET CURSOR
2F2F    TXT VALIDATE
2F3C    TXT WR CHAR
2F40    TXT GET CURSOR

2F56 ******************************** Tegn fra keyboard
```

2F56 TXT GET CURSOR
2F5A TXT VALIDATE
2F60 KM WAIT CHAR
2F63 TXT CUR ON
2F66 TXT GET CURSOR
2F6D KM WAIT CHAR
2F70 TXT CUR OFF

2.6. KARAKTERGENERATOREN

Vi er ikke af den opfattelse, at bogen ikke indeholder stof nok, eller at vi ønsker at kede læseren med de følgende sider - men vi er af den overbevisning, at tegnsættet er et vigtigt middel, der endda kan ændres i BASIC's kommandoregi.

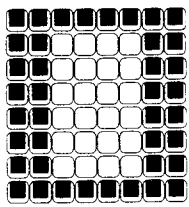
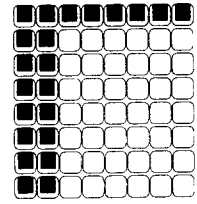
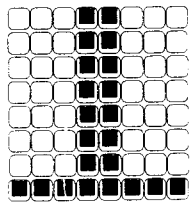
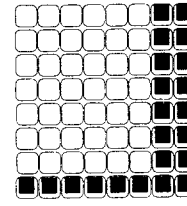
Så man ikke i egne anvendelser hele tiden skal opfinde hjulet på ny, f.eks. ved opbygning af nationale tegnsæt, kan man nøjes med at pynte på eks. "A" og skabe et "Å".

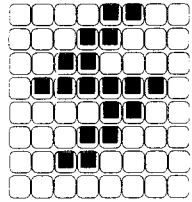
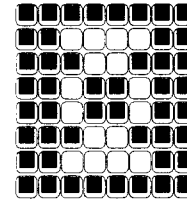
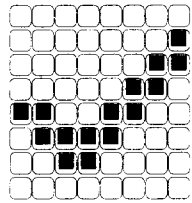
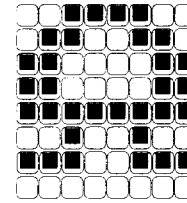
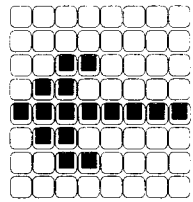
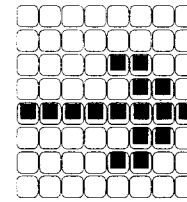
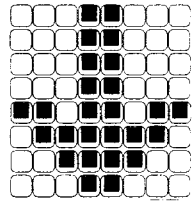
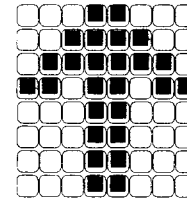
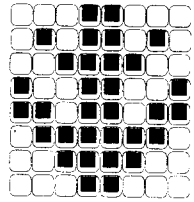
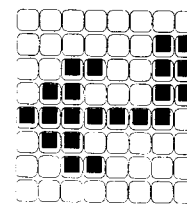
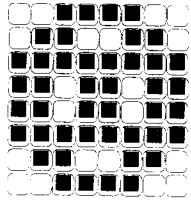
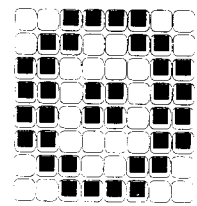
Hvorfor det er relevant, at man stifter bekendtskab med de forhåndenværende tegn, er hurtigt forklaret:

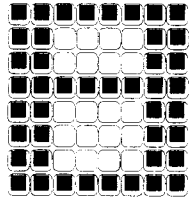
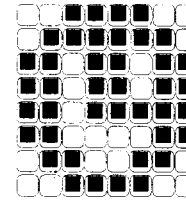
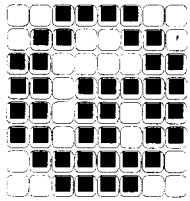
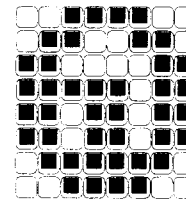
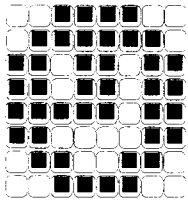
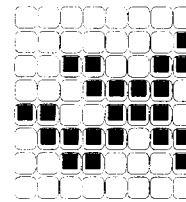
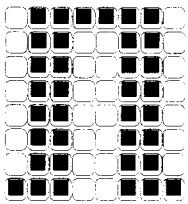
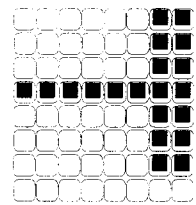
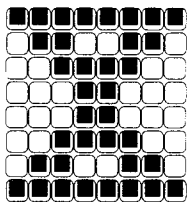
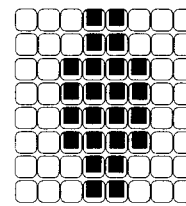
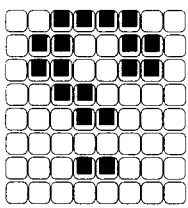
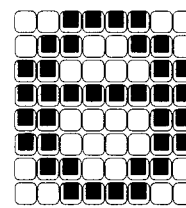
Det er sikkert ikke undgået læserens opmærksomhed, at alle vertikale linieelementer er sammensat af mindst to punkter. Grunden til det er, at det kan være vanskeligt at genfinde et enkelt punkt på skærmen, hvilket især gør sig gældende ved brug af en farvemonitor. Her kan skærmens skyggemaske let komme til at skjule en enkelt pixel.

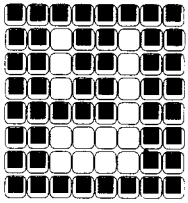
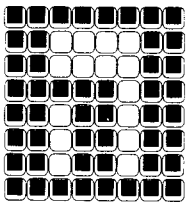
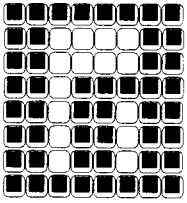
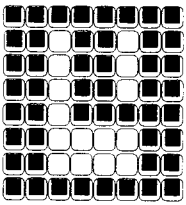
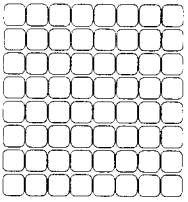
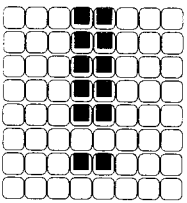
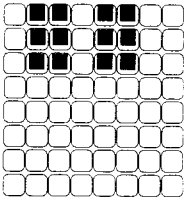
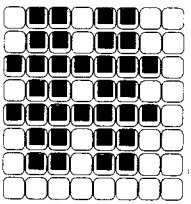
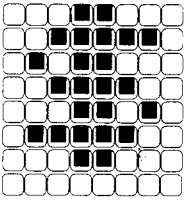
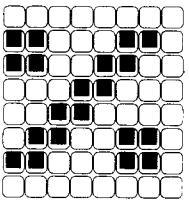
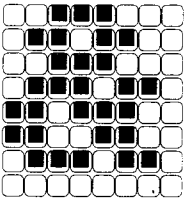
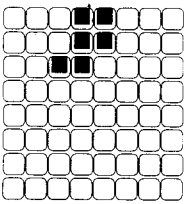
Lad det derfor altid være en regel, at lodrette liniepixels optræder parvis; men kig alligevel efter, om ikke et af de 256 tegn i karaktergeneratoren kan bruges, inden man går igang med at opbygge sit eget!

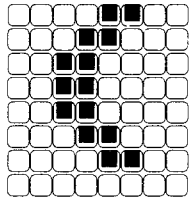
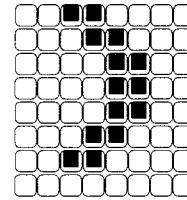
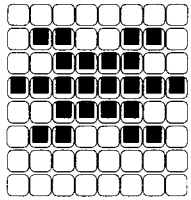
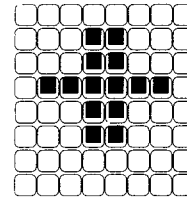
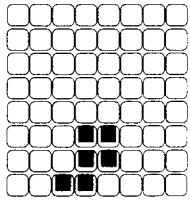
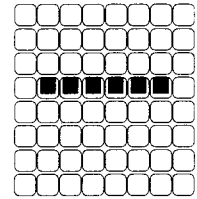
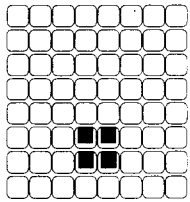
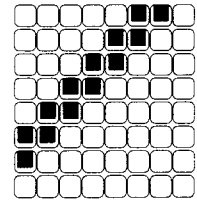
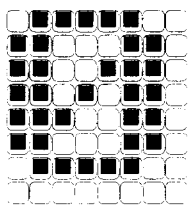
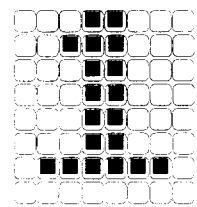
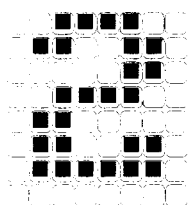
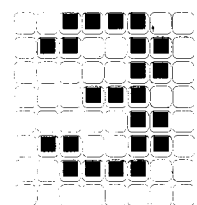
KARAKTERER

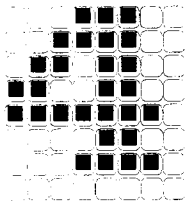
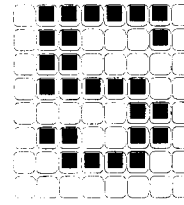
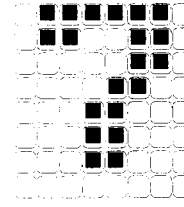
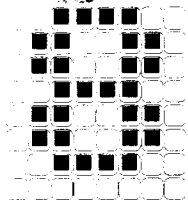
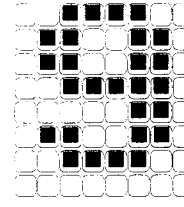
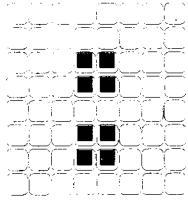
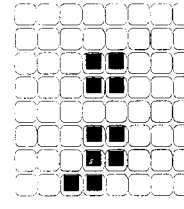
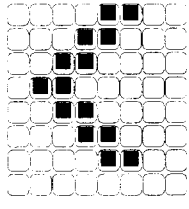
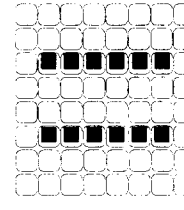
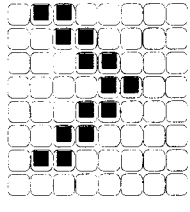
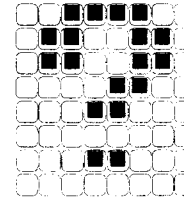
3800	FF		3808	FF	
3801	C3		3809	C0	
3802	C3		380A	C0	
3803	C3		380B	C0	
3804	C3		380C	C0	
3805	C3		380D	C0	
3806	C3		380E	C0	
3807	FF		380F	C0	
3810	18		3818	03	
3811	18		3819	03	
3812	18		381A	03	
3813	18		381B	03	
3814	18		381C	03	
3815	18		381D	03	
3816	18		381E	03	
3817	FF		381F	FF	

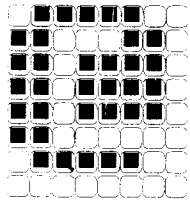
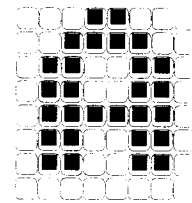
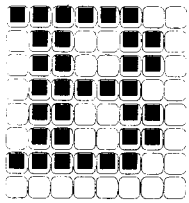
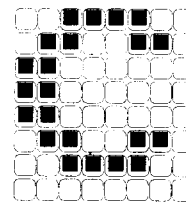
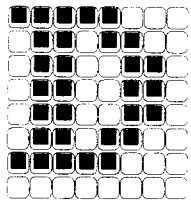
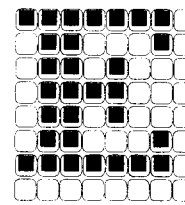
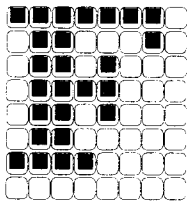
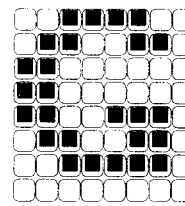
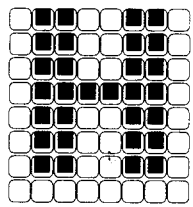
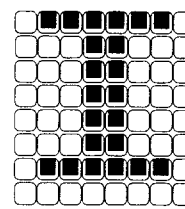
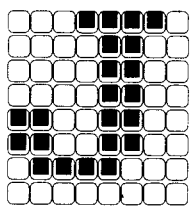
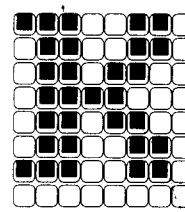
3820	0C		3828	FF	
3821	18		3829	C3	
3822	30		382A	E7	
3823	7E		382B	DB	
3824	0C		382C	DB	
3825	18		382D	E7	
3826	30		382E	C3	
3827	00		382F	FF	
3830	00		3838	3C	
3831	01		3839	66	
3832	03		383A	C3	
3833	06		383B	C3	
3834	CC		383C	FF	
3835	78		383D	24	
3836	30		383E	E7	
3837	00		383F	00	
3840	00		3848	00	
3841	00		3849	00	
3842	30		384A	0C	
3843	60		384B	06	
3844	FF		384C	FF	
3845	60		384D	06	
3846	30		384E	0C	
3847	00		384F	00	
3850	18		3858	18	
3851	18		3859	3C	
3852	18		385A	7E	
3853	18		385B	DB	
3854	DB		385C	18	
3855	7E		385D	18	
3856	3C		385E	18	
3857	18		385F	18	
3860	18		3868	00	
3861	5A		3869	03	
3862	3C		386A	33	
3863	99		386B	63	
3864	DB		386C	FE	
3865	7E		386D	60	
3866	3C		386E	30	
3867	18		386F	00	
3870	3C		3878	3C	
3871	66		3879	66	
3872	FF		387A	C3	
3873	DB		387B	DB	
3874	DB		387C	DB	
3875	FF		387D	C3	
3876	66		387E	66	
3877	3C		387F	3C	

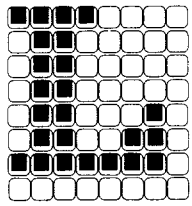
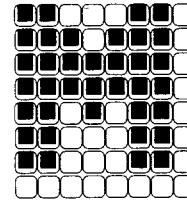
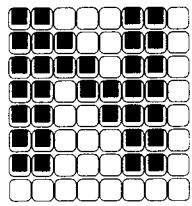
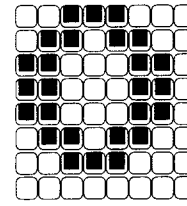
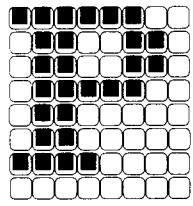
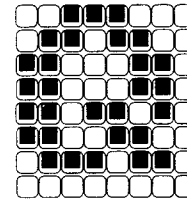
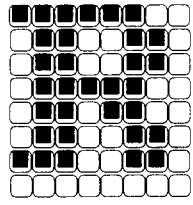
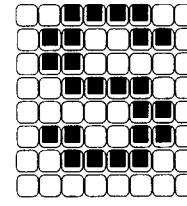
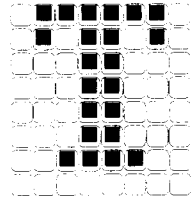
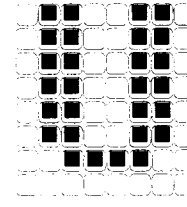
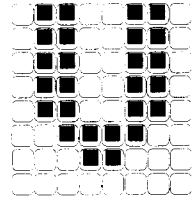
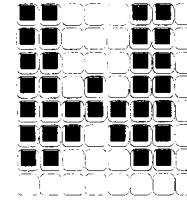
3880	FF		3888	3C	
3881	C3		3889	7E	
3882	C3		388A	DB	
3883	FF		388B	DB	
3884	C3		388C	DF	
3885	C3		388D	C3	
3886	C3		388E	66	
3887	FF		388F	3C	
3890	3C		3898	3C	
3891	66		3899	66	
3892	C3		389A	C3	
3893	DF		389B	FB	
3894	DB		389C	DB	
3895	DB		389D	DB	
3896	7E		389E	7E	
3897	3C		389F	3C	
38A0	3C		38A8	00	
38A1	7E		38A9	01	
38A2	DB		38AA	33	
38A3	DB		38AB	1E	
38A4	FB		38AC	CE	
38A5	C3		38AD	7B	
38A6	66		38AE	31	
38A7	3C		38AF	00	
38B0	7E		38B8	03	
38B1	66		38B9	03	
38B2	66		38BA	03	
38B3	66		38BB	FF	
38B4	66		38BC	03	
38B5	66		38BD	03	
38B6	66		38BE	03	
38B7	E7		38BF	00	
38C0	FF		38C8	18	
38C1	66		38C9	18	
38C2	3C		38CA	3C	
38C3	18		38CB	3C	
38C4	18		38CC	3C	
38C5	3C		38CD	3C	
38C6	66		38CE	18	
38C7	FF		38CF	18	
38D0	3C		38D8	3C	
38D1	66		38D9	66	
38D2	66		38DA	C3	
38D3	30		38DB	FF	
38D4	18		38DC	C3	
38D5	00		38DD	C3	
38D6	18		38DE	66	
38D7	00		38DF	3C	

38E0	FF		38E8	FF	
38E1	DB		38E9	C3	
38E2	DB		38EA	C3	
38E3	DB		38EB	FB	
38E4	FB		38EC	DB	
38E5	C3		38ED	DB	
38E6	C3		38EE	DB	
38E7	FF		38EF	FF	
38F0	FF		38F8	FF	
38F1	C3		38F9	DB	
38F2	C3		38FA	DB	
38F3	DF		38FB	DB	
38F4	DB		38FC	DF	
38F5	DB		38FD	C3	
38F6	DB		38FE	C3	
38F7	FF		38FF	FF	
3900	00		3908	18	
3901	00		3909	18	
3902	00		390A	18	
3903	00		390B	18	
3904	00		390C	18	
3905	00		390D	00	
3906	00		390E	18	
3907	00		390F	00	
3910	6C		3918	6C	
3911	6C		3919	6C	
3912	6C		391A	FE	
3913	00		391B	6C	
3914	00		391C	FE	
3915	00		391D	6C	
3916	00		391E	6C	
3917	00		391F	00	
3920	18		3928	00	
3921	3E		3929	C6	
3922	58		392A	CC	
3923	3C		392B	18	
3924	1A		392C	30	
3925	7C		392D	66	
3926	18		392E	C6	
3927	00		392F	00	
3930	38		3938	18	
3931	6C		3939	18	
3932	38		393A	30	
3933	76		393B	00	
3934	DC		393C	00	
3935	CC		393D	00	
3936	76		393E	00	
3937	00		393F	00	

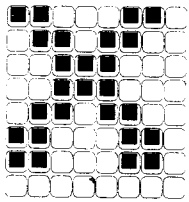
3940	0C		3948	30	
3941	18		3949	18	
3942	30		394A	0C	
3943	30		394B	0C	
3944	30		394C	0C	
3945	18		394D	18	
3946	0C		394E	30	
3947	00		394F	00	
3950	00		3958	00	
3951	66		3959	18	
3952	3C		395A	18	
3953	FF		395B	7E	
3954	3C		395C	18	
3955	66		395D	18	
3956	00		395E	00	
3957	00		395F	00	
3960	00		3968	00	
3961	00		3969	00	
3962	00		396A	00	
3963	00		396B	7E	
3964	00		396C	00	
3965	18		396D	00	
3966	18		396E	00	
3967	30		396F	00	
3970	00		3978	06	
3971	00		3979	0C	
3972	00		397A	18	
3973	00		397B	30	
3974	00		397C	60	
3975	18		397D	C0	
3976	18		397E	80	
3977	00		397F	00	
3980	7C		3988	18	
3981	C6		3989	38	
3982	CE		398A	18	
3983	D6		398B	18	
3984	E6		398C	18	
3985	C6		398D	18	
3986	7C		398E	7E	
3987	00		398F	00	
3990	3C		3998	3C	
3991	66		3999	66	
3992	06		399A	06	
3993	3C		399B	1C	
3994	60		399C	06	
3995	66		399D	66	
3996	7E		399E	3C	
3997	00		399F	00	

39A0	1C		39A8	7E	
39A1	3C		39A9	62	
39A2	6C		39AA	60	
39A3	CC		39AB	7C	
39A4	FE		39AC	06	
39A5	0C		39AD	66	
39A6	1E		39AE	3C	
39A7	00		39AF	00	
39B0	3C		39B8	7E	
39B1	66		39B9	66	
39B2	60		39BA	06	
39B3	7C		39BB	0C	
39B4	66		39BC	18	
39B5	66		39BD	18	
39B6	3C		39BE	18	
39B7	00		39BF	00	
39C0	3C		39C8	3C	
39C1	66		39C9	66	
39C2	66		39CA	66	
39C3	3C		39CB	3E	
39C4	66		39CC	06	
39C5	66		39CD	66	
39C6	3C		39CE	3C	
39C7	00		39CF	00	
39D0	00		39D8	00	
39D1	00		39D9	00	
39D2	18		39DA	18	
39D3	18		39DB	18	
39D4	00		39DC	00	
39D5	18		39DD	18	
39D6	18		39DE	18	
39D7	00		39DF	30	
39E0	0C		39E8	00	
39E1	18		39E9	00	
39E2	30		39EA	7E	
39E3	60		39EB	00	
39E4	30		39EC	00	
39E5	18		39ED	7E	
39E6	0C		39EE	00	
39E7	00		39EF	00	
39F0	60		39F8	3C	
39F1	30		39F9	66	
39F2	18		39FA	66	
39F3	0C		39FB	0C	
39F4	18		39FC	18	
39F5	30		39FD	00	
39F6	60		39FE	18	
39F7	00		39FF	00	

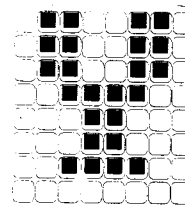
3A00	7C		3A08	18	
3A01	C6		3A09	3C	
3A02	DE		3A0A	66	
3A03	DE		3A0B	66	
3A04	DE		3A0C	7E	
3A05	C0		3A0D	66	
3A06	7C		3A0E	66	
3A07	00		3A0F	00	
3A10	FC		3A18	3C	
3A11	66		3A19	66	
3A12	66		3A1A	C0	
3A13	7C		3A1B	C0	
3A14	66		3A1C	C0	
3A15	66		3A1D	66	
3A16	FC		3A1E	3C	
3A17	00		3A1F	00	
3A20	F8		3A28	FE	
3A21	6C		3A29	62	
3A22	66		3A2A	68	
3A23	66		3A2B	78	
3A24	66		3A2C	68	
3A25	6C		3A2D	62	
3A26	F8		3A2E	FE	
3A27	00		3A2F	00	
3A30	FE		3A38	3C	
3A31	62		3A39	66	
3A32	68		3A3A	C0	
3A33	78		3A3B	C0	
3A34	68		3A3C	CE	
3A35	60		3A3D	66	
3A36	F0		3A3E	3E	
3A37	00		3A3F	00	
3A40	66		3A48	7E	
3A41	66		3A49	18	
3A42	66		3A4A	18	
3A43	7E		3A4B	18	
3A44	66		3A4C	18	
3A45	66		3A4D	18	
3A46	66		3A4E	7E	
3A47	00		3A4F	00	
3A50	1E		3A58	E6	
3A51	0C		3A59	66	
3A52	0C		3A5A	6C	
3A53	0C		3A5B	78	
3A54	CC		3A5C	6C	
3A55	CC		3A5D	66	
3A56	78		3A5E	E6	
3A57	00		3A5F	00	

3A60	F0		3A68	C6	
3A61	60		3A69	EE	
3A62	60		3A6A	FE	
3A63	60		3A6B	FE	
3A64	62		3A6C	D6	
3A65	66		3A6D	C6	
3A66	FE		3A6E	C6	
3A67	00		3A6F	00	
3A70	C6		3A78	38	
3A71	E6		3A79	6C	
3A72	F6		3A7A	C6	
3A73	DE		3A7B	C6	
3A74	CE		3A7C	C6	
3A75	C6		3A7D	6C	
3A76	C6		3A7E	38	
3A77	00		3A7F	00	
3A80	FC		3A88	38	
3A81	66		3A89	6C	
3A82	66		3A8A	C6	
3A83	7C		3A8B	C6	
3A84	60		3A8C	DA	
3A85	60		3A8D	CC	
3A86	F0		3A8E	76	
3A87	00		3A8F	00	
3A90	FC		3A98	3C	
3A91	66		3A99	66	
3A92	66		3A9A	60	
3A93	7C		3A9B	3C	
3A94	6C		3A9C	06	
3A95	66		3A9D	66	
3A96	E6		3A9E	3C	
3A97	00		3A9F	00	
3AA0	7E		3AA8	66	
3AA1	5A		3AA9	66	
3AA2	18		3AAA	66	
3AA3	18		3AAB	66	
3AA4	18		3AAC	66	
3AA5	18		3AAD	66	
3AA6	3C		3AAE	3C	
3AA7	00		3AAF	00	
3AB0	66		3AB8	C6	
3AB1	66		3AB9	C6	
3AB2	66		3ABA	C6	
3AB3	66		3ABB	D6	
3AB4	66		3ABC	FE	
3AB5	3C		3ABD	EE	
3AB6	18		3ABE	C6	
3AB7	00		3ABF	00	

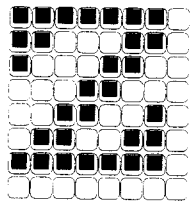
3AC0	C6
3AC1	6C
3AC2	38
3AC3	38
3AC4	6C
3AC5	C6
3AC6	C6
3AC7	00



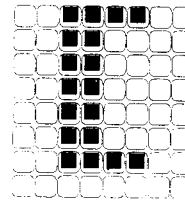
3AC8	66
3AC9	66
3ACA	66
3ACB	3C
3ACC	18
3ACD	18
3ACE	3C
3ACF	00



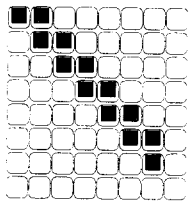
3AD0	FE
3AD1	C6
3AD2	8C
3AD3	18
3AD4	32
3AD5	66
3AD6	FE
3AD7	00



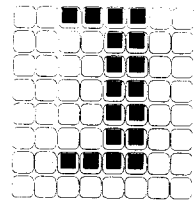
3AD8	3C
3AD9	30
3ADA	30
3ADB	30
3ADC	30
3ADD	30
3ADE	3C
3ADF	00



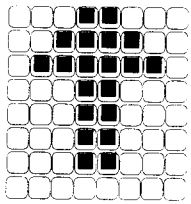
3AE0	C0
3AE1	60
3AE2	30
3AE3	18
3AE4	0C
3AE5	06
3AE6	02
3AE7	00



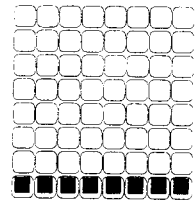
3AE8	3C
3AE9	0C
3AEA	0C
3AEB	0C
3AEC	0C
3AED	0C
3AEE	3C
3AEF	00



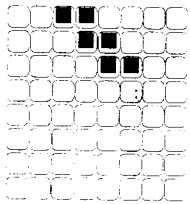
3AF0	18
3AF1	3C
3AF2	7E
3AF3	18
3AF4	18
3AF5	18
3AF6	18
3AF7	00



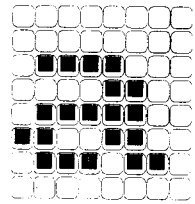
3AF8	00
3AF9	00
3AFA	00
3AFB	00
3AFC	00
3AFD	00
3AFE	00
3AFF	FF



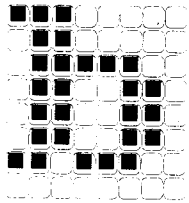
3B00	30
3B01	18
3B02	0C
3B03	00
3B04	00
3B05	00
3B06	00
3B07	00



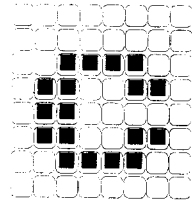
3B08	00
3B09	00
3B0A	78
3B0B	0C
3B0C	7C
3B0D	CC
3B0E	76
3B0F	00



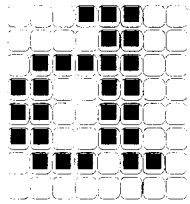
3B10	E0
3B11	60
3B12	7C
3B13	66
3B14	66
3B15	66
3B16	DC
3B17	00



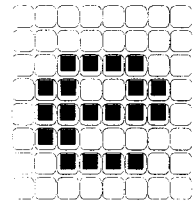
3B18	00
3B19	00
3B1A	3C
3B1B	66
3B1C	60
3B1D	66
3B1E	3C
3B1F	00



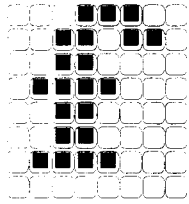
3B20	1C
3B21	0C
3B22	7C
3B23	CC
3B24	CC
3B25	CC
3B26	76
3B27	00



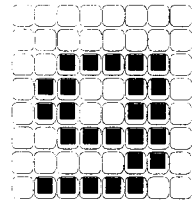
3B28	00
3B29	00
3B2A	3C
3B2B	66
3B2C	7E
3B2D	60
3B2E	3C
3B2F	00



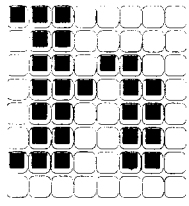
3B30	1C
3B31	36
3B32	30
3B33	78
3B34	30
3B35	30
3B36	78
3B37	00



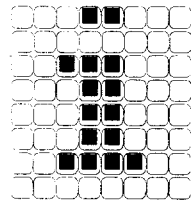
3B38	00
3B39	00
3B3A	3E
3B3B	66
3B3C	66
3B3D	3E
3B3E	06
3B3F	7C



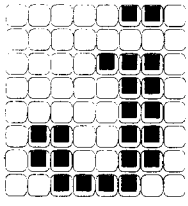
3B40	E0
3B41	60
3B42	6C
3B43	76
3B44	66
3B45	66
3B46	E6
3B47	00



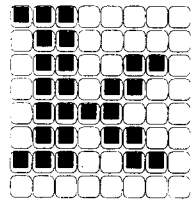
3B48	18
3B49	00
3B4A	38
3B4B	18
3B4C	18
3B4D	18
3B4E	3C
3B4F	00



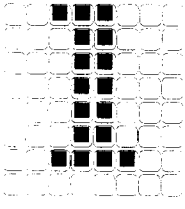
3B50	06
3B51	00
3B52	0E
3B53	06
3B54	06
3B55	66
3B56	66
3B57	3C



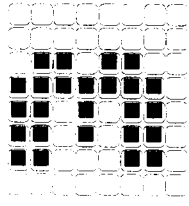
3B58	E0
3B59	60
3B5A	66
3B5B	6C
3B5C	78
3B5D	6C
3B5E	E6
3B5F	00



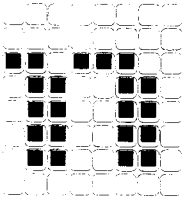
3B60	38
3B61	18
3B62	18
3B63	18
3B64	18
3B65	18
3B66	3C
3B67	00



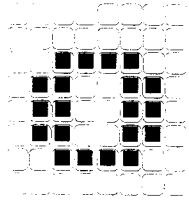
3B68	00
3B69	00
3B6A	6C
3B6B	FE
3B6C	D6
3B6D	D6
3B6E	C6
3B6F	00

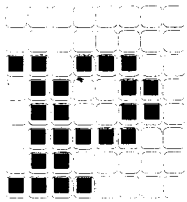
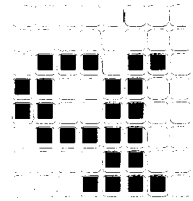
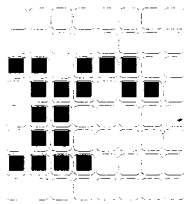
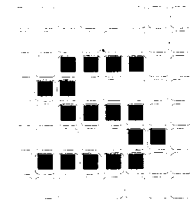
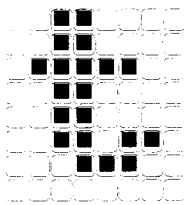
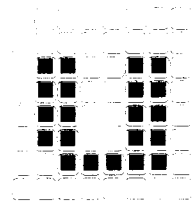
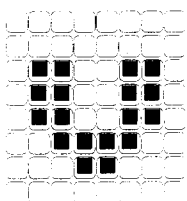
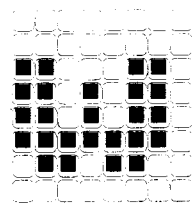
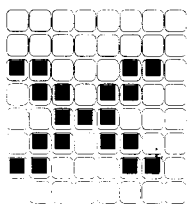
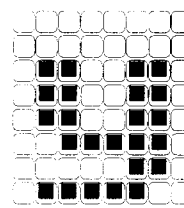
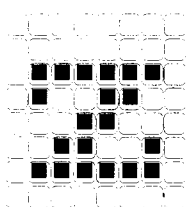
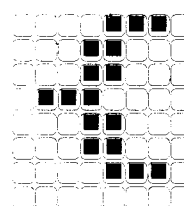


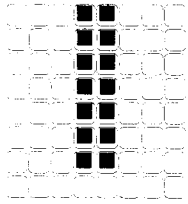
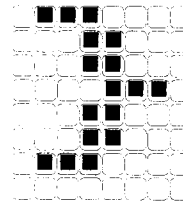
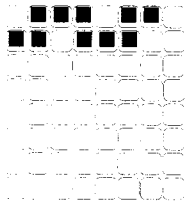
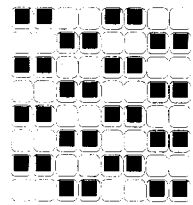
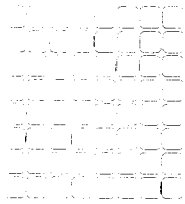
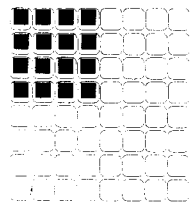
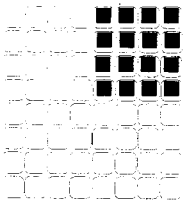
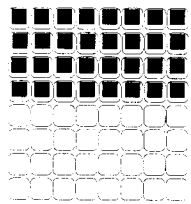
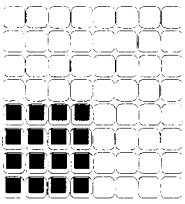
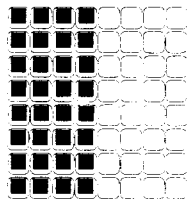
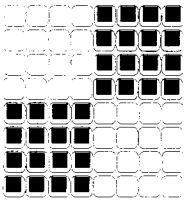
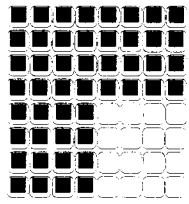
3B70	00
3B71	00
3B72	DC
3B73	66
3B74	66
3B75	66
3B76	66
3B77	00

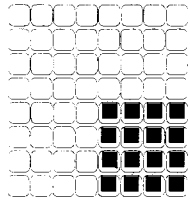
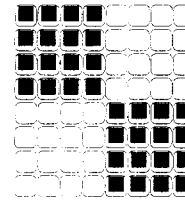
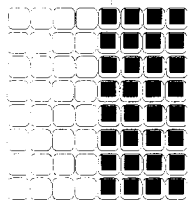
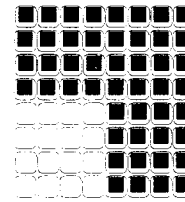
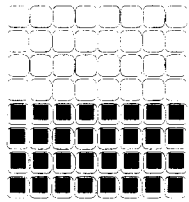
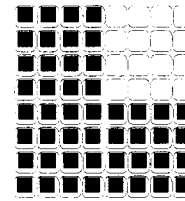
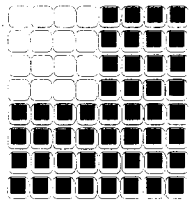
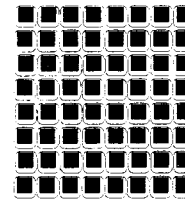
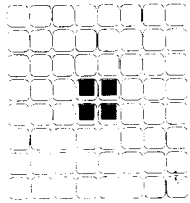
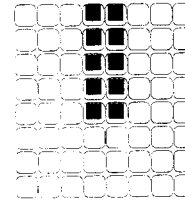
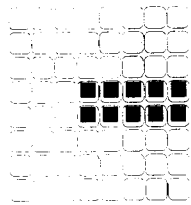
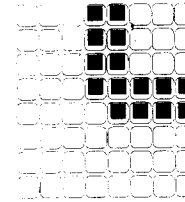


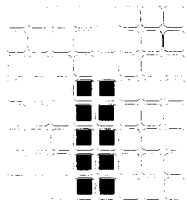
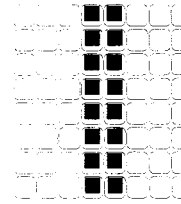
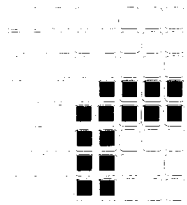
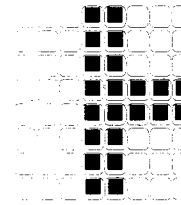
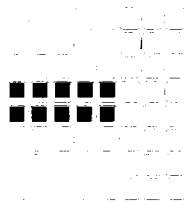
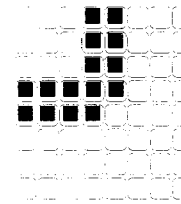
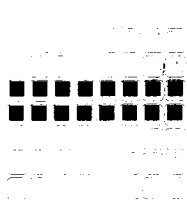
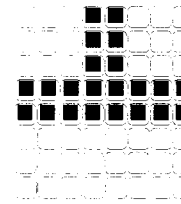
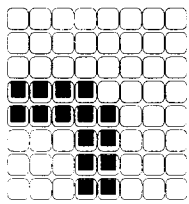
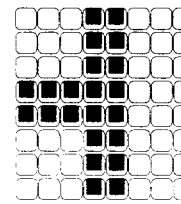
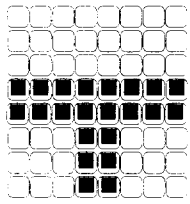
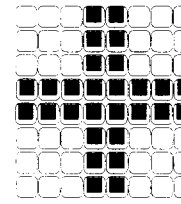
3B78	00
3B79	00
3B7A	3C
3B7B	66
3B7C	66
3B7D	66
3B7E	3C
3B7F	00

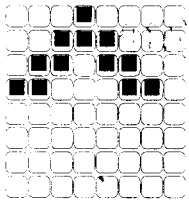
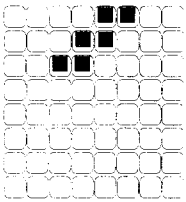
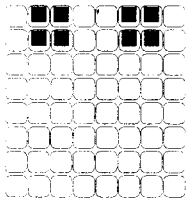
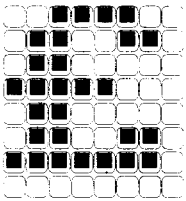
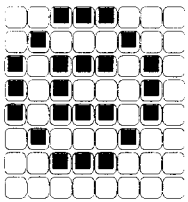
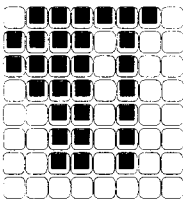
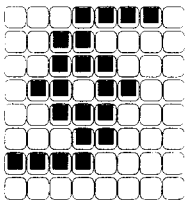
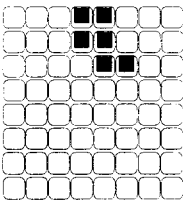
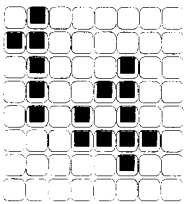
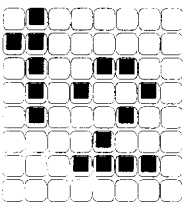
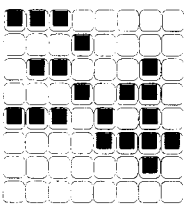
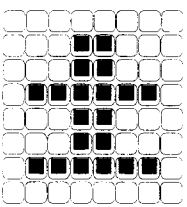


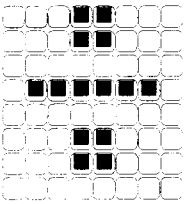
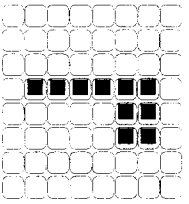
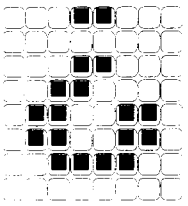
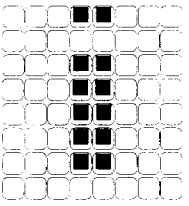
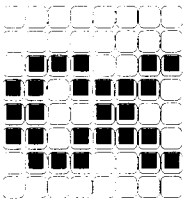
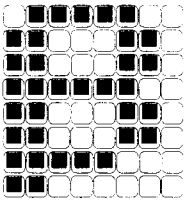
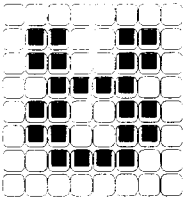
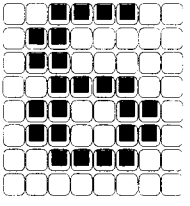
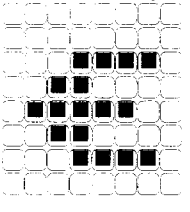
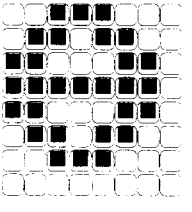
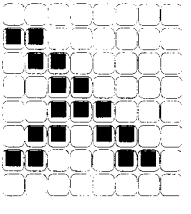
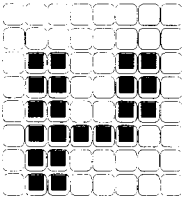
3B80	00		3B88	00	
3B81	00		3B89	00	
3B82	DC		3B8A	76	
3B83	66		3B8B	CC	
3B84	66		3B8C	CC	
3B85	7C		3B8D	7C	
3B86	60		3B8E	0C	
3B87	F0		3B8F	1E	
3B90	00		3B98	00	
3B91	00		3B99	00	
3B92	DC		3B9A	3C	
3B93	76		3B9B	60	
3B94	60		3B9C	3C	
3B95	60		3B9D	06	
3B96	F0		3B9E	7C	
3B97	00		3B9F	00	
3BA0	30		3BA8	00	
3BA1	30		3BA9	00	
3BA2	7C		3BAA	66	
3BA3	30		3BAB	66	
3BA4	30		3BAC	66	
3BA5	36		3BAD	66	
3BA6	1C		3BAE	3E	
3BA7	00		3BAF	00	
3BB0	00		3BB8	00	
3BB1	00		3BB9	00	
3BB2	66		3BBA	C6	
3BB3	66		3BBB	D6	
3BB4	66		3BBC	D6	
3BB5	3C		3BBD	FE	
3BB6	18		3BBE	6C	
3BB7	00		3BBF	00	
3BC0	00		3BC8	00	
3BC1	00		3BC9	00	
3BC2	C6		3BCA	66	
3BC3	6C		3BCB	66	
3BC4	38		3BCC	66	
3BC5	6C		3BCD	3E	
3BC6	C6		3BCE	06	
3BC7	00		3BCF	7C	
3BD0	00		3BD8	0E	
3BD1	00		3BD9	18	
3BD2	7E		3BDA	18	
3BD3	4C		3BDB	70	
3BD4	18		3BDC	18	
3BD5	32		3BDD	18	
3BD6	7E		3BDE	0E	
3BD7	00		3BDF	00	

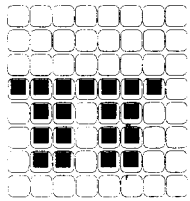
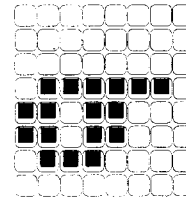
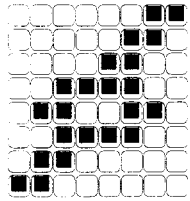
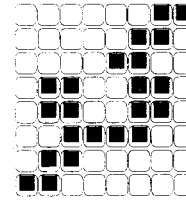
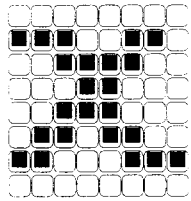
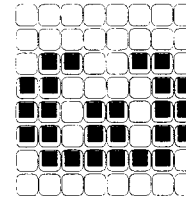
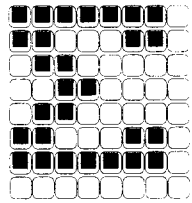
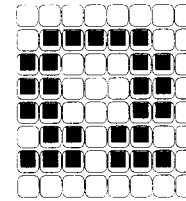
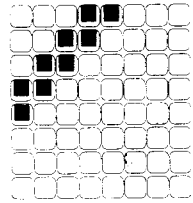
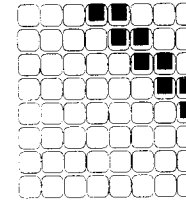
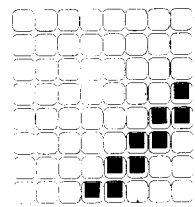
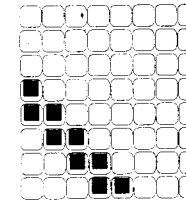
3BE0	18		3BE8	70	
3BE1	18		3BE9	18	
3BE2	18		3BEA	18	
3BE3	18		3BEB	0E	
3BE4	18		3BEC	18	
3BE5	18		3BED	18	
3BE6	18		3BEE	70	
3BE7	00		3BEF	00	
3BF0	76		3BF8	CC	
3BF1	DC		3BF9	33	
3BF2	00		3BFA	CC	
3BF3	00		3BFB	33	
3BF4	00		3BFC	CC	
3BF5	00		3BFD	33	
3BF6	00		3BFE	CC	
3BF7	00		3BFF	33	
3C00	00		3C08	F0	
3C01	00		3C09	F0	
3C02	00		3C0A	F0	
3C03	00		3C0B	F0	
3C04	00		3C0C	00	
3C05	00		3C0D	00	
3C06	00		3C0E	00	
3C07	00		3C0F	00	
3C10	0F		3C18	FF	
3C11	0F		3C19	FF	
3C12	0F		3C1A	FF	
3C13	0F		3C1B	FF	
3C14	00		3C1C	00	
3C15	00		3C1D	00	
3C16	00		3C1E	00	
3C17	00		3C1F	00	
3C20	00		3C28	F0	
3C21	00		3C29	F0	
3C22	00		3C2A	F0	
3C23	00		3C2B	F0	
3C24	F0		3C2C	F0	
3C25	F0		3C2D	F0	
3C26	F0		3C2E	F0	
3C27	F0		3C2F	F0	
3C30	0F		3C38	FF	
3C31	0F		3C39	FF	
3C32	0F		3C3A	FF	
3C33	0F		3C3B	FF	
3C34	F0		3C3C	F0	
3C35	F0		3C3D	F0	
3C36	F0		3C3E	F0	
3C37	F0		3C3F	F0	

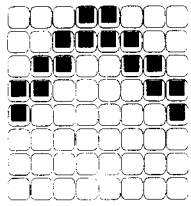
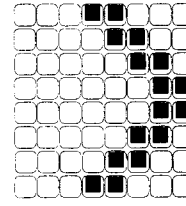
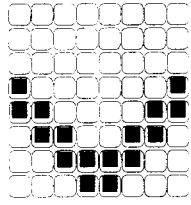
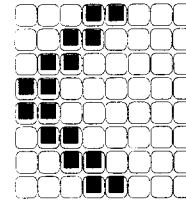
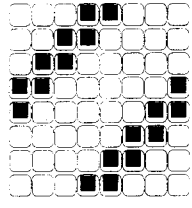
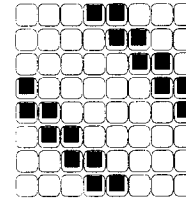
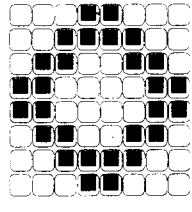
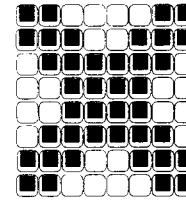
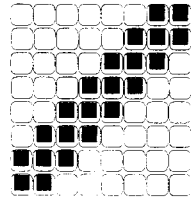
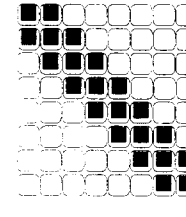
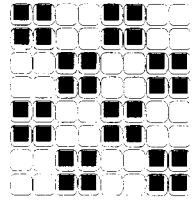
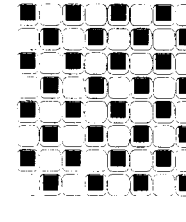
3C40	00		3C48	F0	
3C41	00		3C49	F0	
3C42	00		3C4A	F0	
3C43	00		3C4B	F0	
3C44	0F		3C4C	0F	
3C45	0F		3C4D	0F	
3C46	0F		3C4E	0F	
3C47	0F		3C4F	0F	
3C50	0F		3C58	FF	
3C51	0F		3C59	FF	
3C52	0F		3C5A	FF	
3C53	0F		3C5B	FF	
3C54	0F		3C5C	0F	
3C55	0F		3C5D	0F	
3C56	0F		3C5E	0F	
3C57	0F		3C5F	0F	
3C60	00		3C68	F0	
3C61	00		3C69	F0	
3C62	00		3C6A	F0	
3C63	00		3C6B	F0	
3C64	FF		3C6C	FF	
3C65	FF		3C6D	FF	
3C66	FF		3C6E	FF	
3C67	FF		3C6F	FF	
3C70	0F		3C78	FF	
3C71	0F		3C79	FF	
3C72	0F		3C7A	FF	
3C73	0F		3C7B	FF	
3C74	FF		3C7C	FF	
3C75	FF		3C7D	FF	
3C76	FF		3C7E	FF	
3C77	FF		3C7F	FF	
3C80	00		3C88	18	
3C81	00		3C89	18	
3C82	00		3C8A	18	
3C83	18		3C8B	18	
3C84	18		3C8C	18	
3C85	00		3C8D	00	
3C86	00		3C8E	00	
3C87	00		3C8F	00	
3C90	00		3C98	18	
3C91	00		3C99	18	
3C92	00		3C9A	18	
3C93	1F		3C9B	1F	
3C94	1F		3C9C	0F	
3C95	00		3C9D	00	
3C96	00		3C9E	00	
3C97	00		3C9F	00	

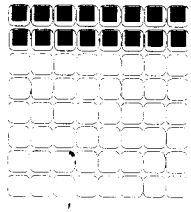
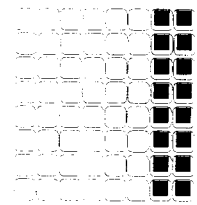
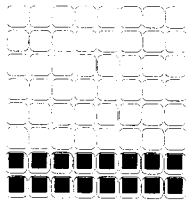
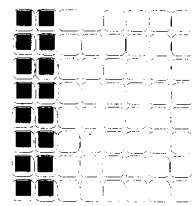
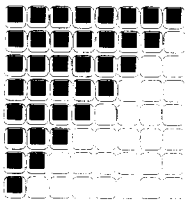
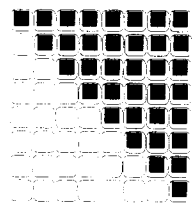
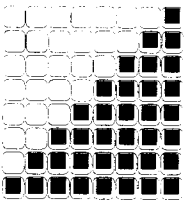
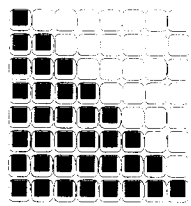
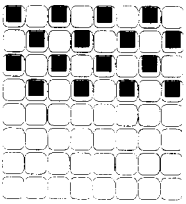
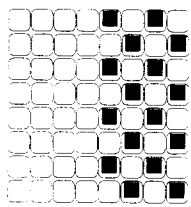
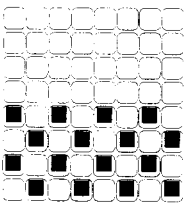
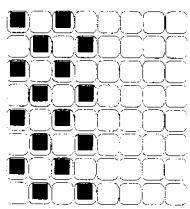
3CA0	00		3CA8	18	
3CA1	00		3CA9	18	
3CA2	00		3CAA	18	
3CA3	18		3CAB	18	
3CA4	18		3CAC	18	
3CA5	18		3CAD	18	
3CA6	18		3CAE	18	
3CA7	18		3CAF	18	
3CB0	00		3CB8	18	
3CB1	00		3CB9	18	
3CB2	00		3CBA	18	
3CB3	0F		3CBB	1F	
3CB4	1F		3CBC	1F	
3CB5	18		3CBD	18	
3CB6	18		3CBE	18	
3CB7	18		3CBF	18	
3CC0	00		3CC8	18	
3CC1	00		3CC9	18	
3CC2	00		3CCA	18	
3CC3	F8		3CCB	F8	
3CC4	F8		3CCC	F0	
3CC5	00		3CCD	00	
3CC6	00		3CCE	00	
3CC7	00		3CCF	00	
3CD0	00		3CD8	18	
3CD1	00		3CD9	18	
3CD2	00		3CDA	18	
3CD3	FF		3CDB	FF	
3CD4	FF		3CDC	FF	
3CD5	00		3CDD	00	
3CD6	00		3CDE	00	
3CD7	00		3CDF	00	
3CE0	00		3CE8	18	
3CE1	00		3CE9	18	
3CE2	00		3CEA	18	
3CE3	F0		3CEB	F8	
3CE4	F8		3CEC	F8	
3CE5	18		3CED	18	
3CE6	18		3CEE	18	
3CE7	18		3CEF	18	
3CF0	00		3CF8	18	
3CF1	00		3CF9	18	
3CF2	00		3CFA	18	
3CF3	FF		3CFB	FF	
3CF4	FF		3CFC	FF	
3CF5	18		3CFD	18	
3CF6	18		3CFE	18	
3CF7	18		3CFF	18	

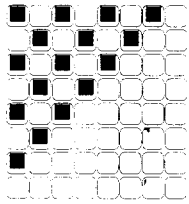
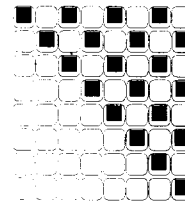
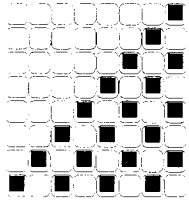
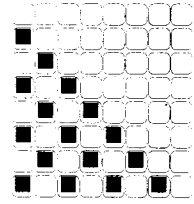
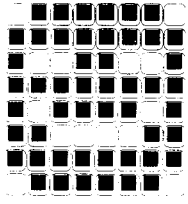
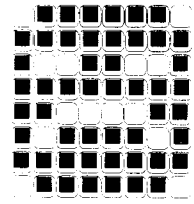
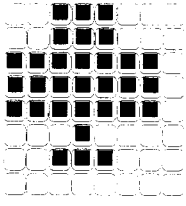
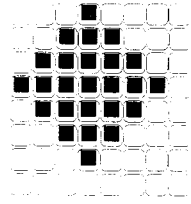
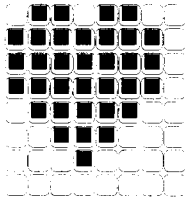
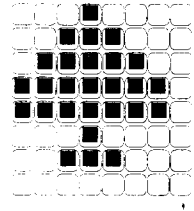
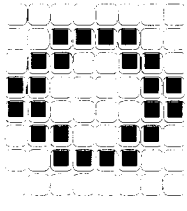
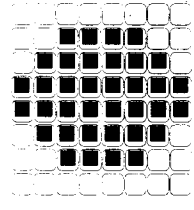
3D00	10		3D08	0C	
3D01	38		3D09	18	
3D02	6C		3D0A	30	
3D03	C6		3D0B	00	
3D04	00		3D0C	00	
3D05	00		3D0D	00	
3D06	00		3D0E	00	
3D07	00		3D0F	00	
3D10	66		3D18	3C	
3D11	66		3D19	66	
3D12	00		3D1A	60	
3D13	00		3D1B	F8	
3D14	00		3D1C	60	
3D15	00		3D1D	66	
3D16	00		3D1E	FE	
3D17	00		3D1F	00	
3D20	38		3D28	7E	
3D21	44		3D29	F4	
3D22	BA		3D2A	F4	
3D23	A2		3D2B	74	
3D24	BA		3D2C	34	
3D25	44		3D2D	34	
3D26	38		3D2E	34	
3D27	00		3D2F	00	
3D30	1E		3D38	18	
3D31	30		3D39	18	
3D32	38		3D3A	0C	
3D33	6C		3D3B	00	
3D34	38		3D3C	00	
3D35	18		3D3D	00	
3D36	F0		3D3E	00	
3D37	00		3D3F	00	
3D40	40		3D48	40	
3D41	C0		3D49	C0	
3D42	44		3D4A	4C	
3D43	4C		3D4B	52	
3D44	54		3D4C	44	
3D45	1E		3D4D	08	
3D46	04		3D4E	1E	
3D47	00		3D4F	00	
3D50	E0		3D58	00	
3D51	10		3D59	18	
3D52	62		3D5A	18	
3D53	16		3D5B	7E	
3D54	EA		3D5C	18	
3D55	0F		3D5D	18	
3D56	02		3D5E	7E	
3D57	00		3D5F	00	

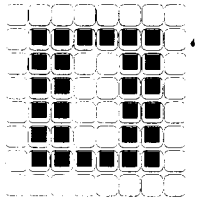
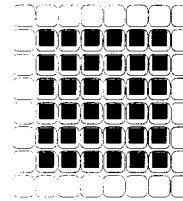
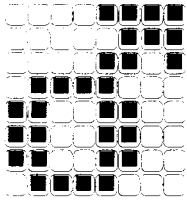
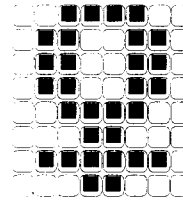
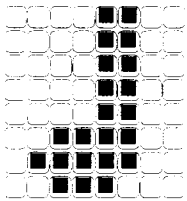
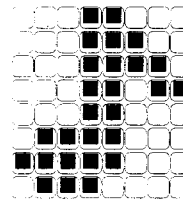
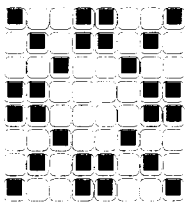
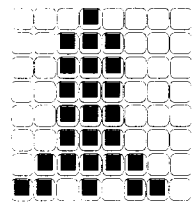
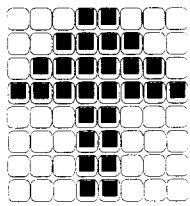
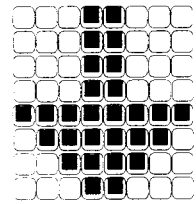
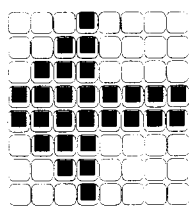
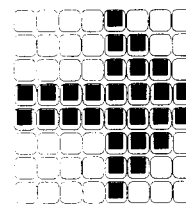
3D60	18		3D68	00	
3D61	18		3D69	00	
3D62	00		3D6A	00	
3D63	7E		3D6B	7E	
3D64	00		3D6C	06	
3D65	18		3D6D	06	
3D66	18		3D6E	00	
3D67	00		3D6F	00	
3D70	18		3D78	18	
3D71	00		3D79	00	
3D72	18		3D7A	18	
3D73	30		3D7B	18	
3D74	66		3D7C	18	
3D75	66		3D7D	18	
3D76	3C		3D7E	18	
3D77	00		3D7F	00	
3D80	00		3D88	7C	
3D81	00		3D89	C6	
3D82	73		3D8A	C6	
3D83	DE		3D8B	FC	
3D84	CC		3D8C	C6	
3D85	DE		3D8D	C6	
3D86	73		3D8E	F8	
3D87	00		3D8F	C0	
3D90	00		3D98	3C	
3D91	66		3D99	60	
3D92	66		3D9A	60	
3D93	3C		3D9B	3C	
3D94	66		3D9C	66	
3D95	66		3D9D	66	
3D96	3C		3D9E	3C	
3D97	00		3D9F	00	
3DA0	00		3DA8	38	
3DA1	00		3DA9	6C	
3DA2	1E		3DAA	C6	
3DA3	30		3DAB	FE	
3DA4	7C		3DAC	C6	
3DA5	30		3DAD	6C	
3DA6	1E		3DAE	38	
3DA7	00		3DAF	00	
3DB0	00		3DB8	00	
3DB1	C0		3DB9	00	
3DB2	60		3DBA	66	
3DB3	30		3DBB	66	
3DB4	38		3DBC	66	
3DB5	6C		3DBD	7C	
3DB6	C6		3DBE	60	
3DB7	00		3DBF	60	

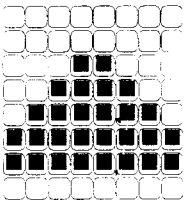
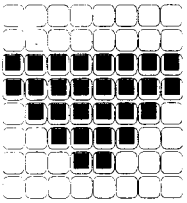
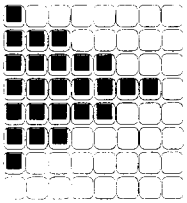
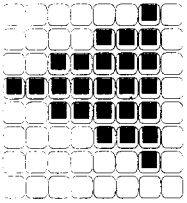
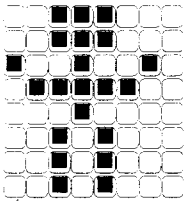
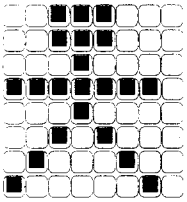
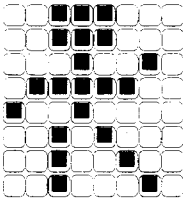
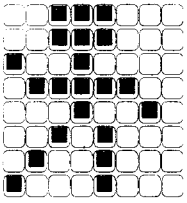
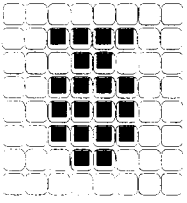
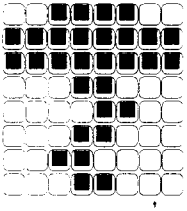
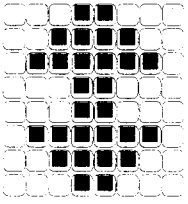
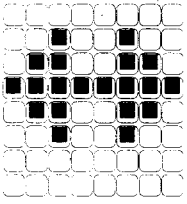
3DC0	00		3DC8	00	
3DC1	00		3DC9	00	
3DC2	00		3DCA	00	
3DC3	FE		3DCB	7E	
3DC4	6C		3DCC	D8	
3DC5	6C		3DCD	D8	
3DC6	6C		3DCE	70	
3DC7	00		3DCF	00	
3DD0	03		3DD8	03	
3DD1	06		3DD9	06	
3DD2	0C		3DDA	0C	
3DD3	3C		3DDB	66	
3DD4	66		3DDC	66	
3DD5	3C		3DDD	3C	
3DD6	60		3DDE	60	
3DD7	C0		3DDF	C0	
3DE0	00		3DE8	00	
3DE1	E6		3DE9	00	
3DE2	3C		3DEA	66	
3DE3	18		3DEB	C3	
3DE4	38		3DEC	DB	
3DE5	6C		3DED	DB	
3DE6	C7		3DEE	7E	
3DE7	00		3DEF	00	
3DF0	FE		3DF8	00	
3DF1	C6		3DF9	7C	
3DF2	60		3DFA	C6	
3DF3	30		3DFB	C6	
3DF4	60		3DFC	C6	
3DF5	C6		3DFD	6C	
3DF6	FE		3DFE	EE	
3DF7	00		3DFF	00	
3E00	18		3E08	18	
3E01	30		3E09	0C	
3E02	60		3E0A	06	
3E03	C0		3E0B	03	
3E04	80		3E0C	01	
3E05	00		3E0D	00	
3E06	00		3E0E	00	
3E07	00		3E0F	00	
3E10	00		3E18	00	
3E11	00		3E19	00	
3E12	00		3E1A	00	
3E13	01		3E1B	80	
3E14	03		3E1C	C0	
3E15	06		3E1D	60	
3E16	0C		3E1E	30	
3E17	18		3E1F	18	

3E20	18		3E28	18	
3E21	3C		3E29	0C	
3E22	66		3E2A	06	
3E23	C3		3E2B	03	
3E24	81		3E2C	03	
3E25	00		3E2D	06	
3E26	00		3E2E	0C	
3E27	00		3E2F	18	
3E30	00		3E38	18	
3E31	00		3E39	30	
3E32	00		3E3A	60	
3E33	81		3E3B	C0	
3E34	C3		3E3C	C0	
3E35	66		3E3D	60	
3E36	3C		3E3E	30	
3E37	18		3E3F	18	
3E40	18		3E48	18	
3E41	30		3E49	0C	
3E42	60		3E4A	06	
3E43	C1		3E4B	83	
3E44	83		3E4C	C1	
3E45	06		3E4D	60	
3E46	0C		3E4E	30	
3E47	18		3E4F	18	
3E50	18		3E58	C3	
3E51	3C		3E59	E7	
3E52	66		3E5A	7E	
3E53	C3		3E5B	3C	
3E54	C3		3E5C	3C	
3E55	66		3E5D	7E	
3E56	3C		3E5E	E7	
3E57	18		3E5F	C3	
3E60	03		3E68	C0	
3E61	07		3E69	E0	
3E62	0E		3E6A	70	
3E63	1C		3E6B	38	
3E64	38		3E6C	1C	
3E65	70		3E6D	0E	
3E66	E0		3E6E	07	
3E67	C0		3E6F	03	
3E70	CC		3E78	AA	
3E71	CC		3E79	55	
3E72	33		3E7A	AA	
3E73	33		3E7B	55	
3E74	CC		3E7C	AA	
3E75	CC		3E7D	55	
3E76	33		3E7E	AA	
3E77	33		3E7F	55	

3E80	FF		3E88	03	
3E81	FF		3E89	03	
3E82	00		3E8A	03	
3E83	00		3E8B	03	
3E84	00		3E8C	03	
3E85	00		3E8D	03	
3E86	00		3E8E	03	
3E87	00		3E8F	03	
3E90	00		3E98	C0	
3E91	00		3E99	C0	
3E92	00		3E9A	C0	
3E93	00		3E9B	C0	
3E94	00		3E9C	C0	
3E95	00		3E9D	C0	
3E96	FF		3E9E	C0	
3E97	FF		3E9F	C0	
3EA0	FF		3EA8	FF	
3EA1	FE		3EA9	7F	
3EA2	FC		3EAA	3F	
3EA3	F8		3EAB	1F	
3EA4	F0		3EAC	0F	
3EA5	E0		3EAD	07	
3EA6	C0		3EAE	03	
3EA7	80		3EAF	01	
3EB0	01		3EB8	80	
3EB1	03		3EB9	C0	
3EB2	07		3EBA	E0	
3EB3	0F		3EBB	F0	
3EB4	1F		3EBC	F8	
3EB5	3F		3EBD	FC	
3EB6	7F		3EBE	FE	
3EB7	FF		3EBF	FF	
3EC0	AA		3EC8	0A	
3EC1	55		3EC9	05	
3EC2	AA		3ECA	0A	
3EC3	55		3ECB	05	
3EC4	00		3ECC	0A	
3EC5	00		3ECD	05	
3EC6	00		3ECE	0A	
3EC7	00		3ECF	05	
3ED0	00		3ED8	A0	
3ED1	00		3ED9	50	
3ED2	00		3EDA	A0	
3ED3	00		3EDB	50	
3ED4	AA		3EDC	A0	
3ED5	55		3EDD	50	
3ED6	AA		3EDE	A0	
3ED7	55		3EDF	50	

3EE0	AA		3EE8	AA	
3EE1	54		3EE9	55	
3EE2	A8		3EEA	2A	
3EE3	50		3EEB	15	
3EE4	A0		3EEC	0A	
3EE5	40		3EED	05	
3EE6	80		3EEE	02	
3EE7	00		3EEF	01	
3EF0	01		3EF8	00	
3EF1	02		3EF9	80	
3EF2	05		3EFA	40	
3EF3	0A		3EFB	A0	
3EF4	15		3EFC	50	
3EF5	2A		3EFD	A8	
3EF6	55		3EFE	54	
3EF7	AA		3EFF	AA	
3F00	7E		3F08	7E	
3F01	FF		3F09	FF	
3F02	99		3F0A	99	
3F03	FF		3F0B	FF	
3F04	BD		3F0C	C3	
3F05	C3		3F0D	BD	
3F06	FF		3F0E	FF	
3F07	7E		3F0F	7E	
3F10	38		3F18	10	
3F11	38		3F19	38	
3F12	FE		3F1A	7C	
3F13	FE		3F1B	FE	
3F14	FE		3F1C	7C	
3F15	10		3F1D	38	
3F16	38		3F1E	10	
3F17	00		3F1F	00	
3F20	6C		3F28	10	
3F21	FE		3F29	38	
3F22	FE		3F2A	7C	
3F23	FE		3F2B	FE	
3F24	7C		3F2C	FE	
3F25	38		3F2D	10	
3F26	10		3F2E	38	
3F27	00		3F2F	00	
3F30	00		3F38	00	
3F31	3C		3F39	3C	
3F32	66		3F3A	7E	
3F33	C3		3F3B	FF	
3F34	C3		3F3C	FF	
3F35	66		3F3D	7E	
3F36	3C		3F3E	3C	
3F37	00		3F3F	00	

3F40	00		3F48	00	
3F41	7E		3F49	7E	
3F42	66		3F4A	7E	
3F43	66		3F4B	7E	
3F44	66		3F4C	7E	
3F45	66		3F4D	7E	
3F46	7E		3F4E	7E	
3F47	00		3F4F	00	
3F50	0F		3F58	3C	
3F51	07		3F59	66	
3F52	0D		3F5A	66	
3F53	78		3F5B	66	
3F54	CC		3F5C	3C	
3F55	CC		3F5D	18	
3F56	CC		3F5E	7E	
3F57	78		3F5F	18	
3F60	0C		3F68	18	
3F61	0C		3F69	1C	
3F62	0C		3F6A	1E	
3F63	0C		3F6B	1B	
3F64	0C		3F6C	18	
3F65	3C		3F6D	78	
3F66	7C		3F6E	F8	
3F67	38		3F6F	70	
3F70	99		3F78	10	
3F71	5A		3F79	38	
3F72	24		3F7A	38	
3F73	C3		3F7B	38	
3F74	C3		3F7C	38	
3F75	24		3F7D	38	
3F76	5A		3F7E	7C	
3F77	99		3F7F	D6	
3F80	18		3F88	18	
3F81	3C		3F89	18	
3F82	7E		3F8A	18	
3F83	FF		3F8B	18	
3F84	18		3F8C	FF	
3F85	18		3F8D	7E	
3F86	18		3F8E	3C	
3F87	18		3F8F	18	
3F90	10		3F98	08	
3F91	30		3F99	0C	
3F92	70		3F9A	0E	
3F93	FF		3F9B	FF	
3F94	FF		3F9C	FF	
3F95	70		3F9D	0E	
3F96	30		3F9E	0C	
3F97	10		3F9F	08	

3FA0	00		3FA8	00	
3FA1	00		3FA9	00	
3FA2	18		3FAA	FF	
3FA3	3C		3FAB	FF	
3FA4	7E		3FAC	7E	
3FA5	FF		3FAD	3C	
3FA6	FF		3FAE	18	
3FA7	00		3FAF	00	
3FB0	80		3FB8	02	
3FB1	E0		3FB9	0E	
3FB2	F8		3FBA	3E	
3FB3	FE		3FBB	FE	
3FB4	F8		3FBC	3E	
3FB5	E0		3FBD	0E	
3FB6	80		3FBE	02	
3FB7	00		3FBF	00	
3FC0	38		3FC8	38	
3FC1	38		3FC9	38	
3FC2	92		3FCA	10	
3FC3	7C		3FCB	FE	
3FC4	10		3FCC	10	
3FC5	28		3FCD	28	
3FC6	28		3FCE	44	
3FC7	28		3FCF	82	
3FD0	38		3FD8	38	
3FD1	38		3FD9	38	
3FD2	12		3FDA	90	
3FD3	7C		3FDB	7C	
3FD4	90		3FDC	12	
3FD5	28		3FDD	28	
3FD6	24		3FDE	48	
3FD7	22		3FDF	88	
3FE0	00		3FE8	3C	
3FE1	3C		3FE9	FF	
3FE2	18		3FEA	FF	
3FE3	3C		3FEB	18	
3FE4	3C		3FEC	0C	
3FE5	3C		3FED	18	
3FE6	18		3FEE	30	
3FE7	00		3FEF	18	
3FF0	18		3FF8	00	
3FF1	3C		3FF9	24	
3FF2	7E		3FFA	66	
3FF3	18		3FFB	FF	
3FF4	18		3FFC	66	
3FF5	7E		3FFD	24	
3FF6	3C		3FFE	00	
3FF7	18		3FFF	00	

3. BASIC

3.1. BASIC-FORTOLKEREN I CPC 664 & CPC 6128

CPC indeholder en hurtig og komfortabel BASIC-fortolker, der har til huse i en 16 KBytes ROM. Den er fælles for begge computere. Den belægger adresseområdet fra &C000 til &FFFF parallelt med skærm-RAM. Til lagring af BASIC-programmer og variabler ligger et 42249 bytes stort område fra &0170 til &A67B.

Fortolkeren understøtter alle muligheder, som tillades fra hardware og operativsystemets side. Herunder hører frem for alt skærmoutput med op til 8 vinduer, højopløselig grafik, lys og EVENT-behandling. Hermed er det for en gangs skyld muligt at udføre flere jobs samtidigt fra BASIC (multitasking). BASIC-fortolkeren understøtter endvidere heltals-aritmetik med op til 16 positioner (værdiområde -32768 til 32767) og en floating point-aritmetik med 8-bits-toereksponent og 32-bits-mantisse, der sikrer en nøjagtighed på 9 decimaler i et værdiområde på +/- 1E-39 til +/- 1E+38.

Floating Point-aritmetikken er ikke en del af BASIC-fortolkeren, men er indeholdt i operativsystemets ROM (adresse &2F73 til &37FF). Den kaldes, som de øvrige funktioner i operativsystemet via hoptabellen i øvre RAM-område (&BB00 til &BDF4), der kan ændres, hvis det er påkrævet.

BASIC-fortolkeren gør det muligt at udføre komfortabel kodning, editering og udførelse af programmer. Førstnævnte udgøres f.eks. af AUTO-kommandoen, editeringen af EDIT-kommandoen, der grundet operativsystemet ikke lader noget tilbage i forhold til en fuldskærms-editor, såvel som kommandoerne RENUM, MERGE og DELETE. Selv ved udførelse af programmer, mangler der ikke komfort. Som eksempel kan nævnes fejlbehandlingen med ON ERROR GOTO, typedefinitionen af variabler med DEF-type, den selektive sletning af felter med ERASE, input/output af tal som decimal-, binær- eller hextal, egendefinerede funktioner med flere argumenter, og alle datatyper og programstrukturer som IF...THEN...ELSE, FOR..NEXT og WHILE...WEND. Belægning af taster og funktionstaster er ligeledes en mulighed, samt definering af specialtegn på skærmen. Der mangler heller ikke hverken TRACE-kommando eller den omfattende PRINT USING.

INDTASTNING AF BASIC-LINIER.

Når man indtaster en BASIC-linie, så overtages den først af en 256 bytes stor buffer, der ligger fra &ACA8 til &ADA7. Der står det indtastede i almindelig tekst. Starter linien med et linienummer, konverteres det til et 16-bits-birnærtal, der lægges i en anden buffer for den konverterede linie. Denne buffer er 300 tegn stor, og ligger før BASIC-programmet i området fra &40 til &16F. Derefter gennemgås linien for reserverede BASIC-ord. Fundne ord erstattes med TOKENS. Ordet "AFTER" erstattes eksempelvis med token &80. Alle kommandoer og BASIC-operatorer, såsom "=" og "AND" har værdier større end 127. Den 7. bit er således sat. BASIC-funktioner såsom EXP eller ROUND har tokens mellem 0 og &7F. For at kunne skelne dem fra normale ASCII-tegn, identificeres de via et foranstillet &FF. Kolon'et, som adskiller to statements udgøres af koden &01, og en linies afslutning udgøres af værdien &00. Kan en bogstavfølge ikke indetificeres som værende en kommando eller funktion, betegnes

den som værende et variabelnavn. Et variabelnavn kan bestå af op til 40 betydende tegn. Der skelnes ikke imellem store- og små bogstaver. Lad os antage, at vi har indtastet følgende linie:

```
start = 77
```

Efter linienummeret lægges følgende:

```
&0D &00 &00 &73 &74 &61 &72 &F4 &EF &19 &4D &00
```

Her betyder &0D, at det er en variabel uden typekendetegn. Derefter følger to nullbytes, som vi senere kommer ind på. Så følger navnet på variablen, ASCII-koderne for s,t,a og r. Ved det sidste tegn "t" adderes &80 til ASCII-værdien &74 (den øverste bit sættes) og vi får &F4. Koden &EF er token for "=". Det efterfølgende tal &19 angiver en 1-byte-konstant: &4D er lig 77. Det sidste &00 angiver liniens afslutning.

Inden linienummeret følger endu to bytes, der angiver liniens længde:

```
&12 &00 &0A &00
```

Linien er altså &12 + 256 * &00 = 18 bytes lang og har linienummer &0A + 256 * &00 = 10.

Det fremgår heraf, at fortolkeren, i forhold til andre BASIC-interpretere, ikke lagrer programkoden i ASCII-format, men derimod i binærform. Fordelen er indlysende. Koden oversættes til binærform een gang for alle, og ikke for hvert gennemløb af linien. Man opnår en fantastisk hastighedsgevinst ved udførelse af programmet.

CPC kender en række numeriske konstanter, der er kendetegnet via tilsvarende tokens. Konstanter, der kun består af et ciffer, d.v.s. tallene fra 0 til 9, kodes af tokens &0E til &17, og optager således kun 1 byte i programteksten. Token &19 for 1-byte-værdier har vi allerede stiftet bekendtskab med. For 2-bytes-integerværdier findes 3 forskellige tokens, alt efter om konstanten er indtastet i decimal-, binær- eller hexform. Belægningen i Lo- og Hi-byte er ens i alle tre tilfælde.

```
&1A To-bytes-værdi, decimal.
&1B To-bytes-værdi, binær.
&1C To-bytes-værdi, hex.
```

Drejer det sig om et helt tal eller et tal højere end 32767, lægges det som floating point-værdi, der kendetegnes med token &1F. Derefter følger 5 bytes for flydende komma. Floating-Point-tal kommer vi ind på andetsteds.

En særstilling indtager linienumre, der efterfølger kommandoer såsom GOTO, GOSUB eller RUN. De lægges ligeledes som 16-bits binære tal, dog med token &1E som kendetegn.

Udføres et program og der mødes en GOTO-kommando, så læses linienummeret, hvorefter hele programmet gennemsøges for dette nummer. Det kan især ved lange programmer vare en stund. Ofte anvendes GOTO- og GOSUB-kommandoer i programløkker, hvor de udføres hundreder eller tusinder af gange. Her kan søgetiden optage hovedparten af programudførelsestiden. BASIC-fortolkeren i CPC udfører kun linie-

søgningen en enkelt gang. Er linien først fundet, så erstattes linienummeret efter GOTO-kommandoen med adressen på den linie, der blev fundet. For at adressen kan skelnes fra et linienummer, skiftes token &1E ud med &1D. Næste gang GOTO-kommandoen udføres, kan fortolkeren gå direkte til liniens adresse, hvilket er utroligt tidsbesparende.

PROGRAMUDFØRELSE VIA BASIC-FORTOLKEREN.

Udførelse af statements via BASIC-fortolkeren kan forklares på følgende vis. Hver programlinie begynder med programlængden og linienummeret. Herefter kommer den egentlige BASIC-kommando. Fortolkeren tester nu, om det er en kommando-token, der har værdien mellem &80 og &E1. Er det tilfældet, benyttes denne token som pointer i en tabel, som indeholder samtlige BASIC-kommandoer. BASIC-kommandoen udføres som underprogram. Herefter returneres til den såkaldte fortolkerløkke. Startede instruktionen ikke med en kommando-token, så fortsættes der ved LET-kommandoen.

Den vigtigste del af en BASIC-fortolker, er vel nok udtryksberegning. CPC skelner mellem tre typer af udtryk: Integer, Flydende komma og String. Hvis, f.eks. en værditilegnelse til en variabel skal udføres, eller der skal beregnes et parameter til en kommando, så kaldes en rutine, der beregner udtrykket og klargør værdien og typen. Variabeltypen kan antage tre værdier:

2 integer
3 string
5 flydende komma

Disse typenumre er samtidigt udtryk for variablernes længder. Ved en streng er det den såkaldte descriptor, som indeholder længde og adresse (se også kapitel om variabelpointere). Stemmer et udtryks type ikke oversens med en variabel, så forsøges en typekonvertering. Det er dog kun muligt, når det drejer sig om heltal og flydende komma tal. Konverteringen koster selvfølgelig regnetid. Derfor bør man, hvor det er muligt, indsætte heltalsvariabler. Det har i praksis vist sig, at man i 90% af tilfældene kan anvende integervariabler. Herved bortfalder ikke kun typekonverteringen, men integeraritmetik er tillige væsentligt hurtigere end flydende komma-aritmetik. Man kan især måle det, ved variabler anvendt i FOR-NEXT-løkker eller tilsvarende tællere.

En konvertering fra string til numerisk værdi eller modsat, er kun muligt med funktionerne VAL og STR$. Alle andre forsøg resulterer i fejlmeddelelsen "TYPE MISMATCH".

3.2. BASIC STACK

En stack tjener til midlertidig lagring af data efter "Last in - First out"-princippet. Processoren benytter hertil adresserummet fra &C000. For hvert element nedtælles (DECREMENTERES) stackpointeren. Hentes data igen fra stack, forhøjes stackpointeren tilsvarende (INKREMENTERES). Processorstack bruges f.eks. til lagring af returhopadresser ved kald af underprogrammer.

BASIC-fortolkeren har tilsvarende brug for at kunne lagre parametre fra GOSUB-kald, FOR-NEXT- og WHILE—WEND-løkker. Her anvendes ikke længere processor-stack, men derimod en særlig BASIC-stack, der er 512 bytes stor og starter ved adresse &AE8B. *I modsætning til processor-stack, vokser denne stack opefter,* ved lagring af elementer, indtil den maksimale adresse &B08A. Som stackpointer anvendes adresserne &B08B/&B08C.

Lad os først kigge på, hvilke parametre, der skal lægges i stack ved en GOSUB-kommando.

&00/&01	Kendetegn for GOSUB
Lo/Hi	Adresse på instruktionen efter GOSUB-kommandoen
Lo/Hi	Linieadresse for GOSUB-instruktionen
&06	Stackelementets størrelse

Først lægges der en byte ind, der angiver typen. Ved en normal GOSUB-kommando er denne en 0-byte. Drejer det sig om kald af et underprogram, der hører ind under AFTER/EVERY-kommandoerne, lægges der et 1-tal ud på stack. Derefter følger adressen for næste kommando efter GOSUB-kommandoen, samt adressen på linien, hvori kommandoen står opført. For at stack-elementet ved RETURN-GOSUB-kommandoen, kan identificeres igen, lægges der endnu en byte i stack, der angiver længden på elementet og således identificerer hele GOSUB-posten.

Data i en WHILE-END-løkke lægges i en lignende form.

LO/Hi	Linieadresse for WHILE-kommandoen.
Lo/Hi	Adresse for WEND-kommandoen.
Lo/Hi	Adresse for WHILE-betingelsen.
&07	Stackelementets størrelse.

Posten indeholder altså 3 adresser, og er kendetegnet med byten 7, antallet af elementer.

Ved FOR-NEXT-løkken bliver det lidt mere kompliceret. Her skelnes mellem løbevariabler af typerne integer og real. I det første tilfælde, er ikke kun udførelsestiden kortere, men den krævede plads i stack er ligeledes tilsvarende mindre. Lad os først kigge på opbygningen af en integer-løkke.

Lo/Hi	Adresse for Løbevariablen (den, der ændres).
Lo/Hi	Slutværdi for løbe-variablen.
Lo/Hi	STEP-værdi.
Sgn	Fortegn for STEP-værdi.
Lo/Hi	Adresse på FOR-instruktionen.
Lo/Hi	Linieadresse på FOR-instruktionen.

Lo/Hi	Adresse på NEXT-instruktionen.
Lo/Hi	Linieadresse for NEXT-instruktionen.
&10	Stack-postens størrelse.

Stack-indholdet for en FOR-NEXT-løkke med integervariabel, er altså 16 bytes stor. Gennemkøres løkken med en real-variabel, skal der lægges 22 bytes i stack.

Lo/Hi	Adresse på løbe-variablen.
5 bytes flydende komma værdi	Slutværdi for løbe-variablen.
5 bytes flydende komma værdi	STEP-værdi.
Sgn	Fortegn for STEP-værdien.
Lo/Hi	Adresse på FOR-instruktionen.
Lo/Hi	Linieadresse for FOR-instruktionen.
Lo/Hi	Adresse på NEXT-instruktionen.
Lo/Hi	Linieadresse for NEXT-instruktionen.
&16	Stackpostens størrelse.

Foruden til brug for lagring af løkkestrukturer, anvendes BASIC-stack også til lagring af mellemværdier ved numeriske beregninger, f.eks. ved komplexe udtryk i paranteser og realisering af hierakiet ved aritmetiske og logiske operatorer.

3.3. BASIC OG MASKINSPROG

3.3.1. CALL-KOMMANDOEN

Bindeleddet imellem BASIC og maskinsprog udgøres af CALL-kommandoen. Ved hjælp af den, kan man kalde et maksinkodeprogram fra BASIC. Sammen med CALL-kommandoen hører der en 16 bits-adresse, der fortæller, hvor maskinkodeprogrammet starter, f.eks.:

```
CALL &8000
```

Her kaldes et maskinkodeprogram i adresse &8000 eller decimalt 32768. Afsluttes programmet med RET-kommandoen, overgives kontrollen igen til fortolkeren, der kan fortsætte udførelsen af det BASIC-program, hvorfra rutinen blev kaldt.

Ved CALL-kommandoen er operativsystemet og BASIC ikke direkte tilgængelige. Man kan selvfølgelig kalde systemrutiner via tilgangsadresser i området fra &B000. Disse rutiner sørger selv for den påkrævede ROM/RAM-konfiguration. Vil man under CALL-kommandoen kalde BASIC-fortolkeren eller rutiner i operativsystemet, der ikke kan kaldes via vektorer, kan man benytte RST 3 og RST 5 rutinerne, der udfører en omskiftning.

CALL-kommandoen tillader også, at der overgives parametre fra BASIC til rutinen. Her kan man efter adresseangivelsen overgive op til 32 parametre adskilt med komma. Parametrene skal, som adressen, angives i 16-bits-værdier. De lægges fra BASIC i stack. BASIC-fortolkeren overfører basisadressen for parameterblokken i IX-register. I akk er der anført, hvor mange parametre, der er overført. Det sidste parameter står således i adresse IX, det næstsidste i IX+2 og det første parameter står i adresse IX+2*(A-1).

Under CALL-kommandoen kan samtlige registerindhold ændres (kræver udnyttelse af andet register, se kapitlet om firmware). Selv stackpointeren kan ændres, når man husker at notere den korrekte returadresse fra stack.

Ved benyttelse af CALL-kommandoen er det kun ens egen fantasi, der sætter grænserne. Man kan f.eks. opbygge udvidet grafik og lignende.

Overførselen af parametre fra maskinkode tilbage til BASIC er ikke implementeret, men er dog muligt via en omvej. Skal f.eks. resultatet af en maskinkodeberegning flyttes til en variabel, så kan man give CALL-kommandoen dens adresse. Det er muligt med brug af alfakrøllen "@".

```
CALL &AB00,@A
```

Dermed står adressen på variablen A til rådighed. Dette forhold er beskrevet nærmere i kapitlet om variabelpointere.

3.3.2. BASIC-UDVIDELSER MED RSX

Operativsystemet og BASIC i CPC understøtter muligheden for at implementere egne kommandoer i BASIC; dette kaldes RSX, som er en forkortelse for Resident System eXtension. Udvidelserne kan kaldes fra BASIC via et navn og tillader parameteroverførsel, som allerede er bskrevet ved CALL-kommandoen. Hvis vi f.eks. ønsker at skrive en grafikudvidelse, der tegner et kvadrat på skærmen, kan kaldet se således ud: er beskrevet nærmere i kapitlet om variabelpointere.

```
IQUADRAT,100,100,50
```

Herved tegnes et kvadrat, hvor øverste venstre hjørne ligger på koordinaterne 100,100. Siden er 50 punkter.

Som det ses, kendetegnes en kommandoudvidelse med en foranstillet lodret streg (SHIFT + @).

En sådan udvidelse kan ligge i Extension-ROM, hvilket er tilfældet, hvis man har tilsluttet et diskettedrev eller arbejder i RAM. Nu kan vi skrive vore egne kommandoer eller udvidelser. For at operativsystemet ved, hvor det skal søge efter en udvidelse, skal den først implementeres. Her kan en rutine fra operativsystemet hjælpe: KL LOG EXT. Det følgende eksempel realiserer det ovennævnte eksempel med tegning af et kvadrat og demonstrerer selve implementeringen.

```
                ; RSX-KOMMANDOUDVIDELSE

                ; LE 15/6/85

BCD1            LOGEXT  EQU     &BCD1       ; implementer udvidelsen
BBC6            ASKCUR  EQU     &BBC6       ; hent grafik-cursoren
BBC0            MOVABS  EQU     &BBC0       ; saet grafik-cursoren
BBF9            DRAWRE  EQU     &BBF9       ; tegn linie relativt
BDC7            CHGSGN  EQU     &BDC7       ; byt fortegn

8000                            ORG     &8000

8000  010980                    LD      BC,RSX      ; adresse paa RSX-komman-
                                                    dotabel
8003  211680                    LD      HL,KERNAL   ; 4 bytes RAM for kernal
8006  C3D1BC                    JP      LOGEXT      ; implementer udvidelsen
8009  0E80      RSX           DEFW      TABLE       ; adresse for kommandoord
800B  C31A80                    JP      QUADRAT

800E  51554144  TABLE         DEFM      "QUADRA"
8014  D4                      DEFB      "T"+&80
8015  00                      DEFB      0           ; tabellens slutning
8016            KERNAL        DEFS      4           ; hukommelse for kernal
801A  FE03      QUADRA        CP        3           ; tre parametre?
801C  C0                      RET       NZ
801D  CDC6BB                  CALL      ASKCURS     ; hent grafikcursor
8020  D5                      PUSH      DE          ; gem x-koordinat
8021  E5                      PUSH      HL          ; gem y-koordinat
8022  DD5605                  LD        D,(IX+5)
8025  DD5E04                  LD        E,(IX+4)    ; x-koordinat
8028  DD6603                  LD        H,(IX+3)
802B  DD6E02                  LD        L,(IX+2)    ; y-koordinat
802E  CDC0BB                  CALL      MOVABS      ; grafikcursor til x,y
8031  DD5601                  LD        D,(IX+1)
8034  DD5E00                  LD        E,(IX)      ; flyt laengde til DE som offset
8037  D5                      PUSH      DE          ; gem
8038  210000                  LD        HL,0        ; y-offset
803B  CDF9BB                  CALL      DRAWREL     ; tegn vandret linie
803E  E1                      POP       HL
803F  E5                      PUSH      HL
8040  CDC7BD                  CALL      CHGSGN      ; y-offset negativ
8043  E5                      PUSH      HL
8044  110000                  LD        DE,0
8047  CDF9BB                  CALL      DRAWREL     ; tegn lodret linie
804A  D1                      POP       DE          ; negativ x-offset
804B  210000                  LD        HL,0        ; y-offset nul
804E  CDF9BB                  CALL      DRAWREL     ; tegn vandret linie
8051  E1                      POP       HL
8052  110000                  LD        DE,0
```

```
8055  CDF9BB        CALL  DRAWREL   ; tegn lodret linie
8058  E1            POP   HL
8059  D1            POP   DE
805A  C3C0BB        JP    MOVABS    ; reetabler koordinater
```

Efter programmet er loaded (som binærfil fra diskette) eller ved hjælp af en DATA-loader står genereret i hukommelsen, skal det initialiseres. Det sker ved kald af CALL &8000, og den nye kommando er til fri brug. Til implementeringen benyttes to tabeller. Den første, i vort tilfælde kaldet RSX, indeholder først adressen på den anden tabel, her kaldet TABLE, og endelig hopkommandoer til den egentlige udvidelse. Den anden tabel indeholder navnene, hvorunder de nye kommandoer kan kaldes. Store bogstaver og punktummer er tilladte. Sidste tegn i et kommando-ord er kendetegnet med en sat bit 7. Herefter kan der følge flere kommandoord. Tabellens slutning markeres med en 0-byte. I hver tabel, skal der naturligvis stå opført det samme antal elementer; for hvert kommandoord, skal der i den første stå anført den tilhørende hopadresse. Under Label KERNAL skal operativsystemet have 4 bytes til rådighed, der anvendes til forvaltning af udvidelserne. De 4 bytes skal ligge i området mellem &4000 og &BFFF.

Rutinen til tegning af kvadratet begynder med label QUADRAT. Først testes for, om der blev overført 3 parametre. Er det ikke tilfældet, returneres øjeblikkeligt. Er de 3 parametre overført, hentes den aktuelle grafik-cursorposition og lægges i stack. Nu hentes de overførte X- og Y-koordinater til DE og HL. Parameterblokkens basis står i IX. Efter at grafik-cursoren er sat på sin position, kan rutinen til tegning af en linie relativt til den aktuelle cursorposition, kaldes fire gange. For at beregne et negativt offset, kaldes integeraritmetikkens rutine CHGSGN. Til sidst genetableres den oprindelige grafik-cursorposition.

Som eksempel på anvendelsen af rutinen, kan man indtaste det følgende korte program:

```
10 CLS
20 FOR I=35 TO 400 STEP 20
30 IQUADRAT,I,I,30
40 NEXT
```

3.3.3. VARIABELPOINTEREN "@"

En for maskinkodeprogrammører særlig interessant funktion, er variabelpointeren, der kan kaldes med "@". Funktionen giver adressen, hvor en variabel befinder sig i hukommelsen. Kaldet ser således ud:

```
PRINT @a
```

Hvis det er indholdet i variabelen, vi ønsker at kende, skal vi skelne mellem 3 mulige typer.

Ved integervariabler er proceduren nemmest at udføre. På den angivne adresse er 16-bit-værdien gemt. Værdien i variablen a% findes efter følgende formel:

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

Her kan vi få værdier mellem 0 og 65535. Skal der tages hensyn til fortegn, skal funktionen UNT anvendes.

```
PRINT UNT(PEEK(@a%)+256*PEEK(@a%+1))
```

Når det gælder flydende komma-variabler, peger variabelpointeren ligeledes på variablens værdi, der dog er sammensat af 5 bytes. De første 4 bytes er den såkaldte mantisse, og den 5. byte er toereksponenten, hvormed mantissen skal multipliceres, for at danne variablens værdi. Hvis vi betegner de 4 mantissebytes med m1 til m4 og eksponenten med ex, så vil følgende formel give os den tilhørende flydende komma-værdi:

```
X = (1-2*SGN(m4 AND 128))*2↑(ex-129)*(1+((m4 AND 127)+
    (m3+(m2+m1/256)/256)/256)/128)
```

Formlen viser tydeligt, at tallets fortegn ligger i m4's øverste bit og at mantissebytes m1 til m4 har stigende værdier. Toereksponenten indeholder en offset på 129, så der kan gives værdier mellem 2↑-129 og 2↑127. Lad os afprøve formlen:

```
100 a=-13: "undersøgte flydende komma-variabel'
110 ad = @a: " a-adresse
120 m1 = PEEK(ad):m2 = PEEK (ad+1):m3=PEEK(ad+2)
130 m4 = PEEK(ad+3):ex =PEEK(ad+4)
140 PRINT (1-2*SGN(m4 AND 128))*2(ex-129)*(1+((m4 AND 127)+
    (m3+(m2+m1/256)/256)/256)/128)
```

Hvis programmet startes op, skrives værdien -13 på skærmen. Prøv at erstatte linie 100 med INPUT a, således at man kan afprøve forskellige værdier.

Variabelpointer-funktionen finder anvendelse i CALL-kommandoen, der som bekendt kun kan overføre 16-bits-værdier. Vil man arbejde med flydende komma-tal, kan man med "@" overføre adressen på et sådant tal.

Det bliver endnu mere interessant med stringvariabler. Også her kan vi benytte variabelpointer, der meddeler os variablens adresse. Det er dog ikke direkte strengens adresse, men derimod den såkaldte stringdescriptor. Den er 3 bytes lang. Den første byte indeholder længden på strengen, altså en værdi mellem 0 og 255. De to næste bytes indeholder strengens adresse.

```
100 INPUT a$
110 ad=@a$
120 I=PEEK(ad)
130 sa = PEEK(ad+1)+256*PEEK(ad+2)
140 FOR I=sa TO sa+I+1: PRINT CHR$(PEEK(I));:NEXT
```

Programmet henter længde og adresse for strengen, læser og udskriver værdien.

Også her kan man via variabelpointeren overføre en streng til CALL-kommandoen. Strenge kan sammen med CALL-kommandoen indsættes på helt anden vis. Man lægger et maskinkodeprogram ind i en streng, hvor det kan kaldes med CALL og variabelpointeren. Maskinkodeprogrammet skal være forskydeligt, og ikke være over 255 bytes længde. Det er ofte nok til mindre utilities. Vil man gøre brug af metoden, skal følgende gøres:

Først lægges maskinkodeprogrammet ind i strengvariablen. Det sker for det meste med READ og DATA. Vil man herefter udføre programmet, så beregnes strengens startadresse (og samtidigt maskinkodeprogrammets) med "@".

3.4. BASIC ROM

3.4.1. FLYDENDE KOMMA ARITMETIK

Samtlige aritmetiske funktioner, der benyttes af BASIC-fortolkeren, står i operativsystemets ROM. De kaldes via hoptabellen fra &BD5E til &BDBB. Hvis man vil ændre på rutinerne, behøver man kun at indføje et hop til den egendefinerede rutine.

Som eksempel på anvendelse af rutinerne i BASIC-fortolkeren, viser vi en rutine til beregning af kvadratroden af et tal. *Selvom funktionen allerede er indbygget i computeren, så kan dens styrke forbedres.*

Den indbyggede SQR-funktion arbejder efter samme algoritme, der beregner potenser.

SQR(X) = EXP (LOG(X)*0.5)

Der skal altså altid beregnes eksponential- og logaritmefunktion, hvilket foregår med polynomberegning. Kvadratroden lader sig dog også beregne via en simpel iteration:

X(N+1) = (X(N)+A/X(N))/2

Hvor A er tallet, hvorfra roden skal findes og X(N) er start- og X(N+1) er den nye tilnærmelsesværdi. Som startværdi kan tallet A bruges, men en bedre startværdi fås, hvis man halverer tallets toereksponent. Så ændres tallet efter 4 iterationer ikke længere. Tidsgevinsten er betydelig. Hvis CPC's egen SQR-rutine skal bruge 27 millisekunder, kan vores rutine nøjes med knap 8 millisekunder og er altså mere end 3 gange så hurtig.

```
                ; HURTIG SQR-RUTINE
                ; LE 10/6/86

A000                    ORG     &A000
BD91            SGN     EQU     &BD91
BD85            DIV     EQU     &BD85
BD79            ADD     EQU     &BD79
```

```
A000  CD91BD NEWSQR CALL  SGN          ; test fortegn
A003  3F            CCF
A004  C8            RET   Z          ; 0 allerede faerdig
A005  F20CA0        JP    P,GOON
A008  3E01          LD    A,1        ; Improper argument
A00A  B7            OR    A
A00B  C9            RET


A00C  E5            GOON  PUSH       HL
A00D  1153A0        LD    DE,STORE1
A010  010500        LD    BC,5
A013  EDB0          LDIR  ; noter radikant
A015  E1            POP   HL


A016  E5            PUSH  HL
A017  DDE1          POP   IX
A019  DD7E04        LD    A,(IX+4)   ; exponent
A01C  D681          SUB   &81        ; normaliser
A01E  3F            CCF
A01F  1F            RRA              ; halver exponent
A020  C601          ADD   A,1
A022  DD7704        LD    (ix+4),A   ; som startvaerdi
A025  0604          LD    B,4        ; 4 iterationer
A027  C5      ITER  PUSH  BC
A028  E5            PUSH  HL
A029  1158A0        LD    DE,STORE2
A02C  010500        LD    BC,5
A02F  EDB0          LDIR             ; tilnærmelsesvise værdi
A031  E1            POP   HL
A032  E5            PUSH  HL
A033  1153A0        LD    DE,STORE1
A036  EB            EX    DE,HL
A037  010500        LD    BC,5
A03A  EDB0          LDIR             ; hent radikanten
A03C  E1            POP   HL
A03D  1158A0        LD    DE,STORE2
A040  CD85BD        CALL  DIV
A043  1158A0        LD    DE,STORE2
A046  CD79BD        CALL  ADD
A049  E5            PUSH  HL
A04A  DDE1          POP   IX
A04C  DD3504        DEC   (IX+4)     ; tal / 2
A04F  C1            POP   BC
A050  10D5          DJNZ  ITER
A052  C9            RET
A053        STORE1  DEFS  5
A058        STORE2  DEFS  5
```

Hvordan får man så fortolkeren til at anvende den nye rutine? SQR-funktionen styres over vektoren &BD9D. På denne adresse skal udføres endnu et hop til vores egen rutine.

```
JP &A000
```

Hvis rutinen kaldes fra BASIC, skal HL-registeret pege på flydende komma-værdien. Efter udførelse af rutinen, skal HL-registeret pege på resultatet. Normalt har værdien i dette register ikke ændret sig. De følgende flag viser funktionens errror-status:

Besidder de en CPC 6128, så skal adresserne for rutinerne SGN, DIV og ADD forhøjes med 3; ligeledes skal hoppet til &A000 ændres til &BDA0.

Fejlstatus:

C=1	Korrekt udførelse.
C=0 & Z=1	DIVISION BY ZERO
C=0 & N=1	OVERFLOW
C=0 & Z=0	IMPROPER ARGUMENT

På de næste sider findes listningen af flydendekomma-aritmetikken, hvor hver rutine også indeholder adressen på hoptabellen, hvor BASIC-fortolkeren kan kommunikeres. Integeraritmetikken befinder sig i BASIC-ROM fra adresse DD2F til DE19.

*******	BD97	PI
2F73		
2F78	PI	
******	BD5E	kopiere variabler
2F91		
******	BD64	4-byte-værdien efter følgende komma
2FC8		
******	BDB5	4-byte-værdien x 256 efter integer
2FD1		
******	BD67	flydende komma efter integer
2FD9		
******	BD6A	flydende komma efter integer
3001		
******	BD6D	FIX
3014		
******	BD70	INT
3055		
******	BD73	
305F		
******	BD76	Tal x 10↑A
30C6		
******	BDB8	RND INIT
3136		
******	BDBB	SET RANDOM SEED
3143		
******	BD7C	RND
3159		
******	BD88	hente sidste RND-værdi
3188		
******	BDA3	LOG10
31B1		
******	BDA0	LOG
31B6		
******	BDA6	EXP
322F		
******	BD9A	SQR
32AC		
******	BD9D	potentiering
32AF		
******	BD94	DEG/RAD
3345		
******	BDAA	COS
3349		
******	BDA7	SIN
3353		
******	BDAF	TAN

33C8
****** BDB2 ATN
33D8
****** BD7F Subtraktion
349E
****** BD79 Addition
34A2
****** BD82 Multiplikation
3577
****** BD85 Division
3604
****** BD8B Sammenligning
35DF
****** BD91 SGN
3727
****** BD8E Fortegns skiftning
3731

****** CPC 664 & 6128 BASIC 1.1
C000 første forgrunds-ROM
C001 Mark 1
C002 Version 1
C003 Modifikation 0
C004 navnet på adressen
****** BASIC-Initialisering
C006 stak fra C000
C009 KL ROM WALK
C00C lagerkonfiguration
C00F for lidt lager, så reset
C013 slette flag for blanks undertrykkelse
C016 Cursor på BASIC 1.1
C019 udlæs tekst
C01C aktuel linieadresse på nul
C01F slette fejlnummer
C022 RND-Init
C025 AUTO-Mode slettes
C028 NEW-kommando
C02B 240
C02E SYMBOL AFTER 240
C031 til READY-Mode
C033 'BASIC 1.1',LF,LF,0
C040 'BASI', 'C'+80H,0
****** BASIC-kommando EDIT
C046 hente linienummer efter DE
C04A Initialisere stak
C04D BASIC-linie DE søges
C050 BASIC-linie i pufferlisten
C053 Hente indgangslinie

****** READY-mode
C058 Initialisere stak
C05B diverse initialiseringer
C05E hente linieadresse
C061 SOUND HOLD
C064 Break-Event slettes
C067 billedskærm initialisering
C06A beskyttet program
C06E ja, slette program og variable
C071 ERROR-nummer
C074 'Syntax error'?
C076 nej
C078 ERROR-nummer på nul
C07B hente nummer på ERROR-linie
C07F til EDIT-kommando
C081 Cursor på 'Ready'
C084 udgave
C087 aktuel linieadresse på nul
C08A AUTO-Flag sat
C08E nej
C090 brug næste linienummer
C093 til READY-mode
C095 gennemlæs Blank, TAB og LF
C09D gennemlæs Blank, TAB og LF
C0AF hente indgangslinie
C0B2 'ESC' trykket, så gentag
C0B4 LF udgave
C0B7 gennemlæs Blank, TAB og LF
C0D4 til interpretersløjfe
C0D7 'Ready',LF,0
****** AUTO-Mode slettes
C0DF
****** AUTO-Mode slettes
C0E1 linienummer
C0E6 sæt Flag for AUTO
****** BASIC-kommando AUTO
C0EA 10, Default
C0EF ','
C0F1 hente linienummer efter DE
C0F5 10, Default
C0F8 følger komma?
C0FB ja, hente linienummer efter DE
C0FE linie slut, ellers 'Syntax error'
C102 AUTO-inkrement mærkes
C106 Flag for AUTO-mode mærkes
C10D linienummer
C115 AUTO-mode slettes

C118 editere linie
C11E linienummer
C121 plus inkrement
C122 AUTO-MODE sættes
****** BASIC-kommando NEW
C128
C129 slet program og variabler
C12C til READY-mode
****** BASIC-kommando CLEAR
C12F 'INPUT'
C13A diverse initialiseringer
****** CLEAR INPUT
C13F Gennemlæs Blank
****** program og variabler slettes
C145 Begyndelsen af de frie RAMs
C149 HIMEM
C152 Akku slettes
C154 Slet fri RAM til HIMEM
C156 flag for sletning af program
C159 tilbagesæt variablepointer
C15F brække Disk I/O
C163 sæt RAD-Mode
C166 initialisere Descriptorstak
C16C Stream-Reset
C16F TROFF
C172 slet AUTO-Mode
C175 diverse initialiseringer
C17A slet String
C17D tilbagesæt variablepointer
C180 alle variabler af typen 'Real'
C183 Blank, TAB og LF gennemlæses
****** diverse initialiseringer
C189 initialisere Tabulator-stop
C18C slet programpointer
C18F slet ON ERROR
C192 slet programpointer efter afbrydelsen
C195 SOUND og Event-Reset
C198 initialisere BASIC-Stak
C19B flag for sletning af FN
C19E RESTORE
C1AB < 8?
C1AD TXT STR SELRCT
C1B1 aktuel Streamnummer
C1B7 indleveringskanal
C1C1 aktuel Streamnummer
C1C4 Printer ?
C1C7 indleveringskanal
C1CA Diskette ?
C1CD på Streamnummer test
C1D2 på Streamnummer test
C1D7 på Streamnummer test
****** hente Streamnummer
C1E8 på Streamnummer test
C1ED 'Inproper argument'
C1F5 ','
C1F7 Spring efter (BC), udførselsfunktionen
****** på Streamnummer test
C1FF '#'
C201 0 som Default
C204 hente Streamnummer
C208 følgende komma
C20B nej, så slutter Statementet
****** hente Streamnummer
C210 test følgende tegn
C213 '#'
C214 10, Maximalværdi +1
C218 Maximalværdi efter B
C219 hente 8-bits-værdi
C21C sammenlign med maximalværdi
C21F mindre, ok
C220 'Improper argument'
****** hente 8-bits-værdi mindre end 2
C223 Maximalværdi 2
C225 hente og teste argument
****** BASIC-kommando PEN
C227 hente Streamnummer
C22A TXT SET PEN
C230 følgende komma
C234 8-bits-værdi mindre end 2
C237 TXT SET BACK
****** BASIC-kommando PAPER
C23C hente Streamnummer
C23F TXT SET PAPER
C242 hente argument < 16
C246 Spring efter (BC), udførselsfunktion
****** BASIC-kommando BORDER
C24B hent 2 argumenter mindre end 32
C24F SCR SET BORDER
****** BASIC-kommando INK
C254 Argument mindre end 16
C258 test på ','

```
C25B   hent 2 argumenter < 32
C260   SCR SET INK
******  hent 2 argumenter < 32
C265   hent argument < 32
C268   efter B
C269   følgende komma
C26D   32
C26F   hent argument < 32
C272   efter C
******  hent argument < 16
C274   16
C276   hent argument < 16
******  BASIC-kommando MODE
C278   3
C27A   hent argument < 3
C27E   SCR SET MODE
******  BASIC-kommando CLS
C283   hent Streamnummer
C287   TXT CLEAR WINDOW
C28C   hent Streamnummer
C291   'Improper argument'
C294   Test på ')'
******  BASIC-kommando
       COPYCHR$
C29B   hent Streamnummer, klamme
       til
C29E   TXT RD CHAR
C2A1   overfør tegn til String
******  BASIC-funktionen VPOS
C2A4   hent Streamnummer
C2A8   hent cursorlinie
******  BASIC-funktionen POS
C2AD   hent Streamnummer
C2B0   test på ')'
C2B4   hent position
C2B7   overfør 'Akku' indholdet som
       heltal (Integer)
******  hent aktuel PRINT-position
******  hent cursorlinie
C2CA   TXT GET CURSOR
C2CD   TXT VALIDATE
C2DA   TXT GET WINDOW
******  BASIC-kommando LOCATE
C302   hent Streamnummer
C305   hent 2 8-bits-værdier ulig 0
C30C   TXT SET CURSOR
******  BASIC-kommando WINDOW
C311   'SWAP'
C315   hent Streamnummer
C318   hent 2 8-bits-værdier ulig 0
C31C   test på ','
C31F   hent 2 8-bits-værdier ulig 0
C326   TXT WIN ENABLE
******  WINDOW SWAP
C32B   gennemlæs Blank
C32E   hent argument < 8
C332   følgende komma ?
C335   Default 0
C337   ja, hent argument < 8
C33C   TXT SWAP STREAMS
******  hent agument < 8
C341   8, maximalværdi
C343   hent argument
******  BASIC-kommamdo TAG
C346   hent Streamnummer
******  BASIC-kommando TAGOFF
C34D   hent Streamnummer
C351   TXT SET GRAPHIK
******  hent 2 8-bits-værdier ulig 0
C354   hent første værdi
C357   efter D
C358   test på ','
C35C   hent 8-bits-værdier ulig 0
C360   værdi efter E
******  BASIC-kommando CURSOR
C363   hent Streamnummer
C366   0 ?
C368   hent 8-bits-værdier < 2
C36C   TXT CUR OFF
C35F   TXT CUR ON
C372   følgende komma ?
C375   nej
C376   hent 8-bits-værdier < 2
C37A   TXT CUR DISABLE
C37D   TXT CUR ENABLE
******  vis String
C380   Stringadresse
C381   132
C384   WIDTH på 132
C387   POS sat på En
C390   hent tegn
C391   forhøje pointer
C392   sidste tegn
C393   nej, vis
C396   næste tegn
******  vis LF
C39C   LF
C39E   vis
```

****** vis tegn
C3A5 vis tegn
C3AB LF
C3AD nej
C3AF enhedsadresse
C3B2 Printer ?
C3B5 Disk ?
C3B8 vis tegn
****** vis tegn
C3BE vis tegn
****** selekter udgangskanal
C3C4 udgangskanal
C3C9 udgang på Printer
C3CC på Disk
C3D0 på billedskærm
C3D4 TXT SET GRAPHIC
C3D9 TXT SET BACK
C3DD TXT VDU ENABLE
C3E0 TXT VALIDATE
C3E5 CR
C3EA LF
C3EC TXT OUTPUT
C3F1 TXT VALIDATE
****** VIS CR & LF PÅ Printer
C3F8 CR
C3FD LF
C40B MC PRINT CHAR
C40F test for afbrydelse med 'ESC'
C415 CR
C41A ' '
C434 CR
C439 LF
C449 DISK OUT CHAR
C44C fejlfri ?
C44D vis fejlmelding
C44F set DERR
C453 CAS TEST EOF
C45A overfør fortegn som heltal (integre)
C45F CAS IN CHAR
C462 fejlfri ?
C468 vis fejlmeldingen
C46B 'Diskettefejl'
****** POS sat på En
C472 KM READ CHAR
****** test for afbrydelse med 'ECS'
C475 KM READ CHAR
C479 'Break'
C47C vent ved andet tryk på tasten
C47F 'ESC', så afbrydelse
C485 Break-Event-Rutine's adresse
C488 BASIC-ROM Select
C48E KM ARM BREAK
****** Break-Event-Rutine
C495 KM READ CHAR
C498 ingen 'gentryk' på tasten ?
C49A Break ved hjælp af 'ESC'
C49C
C49E venter på andet 'ESC'
C4A1 test på ON BREAK GOSUB
****** venter på et tasttryk efter 'ESC'
C4A7 SOUND HOLD
C4B2 TXT CUR ON
C4B5 'ESC'
C4BB TXT CUR OFF
C4C3 ' '
C4C5 KM CHAR RETURN
C4CA SOUND CONTINUE
C4DC KM DISARM BREAK
****** BASIC-kommando ORIGIN
C4E1 hent 2 argumenter
C4E6 følgende komma ?
C4E9 nej
C4EB hent 2 argumenter
C4F0 test på ','
C4F3 hent 2 argumenter
C4F8 GRA WIN HEIGHT
C4FE GRA WIN WIDTH
C504 GRA SET ORIGON
C509 slutningen på Statements ?
C510 GRA CLEAR WINDOW
****** BASIC-kommando FILL
C515 hent argument < 16
C51A Garbage Collection
C51D fri plads tilbage
C520 mindst 29 Bytes
C523 sammenlign HL <> BC
C526 er 'Memory full'
C528 vis fejlmelding
C52D FILL
****** BASIC-kommando MOVE
C532 GRA MOVE ABSOLUTE
****** BASIC-kommando MOVER
C537 GRA MOVE RELATIVE
****** BASIC-kommando DRAW
C53C GRA LINE ABSOLUTE
****** BASIC-kommando DRAWR

C541 GRA LINE RELATIVE
****** BASIC-kommando PLOT
C546 GRA PLOT ABSOLUTE
***** BASIC-kommandp PLOTR
C54B GRA PLOT ABSOLUTE
C54F hent 2 heltalsargumenter
C552 følgende komma?
C555 nej
C557 ','
C55C følgende komma?
C55F nej
C563 hent 8-bits-værdier < 4
C567 SCR ACCESS
C56F Spring efter (BC)
****** BASIC-funktionen TEST
C574 GRA TEST ABSOLUTE
****** BASIC-funktioneb TESTER
C579 GRA TEST RELATIVE
C57D hent 2 argumenter
C580 test på ')'
C587 Spring efter (BC)
C58A overfør 'Akku' indholdet som heltal
****** hent 2 heltalsargumenter
C58F hent 16-bits-værdier -32768 til 32767
C593 test på ','
C596 hent 16-bits-værdier -32768 til 32767
C59A udfaldet af BC
****** BASIC-kpmmando GRAPHICS
C59D 'PAPER'
C5A1 test på efterfølgende tegn
C5A4 'PEN'
C5A5 ','
C5AA følgende komma ?
C5AE hent 8-bits-værdi < 2
****** GRAPHICS PAPER
C5B4 gennemlæs Blank
C5B7 hent argument < 16
C5BA GRA SET PAPER
****** GRAPHICS PEN
C5BD hent argument < 16
C5C0 GRA SET PEN
****** BASIC-kommando MASK
C5C3 ','
C5C7 hent 8-bits-argument
C5CD følgende komma ?
C5D1 hent 8-bits-værdier < 2
****** BASIC-kommando FOR
C5D7 læs variabler
C5DD søg tilhørende NEXT
C5E0 adressen mærkes
C5E6 søg åbne FOR-NEXT sløjfer
C5E9 fundet, sæt BASIC-stakpointer
C5ED slutning på Statement ?
C5F0 Default 0
C5F3 nej, hent variabel
C5FC Sammenlign HL <> DE
C5FF 'Unexpected NEXT'
C603 aktuel linieadresse efter HL
C607 sæt aktuel linieadresse
C610 22 Bytes, Type 5 'Real'
C616 16 Bytes, Type 2 'Heltal'(Integer)
C61A 'Type mismatch'
C61C vis fejlmelding
C61F antal Bytes efter A
C620 resaver plads i Basic-Stak
C624 variabeladresse på BASIC-Stak
C628 test på '='
C62B hent udtryk
C62F sammenlign variabeltyper
C633 mellemlager for FOR-variabel
C636 kopier variablen efter HL
C63A test på næste tegn
C63D 'TO'
C63E hent udtryk
C643 sammenlign variabeltype
C646 slutværdi på BASIC-Stak
C64C en som Default STEP-værdi
C64F overfør HL heltal
C653 næste tegn
C654 'STEP'
C656 nej
C658 gennemlæs Blank
C65B hent udtryk
C65F sammenlign variabeltyper
C663 kopier variablen efter (HL)
C666 hent fortegn
C66A fortegn fra STEP-værdi til BASIC-Stak
C66E slutning på Statements, er 'Syntax error'

C673 FOR-kommando's adresse på BASIC-Stak
C677 aktuel tegnadresse efter HL
C67C FOR's tegnadresse på BASIC-Stak
C681 adresse for NEXT-kommando på BASIC-Stak
C689 linieadresse for NEXT-kommando på BASIC-Stak
C68C #10 eller #16 for heltal/Real på Stak
C68E pointer på mellemlager
C691 hent FOR-variablen tilbage
C695 Flag for første gennemløb
C699 sæt aktuel linieadresse
C69F til NEXT-kommando
C6A1 vis fejlmelding
C6A4 'Unexpected NEXT'
****** BASIC-kommando NEXT
C6A5
C6A7 Flag for Inkrement adition
C6AB søg åben FOR-NEXT-sløjfe
C6B1 sæt BASIC-Stakpointer
C6B6 test sløjferne
C6BE programpointer efter DE
C6C2 linie adresse efter HL
C6C5 sæt aktuel linieadresse
C6CA BASIC-Stak pointer
C6CD plus 5
C6CF programpointer efter 'NEXT'
C6D2 sæt BASIC-Stakpointer
C6D6 følgende komma ?
C6D9 ja, næste NEXT-sløjfe
****** søg åben FOR-NEXT-sløjfe
C6DC BASIC-Stakpointer
C6EB 'WHILE-WEND' ?
C6F8 sammenlign HL <> DE
C70A Heltal ?
C70C ja
C713 sæt variabel-type og -adresse
C717 Flag for første gennemløb
C71B ja, spring over addition
C71F Addition
C729 kopier variablen efter (HL)
C730 aritmetisk sammenligning
C734 10
C73F første gennemløb ?
C742 ja, spring over addition
C749 hent STEP-værdi efter HL
C74C Heltals-addition HL:= HL + DE
C74F 'Overflow'
C751 vis fejlmelding
C761 Heltals-sammenligning
****** BASIC-kommando IF
C76A hent udtryk
C76D 'GOTO'
C771 test næste tegn
C774 'THEN'
C778 slutning på linie eller søg ELSE-linie
C77C slutning på Statement ?
C77F ja
C780 linienummer
C782 ja til GOTO-kommandoen
C784 linieadresse ?
C786 nej, udfør BASIC-kommandoen
****** BASIC-kommando GOTO
C789 hent linieadresse
C78D overfør adresse som programpointer
****** BASIC-kommando GOSUB
C78F hent linieadresse
C794 kendetegn for normale 'GOSUB'
C796 Underprogrammernes adresse mærkes
C797 6 Bytes
C799 resever plads i BASIC-Stak
C79F adresse efter anvisning på 'GOSUB'
C7A0 på BASIC-Stak
C7A3 aktuel linieadresse på HL
C7A8 linieadresse på BASIC-Stak
C7AB kendetegn for 'GOSUB'
C7B1 programpointer på underprogrammer
****** BASIC-kommando RETURN
C7B4 søg 'GOSUB' på BASIC-Stak
C7B7 tilbagesæt BASIC-stakpointer
C7BA Ken byte
C7BD adresse efter anvisning på 'GOSUB'
C7BE hent DE
C7C1 linieadresse efter HL
C7C4 sæt aktuelt linienummer
C7C8 Kenbyte

C7C9 en lille en ?
C7CB ja normal 'GOSUB'
C7CC en gang, så GOSUB efter AFTER/EVERY
C7CF til Event-Rutine
C7D6 hent kendetegn på BASIC-Stack
C7DB tilbagesæt BASIC-Stackpointer
C7E0 'GOSUB'
C7E6 vis fejlmelding
C7E9 'Unexpected RETURN'
****** BASIC-kommando WHILE
C7EB søg tilhørende WEND
C7EE adresse mærkes
C7F0 linieadresse for 'WHILE-WEND'
C7F6 sæt BASIC-Stackpointer
C7F9 7 Bytes
C7FB reserver plads i BASIC-stack
C7FF aktuel linieadresse efter HL
C804 linieadresse på BASIC-Stack
C809 'WEND's adresse på BASIC-Stack
C810 WHILE-betingelsens adresse
C811 på BASIC-Stack
C813 kendetegn for 'WHILE'
C816 sæt BASIC-Stackpointer
C81B WHILE-betingelsen testes
****** BASIC-kommando WEND
C81D
C822 'Unexpected WEND'
C824 vis fejlmelding
C82C sæt BASIC-Stackpointer
C82F aktuel linieadresse efter HL
C832 linieadresse for WHILE-WEND
C83B sæt aktuel linieadresse
C848 hent udtryk
C84B hent fortegn
C84F er betingelsen opfyldt ?
C850 linieadresse for WHILE-WEND
C853 sæt som aktuel linieadresse
C858 frigiv plads i BASIC-Stack
******
C860 BASIC-Stackpointer
C87B sammenlign HL <> DE
****** BASIC-kommando ON
C885 'ERROR'
C88A hent 8-Bits-værdi
C88F 'GOTO'
C894 test næste tegn
C897 'GOSUB'
C899 gennemlæs næste Blank
C89C fornedre tæller
C89F hent linienummer efter DE
C8A2 følgende komma ?
****** Event-forarbejdning (AFTER/EVERY)
C8B5
C8B9 KL NEXT SYNC
C8BC er der ikke sket noget ?
C8BE prioriteter mærkes
C8C2 slet Bit 7
C8C8 adresse for Eventblocks
C8C9 KL DO SYNC
C8D4 KL DONE SYNC
C8D9 næste Event
C8E0 afbrydelse gennem 'ESC'
C8ED 'Break'
C8F2 på heltalssløjfe
C8FC ON-BREAK-adresse
C901 linienummer efter HL
C906 Direkte mode ?
C909 SOUND CONTINUE
C915 test på næste tegn
C918 'GOSUB'
C919 hent linieadresse
C91D efter BC
C920 10
****** Event-rutine
C929
C92D hent linienummer/direktmode ?
C932 ja
C934 Kendebyte for AFTER/EVERY-GOSUB
C937 GOSUB-kommando
C93A adresse på aktuelle Statements
C949 adresse på aktuelle Statements
C95A -8
C95E KL DONE SYNC
C968 -4
C96C KL DONE SYNC
C96F afbrydelse gennem 'Break'
C976 på heltalssløjfe

****** BASIC-kommando ON BREAK
C979
C97C gennemlæs Blank
C97F 'CONT'
C984 'STOP'
C986 Defaultværdi 0 ved stop
C98B test næste tegn
C98E 'GOSUB'
C98F hent linieadresse
C993 ON-BREAK-adresse
C997 KM DISARM BREAK
****** BASIC-kommando DI
C99A
C99B KL EVENT DISABLE
****** BASIC-kommando EI
C9A0
C9A1 KL EVENT ENABLE
****** SOUND- og Event-Reset
C9A6 SOUND RESET
C9A9 basisadresse for Eventblokke
C9AC 4 timer
C9AF KL DEL TICKER
C9B3 18
C9B6 addition
C9B7 næste Timer
C9B9 KM DISARM BREAK
C9BC KL SYNC RESET
C9C2 slet ON-BREAK-adresse
C9C5 afbrydelse gennem BREAK
C9C8 adresse på Sound-Queue
C9D4 adresse på Eventblokke
C9DF BASIC-ROM select
C9E1 adresse på Event-rutinen
C9E4 KL INIT EVENT
****** BASIC-kommando ON SQ
C9F8 TEST PÅ '('
C9FB hent 8-Bits-værdi
C9FF beregn adressen på Sound-Queue
CA05 test på ')'
CA08 hent 'GOSUB' og adresse
CA0E SOUND ARM EVENT
****** beregn adressen på Sound-Queue
CA13 er Bit 0 sat ?
CA18 er Bit 1 sat ?
CA1D er Bit 2 sat ?
CA22 'Improper argument'

****** BASIC-kommando AFTER
CA25 hent 16-bits-værdi 0-32767
CA28 Recharge Count på 0
****** BASIC-kommando EVERY
CA2D hent 16-bits-værdi 0-32767
CA30 som Count og
CA31 Recharge Count
CA34 følgende komma ?
CA37 Defaultværdi 0
CA3A ja, hent heltalsværdien med fortegn
CA3E hent Eventblok's Timer# adresse
CA42
CA47 hent 'GOSUB' og adresser
CA4E KL ADD TICKER
****** BASIC-funktionen REMAIN
CA53 CINT
CA56 hent adresse på Eventblok
CA59 KL ADD TICKER
CA5C fundet ?
CA5E nej, 0
CA62 overfør heltal i HL
****** beregn adressen på Eventblok
CA65
CA66 Hi-Byte lig 0 ?
CA67 nej 'Improper arguments'
CA6A stor lig 4
CA6C ja,'Improper arguments'
Ca79 *18
CA74 Basisadresse Eventtabeller
CA77 plus Offset
****** søg tilhørende NEXT
CA79
CA7A aktuel linieadresse til HL
CA7F tæller for indkapsling
CA81 fejlnummer for 'NEXT missing'
CA87 gennemlæs Blank
CA8A 'NEXT'
CA8F 'FOR'
CA93 forhøje indkapslingen
CA94 søg videre
CA99 aktuel linieadresse for HL
CA9D sæt aktuel linieadresse
CAA1 fornedre indkapslinger
CAA2 er tilhørende NEXT fundet ?
CAA4 gennemlæs Blank
CAA7 linierne ?

CAAB søg variabel
CAB0 følgende komma ?
CAB3 nej
CAB5 næste variabel efter NEXT
CAB8 er tilhørende NEXT fundet ?
CABA ja
CABD aktuel linieadresse efter HL
CAC1 sæt aktuel linieadresse
CAC6 søg videre
CAC9 gennemlæs Blank
****** søg tilhørende WEND
CACC
CACE aktuel linieadresse i HL
CAD2 tæller for indkapslingen
CAD4 forhøje
CAD5 fejlnummer for 'WEND missing'
CADB gennemlæs Blank
CADF 'WHILE'
CAE1 forhøj indkapslingen
CAE3 'WEND'
CAE7 formindsk indkapslingen
CAE9 gennemlæs Blank
CAEC gennemlæs Blank
CAEF hent indgangslinie
CAF3 vælg Stream 0
CAF6 initialisere Stackpointer
CAF9 på Heltalssløjfe
****** hent indgangslinie
CAFC Pointer på indgangsbuffer
CAFF slet bufferindholdet
CB01 hent indgangslinie
****** editer linie
CB04 Pointer på indgangsbuffer
CB07 editer linie
CB0A vis LF
****** hent indgangslinie for Diskette
CB0D
CB0E Pointer på indgangsbuffer
CB18 DISK IN CHAR
CB1D CR
CB25 LF
CB2D vis fejlmelding
CB30 'Line too long'
CB32 LF
****** slet fejlnummer
CB3A
****** sæt fejlnummer
CB3B Diskettefejl
CB3E fejlnummer
CB41 aktuel linieadresse i HL
CB44 som ERROR-Line
****** vis fejlmelding
CB48 tilbagespringsadresse i HL
CB49 hent tegn i CALL-kommandoen
CB4A som fejlnummer, vis melding
****** vis 'Syntax error'
CB4C fejlnummer for 'Syntax error'
CB4E vis fejlmelding
****** vis 'Improper argument'
CB50 fejlnummer for 'Improper argument'
CB52 vis fejlmelding
****** BASIC-kommando ERROR
CB54 hent 8-Bits-værdi ulig 0
CB58 sæt fejlnummer og linie
CB5B adresse på aktuel Statement
CB5E Programpointer i ERROR
CB61 linieadresse og programpointer mærkes
CB67 Stackpointer på C000
CB70 initialisere descriptorstack
CB79 adresse på ON-ERROR-rutine
CB7D flag for fejlbehandling ?
CB8B i Heltalssløjfe
CB90 fejlnummer
CB93 beregn adressen på fejlmeldingen
CB99 som aktuel tegnnummer
CBA3 diskettefejl
CBAA til READY-mode
CBAD adresse på ERROR-linie
CBB0 hent linienummer på HL
CBBA pointer på 'Division by zero'
CBBD fejlnummer
CBC3 pointer på 'Overflow'
CBC6 fejlnummer
CBCA adresse på ON-ERROR-rutinen
CBD0 fejlnummer i 'Akku'
CBD1 vis fejlmelding
CBD4 Streamnummer på 0
CBD5 vælg Stream
CBD8 gamle Streamnumre mærkes
CBDA vis fejlmelding
CBDE vis LF
CBE1 gamle Streamnumre

CBE2 vælg
CBE9 initialisere billedskærm
CBEC vis 'Undefinied line'
CBEF vis linienummer
CBF2 vis 'i linienummer'
CBF4 'Undefinieed line',0
CC04 pointer på 'Break'
CC0A initialisere billedskærm
CC0D vis fejlmelding
CC10 hent linieadresse
CC13 Direktemode ?
CC15 pointer på 'in'
CC18 vis String
CC1C vis linienummer
CC1F 'Break'
CC24 'in',0
****** BASIC-kommando STOP
CC29
CC2B vis 'Break in linienummer'
CC32 til READY-mode
****** BASIC-kommando END
CC34
****** Diskettefejl
CC3A Disk-Error mærkes
CC3D vis fejlmelding
CC40 'Nr.32'
CC4A til READY-mode
CC66 til READY-mode
CC6A hent linienummer i HL
CC6E Direktemode ?
CC70 slutning på Statement ?
CC7B sæt aktuel linieadresse
CC87 Direktemode ?
CC8A ja
CC8B linieadresse i HL
CC8E linieadresse i afbrydelse
CC92 programpointer i afbrydelse
****** BASIC-kommando CONT
CC96
CC97 programpointer i afbrydelse
CC9B test på direktemode
CC9C 'Cannot Continue'
CC9E vis fejlmelding
CCA2 linieadresse i afbrydelse
CCA5 sæt aktuel linieadresse
CCA8 SOUND CONTINUE
CCAC til Heltalssløjfe
CCB0 slet flag for i fejlbehandling
CCB6 Adresse på ON-ERROR-rutinen
****** ON ERROR
CCBB gennemlæs Blank
CCBE test på næste tegn
CCC1 'GOTO'
CCC2 hent linienummer i DE
CCC6 søg BASIC-linie DE
CCCB sæt adresse på ON-ERROR-rutinen
****** BASIC-kommando ON ERROR GOTO 0
CCCD ON-ERROR-adresse på 0
CCD0 i fejlbehandlingen ?
CCD4 nej
CCD5 vis fejl
****** BASIC-kommando RESUME
CCD8
CCDA 'NEXT'
CCDE hent linieadresse
CCE2 i fejlbehandlingen ?
CCE8 til Heltalssløjfe
CCEB i fejlbehandlingen ?
CCEF til Heltalssløjfe
CCF2 gennemlæs Blank
CCF6 i fejlbehandlingen
CCFA gennemlæs resten af linie
CCFD i fejlbehandlingen ?
CD01 'Unexpected RESUME'
CD03 nej, vis fejlmelding
CD07 slet ERROR-nummer
CD0A slet flag for i fejlbehandlingen
CD0D adresse på ERROR-linie
CD10 som aktuel linieadresse
CD13 programpointer i ERROR
****** fejlmelding
****** vis fejlmelding
CE76 basisadresse på fejlmelding
CE79 sæt pointer på fejlmelding
CE7D hent tegn i fejlmelding
CE7E slet Bit 7
CE80 printbar tegn ?
CE82 ja, vis
CE85 nej, sæt fejlmelding
CE89 hent tegn endnu engang
CE8A forhøj pointer
CE8B test Bit 7
CE8C ikke sat, vis videre
****** sæt pointer DE på fejlmelding

```
CE8F
CE95    nummer 0 ?
CE96    færdig
CE98    fejlnummer i B
CE99    hent tegn
CE9A    forhøj pointer
CE9B    test Bit 7
CE9C    ikke sat, gennemlæs melding
CE9E    næste fejlmelding
CEA0    DE viser begyndelsen af
        meldingen
******  hent 8-bits-værdi
CEBB    hent heltalsværdi med fortegn
CEBF    Hi-Byte
CEC1    ulig 0, 'Improper argument'
CEC4    overfør Lo-Byte
******  hent 8-Bits-værdi ulig 0
CEC6    hent heltalsværdi med fortegn
CECC    ulig 0 ?
CECE    'Improper argument'
******  hent 16-Bits-værdi, 0 til 32767
CED1    hent heltalsværdi med fortegn
CED5    Hi-Byte
CED6    test Bit 15
CED7    sat, 'Improper argument'
******  hent heltalsværdi med fortegn
CEDB    hent udtryk
CEE0    CINT
CEE6    hent udtryk
CEE9    test String
CEEC    nej
******  hent 16-Bits-værdi, andre
        udtryk
CEF8    hent udtryk
CEFE    UNT
******  hent Stringudtryk og
        parametre
CF06    hent udtryk
CF09    hent Stringparametre
******  hent Stringudtryk
CF0C    hent udtryk
CF0F    Type String, er 'Type
        mismatch'
******  hent linienummerområde
CF12    1 og
CF15    65535 som Default
CF18    følgende komma ?
CF1B    nej, slutning på Statement
CF1E    ja
CF1F    '#'
CF22    '-'
CF26    hent linienummer i DE
CF2A    og i BC
CF2C    følgende komma ?
CF2F    ja
CF30    test følgende tegn
CF33    '-'
CF34    65535 som Default-Endværdi
CF38    følgende komma ?
CF3B    ja
CF3C    hent linienummer i DE
CF3F    følgende komma ?
CF46    'Improper argument'
******  hent linienummer i DE
CF4B    konstanttype
CF4E    værdi i DE
CF50    linienummer ?
CF52    ja, færdig
CF54    linieadresse ?
CF56    nej, 'Syntax error'
CF5A    HL vises på liniebegyndelsen
CF5F    linienummer i DE
CF62    gennemlæs Blank
******  hent udtryk
CF65
CF66    Hierarchie-kode 0
CF68    hent Term
CF6D    gennemlæs Blank
******  hent Term
CF70
CF72    hent udtryk
CF78    operator
CF79    '>'
CF7B    mindre ?
CF7C    'NOT'
CF7E    større lighed ?
CF7F    '+'
CF81    mindre end
        sammenlignsoperator
CF83    '+', så test String
CF86    ingen String
CF8A    Stringdescriptor
CF8D    på Stack
CF8E    hent udtryk
CF91    Type String, er 'Type
        mismatch'
CF95    Stringaddition
CF98    bearbejd næste Term
```

****** arithmetiske operator
CF9A
CF9B minus #F4
CF9E gange 4
CFA2 plus #CFF0, tabeladresse
CFA9 Hierarchie-kode
CFAB mindre, færdig
CFAD formindskelse lægges på Stack
CFB3 Hierarchie-Kode
CFB4 hent Term
CFC0 reserver plads i BASIC-Stack
CFC3 JFP (DE), gennemfør operationen
CFC6 bearbejd næste Term
****** sammenlignsoperationen
CFC8
CFCD Token
CFCE minus Offset
CFD1 test String
CFD4 adresse på arithmetisk sammenligning
CFD7 ingen String
CFDA Stringdescriptor
CFDD på Stack
CFDF Hierarchie-kode
CFE1 hent Term
CFE7 Stringsammenligning
CFEB hent formindsket sammenligning
CFEE bearbejd næste Term
****** BASIC-operatoren Hierarchie-kode + adresse
CFF0 F4, '+'
CFF3 F5, '-'
CFF6 F6, '*'
CFF9 F7, '/'
CFFC F8, '↑'
CFFF F9, 'Backslash'
D002 FA, 'AND'
D005 FB, 'MOD'
D008 FC, 'OR'
D00B FD, 'XOR'
****** arithmetisk sammenligning
D00E
D012 arithmetisk sammenligning
D01D overfør fortegn
****** '-'negative fortegn
D020 Hierarchie-kode
D022 hent Term
D926 Fortegns skifter
****** BASIC-operator NOT
D02B Hierarchie-kode
D02D hent Term
D031 NOT-operator
****** hent udtryk
F036 gennemlæs Blank
****** hent udtryk
D039 'Operand missing'
D03D hent variabel
D041 hent numerisk værdi
D043 ""
D045 hent String
D048 Funktion ?
D04A til funktionsberegning
D04E Basisadresse for tabellerne
D051 gennemsøg tabellerne
D055 gennemlæs Blank
D058 vis fejlmelding
D05B 'Operand missing'
****** særfunktion
D05C Antal tabelindtastning
D05D ikke fundet, 'Syntax error'
D05E '-'
D062 '+'
D065 '('
D068 'NOT'
D06B 'ERL'
D06E 'FN'
D071 'MID$'
D074 'SS'
****** hent variabel
D077 hent variabeladresse
D07A endnu ikke anlagt ?
D07C variabeltype
D07E String ?
D087 String ?
D089 slet variabel
D08C pointer på 0
D08F som Stringdescriptor
D094 Stringlængde 0
****** hent numerisk værdi
D095 stryge Offset
D09A mindre end 10?
D09C ja, hent ciffer
D0A0 Een-Byte-værdi ?
D0A2 ja
D0A6 To-Byte-værdier (dez,hex,bin) ?

D0A8 ja
D0AA flydende kommaværdi placering
D0AC ja
D0AE 'Syntax error'
D0B1 'Real'
D0B3 sæt variabeltype
****** hent To-Byte-Værdi
D0B9
****** hent flydende kommaværdi placering
D0C0
D0CD variabeltype af 'Real'
D0D1 gennemlæs Blank
****** '(' hent Term i klammen
D0D4 hent udtryk
D0D7 test ')'
******
D0DA 'Syntax error'
****** funktionsberegning
D0DD forhøj programpointer
D0DE hent Token
D0DF gennemlæs Blank
D0E2 test Token
D0E5 40-49, reserverede variabler
D0E9 hent adresse på reserverede variabel
D0EC test '('
D0F0 Token 2 gange
D0F6 'Syntax error'
D0FA beregn funktionen
D0FC hent funktionsargument i klammen
D100 beregn funktionen
****** beregn funktionen
D105 funktionens adresse
D10A slet Hi-Byte
D10C adder Token 2 gange
D111 udfør funktionen
****** hent adresse på reserverede variabler
D113 fordobbel Token
D115 Basis adresse for tabel-Offset
****** adresse på reserverede variabler
D11A 40, EOF
D11C 41, ERR
D11E 42, HIMEM
D120 43, INKEY$
D122 44, PI
D124 45, RND
D126 46, TIME
D128 47, XPOS
D12A 48, YPOS
D12C 49, DERR
****** reserverede variabler DERR
D12E Disk-Error-nummer
****** reserverede variabler ERR
D133 ERROR-nummer
D137 overfør Akkuindholdet som heltal
****** reserverede variabler TIME
D13D KL TIME PLEASE
D140 4-bytes-værdi i flydende komma forvandling
****** reserverede variabel ERL
D146 hent ERROR-linienummer
****** reserverede variabler HIMEM
D14B
D14C HIMEM
D14F overfør værdi
****** 'SS', variabelpointer
D151 hent variabeladresse
D154 ikke defineret, 'Improper argument'
D15A String ?
D15F overfør værdi
****** reserverede variabler XPOS
D164
D165 GRA ASK CURSOR
D168 spalteværdi i HL
****** reserverede variabler YPOS
D16C FRA ASK CURSOR
D16F overfør heltal i HL
****** BASIC-kommando DEF
D174 test på næste tegn
D177 'FN'
D179 hent linienummer i HL
D17D 'Invalid direct command'
D17F vis fejlmelding
D182 søg funktion
D18A gennemlæs resten af Statement
****** BASIC-funktion FN
D18D søg funktion
D199 'Unknown user function'
D19B vis fejlmelding
D1A2 '('

```
D1A6    gennemlæs Blank
D1AA    test '('
D1B3    hent udtryk
DaB8    vis en værdi af en variabel
D1BC    følgende komma ?
D1BF    nej
D2C2    test ','
D1C5    næste variabel
D1C7    test ')'
D1CB    test ')'
D1D1    test '='
D1D4    hent udtryk
D1D7    'Syntax error'
D1DA    test String
D1E5    prøv lignende variabeltyper
******  BASIC-funktionen med flere
        argumenter
D1E8    71, BIN$
D1EA    72, DEC$
D1EC    73, HEX$
D1EE    74, INSTR
D1F0    75, LEFT$
D1F2    76, MAX
D1F4    77, MIN
D1F6    78, POS
D1F8    79, RIGHT$
D1FA    7A, ROUND
D1FC    7B, STRING$
D1FE    7C, TEST
D200    7D, TESTER
D202    7E, COPYCHR$
D204    7F, VPOS
******  BASIC-funktionens adresse
D206    00, ABS
D208    01, ASC
D20A    02, ATN
D20C    03, CHR$
D20E    04, CINT
D210    05, COS
D212    06, CREAL
D214    07, EXP
D216    08, FIX
D218    09, FRE
D21A    0A, INKEY
D21C    0B, INP
D22E    0C, INT
D220    0D, JOY
D222    0E, LEN
D224    0F, LOG
D226    10, LOG 10
D228    11, LOWER$
D22A    12, PEEK
D22C    13, REMAIN
D22E    14, SGN
D230    15, SIN
D232    16, SPACE$
D234    17, SQ
D236    18, SQR
D238    19, STR$
D23A    1A, TAN
D23C    1B, UNT
D23E    1C, UPPER$
D240    1D, VAL
******  BASIC-funktionen MIN
D242    flag for MIN
******  BASIC-funktionen MAX
D246    flag for MAX
D248    hent udtryk
D24B    følgende komma ?
D24E    nej, test ')', færdig
D251    læg variabel på Basic-Stack
D254    hent udtryk
D259    frigiv plads i BASIC-Stack
D25E    aritmetisk sammenligning
D267    hent formindsket
        sammenligning
D26B    næste argument
******  BASIC-funktionen ROUND
D26D    hent udtryk
D270    og læg på BASIC-Stack
D273    følgende komma ?
D276    Default 0
D279    ja, hent heltalsværdien med
        fortegn
D27C    test ')'
D281    39
D284    adder
D285    79
D288    sammenlign HL<>DE
D28B    større, 'Improper argument'
D290    frigiv plads i BASIC-Stack
D293    tal afrundes efter B
D294    tal afrundes
******  BASIC-kommando CAT
D29B    afbryd Disk I/O
D2A1    DISK CATALOG
D2A4    diskettefejl mærkes
```

******	BASIC-kommando OPENOUT
D2AB	
D2B4	DISK OUT OPEN
******	BASIC-kommando OPENIN
D2B7	
D2BD	vis fejlmelding
D2C0	'File type error'
D2C1	hent filnavn
D2C7	DISK IN OPEN
D2CD	hent Stringudtryk og -parametre
D2D2	test Systemmelding
D2D5	diskettefejl mærkes
D2DA	vis fejlmelding
D2DD	'file already open'
******	test Systemmelding
D2DE	
D2E0	ingen filnavn ?
D2E3	første tegn af navnet
D2E4	'!'
D2E8	nej
D2EA	sæt pointer på andet tegn
D2EB	formindsk længden
D2EC	flag vendes
D2ED	CAS NOISY
******	BASIC-kommando CLOSEIN
D2F0	
D2F1	DISK IN CLOSE
******	BASIC-kommando CLOSEOUT
D2F8	
D2F9	DISK OUT CLOSE
D2FC	Diskettefejl mærkes
******	afbryd Disk I/O
D303	
D306	DISK IN ABANDOM
D30C	DISK OUT ABANDOM
******	BASIC-kommando SOUND
D316	hent 8-bits værdi
D319	Kanal-Status
D31C	test ','
D31F	hent argument 0 til 4095
D322	Ton-Periode
D326	følgende komma ?
D329	Defaultværdi 20
D32C	ja, hent heltalsværdi med fortegn
D32F	Dauer
D333	max. 15, Default 12
D336	hent allerede eksisterende argument
D339	Lydstyrke
D33C	max. 15, Default 0
D33E	hent allerede eksisterende argumenter
D341	Lydstyrke-hulkurve
D344	hent allerede eksisterende argumenter
D347	Ton-Hilkurve
D34A	max. 31, Default 0
D34C	hent allerede eksisterende argumenter
D34F	Ryge pause
D352	slutningen på Statement, er 'Syntax error'
D356	Adresse og Sound-Parametreblocks
D359	SOUND QUEUE
D35F	heltalssløjfe
******	hent allerede eksisterende 8-bits-værdier
D362	følgende komma ?
D365	load Defaultværdi
D366	ingen komma, færdig
D358	','
D36C	hent 8-bits-værdi
D36F	sammenligning med maximalværdi
D370	mindre, ok
D371	'Improper argument'
******	BASIC-kommando RELEASE
D373	8
D375	hent 8-bits-værdi $<$ 8
D379	SOUND RELEASE
******	BASIC-funktion SQ
D37E	CINT
D383	test bit 0
D384	sat ?
D386	test Bit 1
D387	sat ?
D379	test Bit 2
D38A	ikke sat, 'Improper argument'
D38C	Hi-Byte $>$ 0?
D38D	ja, 'Improper argument'
D390	SOUND CHECK
D393	overfør akkuindhold som heltal

```
******  hent argument -128 til +127
D396    hent heltalsværdi med fortegn
D39E    'Improper argument'
******  BASIC-kommando ENV
D3A1    hent 8-bits-værdi ulig 0
D3A4    større end 16
D3A6    ja, 'Improper argument'
D3AC    hent parametre
D3B1    adresse på parametreblokke
D3B5    SOUND AMPL ENVELOPE
D3BB    '='
D3BF    gennemlæs Blank
D3C2    16
D3C4    hent 8-bits-værdi < 16
D3C7    sæt Bit 7
D3CA    test ','
D3CD    hent 16-bits-værdi
D3D0    128
D3D2    hent 8-Bits-værdi < 128
D3D5    hent 2 argumenter
******  BASIC-kommando ENT
D3D7    hent argument -128 til +127
D3DD    0 ?
D3E2    0 ?
D3E3    'Improper argument'
D3E5    større end 16 ?
D3E7    'Improper argument'
D3ED    hent parametre
D3F2    adresse på parameterblokke
D3FB    SOUND TONE ENVELOPE
D401    '='
D405    gennemlæs Blank
D408    hent argument 0 til 4095
D412    240
D414    hent 8-Bits-værdi < 240
D418    test ','
D41B    hent argument -128 til +127
D41F    test ','
D422    hent 8-Bits-værdi
******  hent parametre for ENT &
        ENV
D428
D42B    følgende komma ?
D432    JP (DE)
D439    adresse på parametreblokke
D44C    slutning på Statement, er
        'Syntax error'
******  hent argument 0 til 4095
D44F    hent heltalsværdi med fortegn
D452    Hi-Byte
D453    er Bits 12 - 15 sat
D455    ja, 'Improper argument'
******  BASIC-funktion INKEY
D459    CINT
D45C    80
D45F    sammenlign HL <> DE
D462    'Improper argument'
D465    KM TEST KEY
D468    -1 når der ikke gentrykkes
D46D    formindsk L
D470    overfør heltal i HL
******  BASIC-funktion JOY
D473    KM GET JOYSTICK
D477    CINT
D483    overfør akkuindhold som
        heltal
D486    'Improper argument'
******  BASIC-kommando KEY
D489    'DEF'
D48D    hent 8-Bits-værdi
D481    test ','
D494    hent Stringudtryk og
        parametre
D497    Stringlængde efter C
D499    tastnummer efter B
D49B    Stringadresse efter HL
D49C    KM SET EXPAND
D4A0    'Improper argument'
******  KEY DEF
D4A3    gennemlæs Blank
D4A6    80 som maximalværdi
D4A8    hent 8-bits-værdi < 80
D4AC    test ','
D4AF    2
D4B1    hent argument < 2
D4BA    KM SET REPEAT
D4BF    KM SET TRANSLATE
D4C2    test videre argument
D4C5    KM SET SHIFT
D4C8    test videre argument
D4CB    KM SET CONTROL
D4CE    følgende komma ?
D4D1    nej, færdig
D4D3    hent 8-Bits-værdi
D4D9    spring efter (HL)
******  BASIC-kommando SPEED
D4DE    'WRITE'
D4E2    'KEY'
```

D4E4 KM SET DELAY
D4E9 'INK'
D4EB SCR SET FLASHING
D4EE 'Syntax error'
****** SPEED KEY & INK
D4F1
D4F2 gennemlæs Blank
D4F5 hent 8-Bits-værdi ulig 0
D4F9 test ','
D4FC hent 8-Bits-værdi ulig 0
D503 spring efter (BC)
****** SPEED WRITE
D508 gennemlæs Blank
D50B 2
D50D hent argument < 2
D511 167
D517 0 ?
D519 nej, sidekonstant fordoblet
D51B CAS SET SPEED
****** reserverede variabel PI
D520
D521 sæt Type på 'Real'
D524 variabeltype efter C, HL på variabel
D527 hent PI
****** BASIC-kommando DEG
D53C flag for DEG
****** BASIC-kommando RAD
D530 flag for RAD
D531 sæt DEG/RAD-Mode
****** BASIC-funktion SQR
D534 SQR-funktion
****** BASIC-operator '↑'
D539
D53B CREAL
D53F mellemlager for flydende kommavariabel
D542 kopier (DE)'s variabel efter (HL)
D548 sæt variabeltype og adresse
D54C potensering
D54F udfør funktionen
D552 fejlfri ?
D553 'Fivision by zero'
D556 'Overflow'
D669 'Improper argument'
****** udfør flydende komma funktionen
D55C
D55E CREAL
D562 udfør funktionen
****** BASIC-funktion EXP
D563 EXP-funktionen
****** BASIC-funktion LOG 10
D568 LOG 10-funktion
****** BASIC-funktion LOG
D56D LOG-funktion
****** BASIC-funktion SIN
D572 SIN-funktion
****** BASIC-funktion COS
D577 COS-funktion
****** BASIC-funktion TAN
D57C TAN-funktion
****** BASIC-funktion ATN
D581 ATN-funktion
D586 'Random number seed ?',0
****** BASIC-kommando RANDOMIZE
D59C
D59E hent udtryk
D5A5 'Random number seed ?'
D5A8 vis
D5AB hent indgangslinie
D5AE vis LF
D5B1 læs indgang
D5B4 ugyldig, hent videre
D5B6 gennemlæs Blank, TAB og LF
D5BA ugyldig, hent videre
D5BC CREAL
D5BF SET RANDOM SEED
****** reserveret variabel RND
D5C4
D5C5 ')'
D5C9 gennemlæs Blank
D5CC hent udtryk
D5CF test ')'
D5D3 CREAL
D5D6 SGN
D5DB hent sidste RND-værdi
D5E0 negativ, SET RANDOM SEED
D5E5 sæt type og flydende komma
D5E8 RND
****** sæt variabelpointer tilbage
D5ED slet tabeller
D5F0 programslutning
D5F3 variabelstart
D5F6 Arraystart

```
D5F9   Arrayslutning
****** slet tabeller
D5FD   tabellers basis
D600   54 = 2*27, A - Z plus
       funktionen
D602   slet #ADB7 til #ADEC
D60A   slet #ADED til #ADF2
****** slet flag for FN
D611
****** beregn tabeladresse
D61A   'Z'+1
D61C   variabelstart
D620   minus 1
D621   gange 2
D624   plus #AD35
****** beregn tabeladresse for Array
D62A   Araystart
D62E   minus 1
D632   gange
D635   plus #ADED
****** alle variabler af typen REAL
D63B   'AZ'
D63E   type 'Real'
D641   antal efter A
D642   lille 1, så 'Syntax error'
D646   basis for tabeller lig #ADB2
D64B   bogstav lig pointer i tabeller
D64E   alle bogstaver
****** BASIC-kommando DEFSTR
D653   type 'String'
****** BASIC-kommando DEFINT
D657   type 'heltal'
****** BASIC-kommando DEFREAL
D65B   type 'Real'
D65D   hent bogstav
D65E   test bogstav
D661   'Syntax error'
D663   efter BC (fra - til)
D665   gennemlæs Blank
D668   '-'
D66C   gennemlæs Blank
D66F   test bogstav
D672   'Syntax error'
D674   til
D675   gennemlæs Blank
D678   sæt variabeltype
D67B   følgende komma ?
D67E   ja, lav videre
D681   'Syntax error'
D684   vis fejlmelding
D687   'Subscript out of range'
D688   vis fejlmelding
D68B   'Array already dimensioned'
D68C   2*#7C '1'
D68E   kommandosvar
****** BASIC-kommando LET
D691   hent variabel
D695   test '='
D698   hent udtryk
D69D   vis værdi af variabel
****** vis værdi af variabel
D6A2   variabeltype
D6A3   og udfaldstype
D6A6   sammenlign
D6A8   typetilpasning er 'type
       mismatch'
D6AB   test String
D6AE   nej, kopier variabel efter (HL)
D6B2   Stringforvaltning
D6B6   overfør pointer af String
****** BASIC-kommando DIM
D6B9   Dimensionering
D6BC   følgende komma ?
D6BF   ja, næste variabel
****** søg variabel
D6C2   læs variabelnavn
D6C5   test dimensionerede variabel
D6C8   hent variabeltype
****** hent variabeladresse
D6CC   læs variabelnavn
D6CF   test dimensionerede variabel
D6D2   hent variabeltype
D6D5   første bogstav
D6D6   beregn tabelposition
****** søg funktion
D6DE   læs variabelnavn
D6E4   beregn tabelposition for FN
D6EA   anlæg funktion
D6EF   læs variabelnavn
D6F5   første bogstav
D6F6   beregn tabelposition
D6FC   variabeltype
D705   variabelstart
D70C   variabeltype
D712   læs variabelnavn
D715   test indiserede variabler
D718   hent variabeltype
D72A   søg Array
```

D72E fundet ?
D733 søg Array
D736 fundet ?
****** søg Array
D75B variabeltype
D77A sæt Bit 6, 'FN'
D787 Arraystart
D78B reserver plads i variabelområdet
D78E forhøj pointer for Arryområdet
D7C6 LDIR
D7C9 variabeltype
****** Dimensionering
D7E4 hent variabelnavn
D7E8 '('
D7DC '['
D7EE 'Syntax error'
D7F6 variabeltype
D7F9 beregn tabelposition for Array
D7FC søg Array
D7FF fundet, 'Array already dimensioned'
****** test dimensionerede variabel
D80A
D80C '('
D810 '['
D817 variabelstart
****** dimensionerede variabler
D820
D827 Arraystart
D830 variabeltype
D833 beregn tabelposition for Array
D836 søg Array
D839 ikke fundet ?
D845 'Subscript out of range'
D857 Antal dimensioner
D85C Arraygrænse efter DE
****** læs Indices
D887
D888 gennemlæs Blank
D88B variabeltype
D88E mærkes
D88D antal Indices
D891 hent 16-Bits-værde 0 - 32767, Index
D897 reserver plads i BASIC-Stack
D89B Index af BASIC-Stack
D89E forhøj antal Indices
D89F følgende komma ?
D8A2 ja, næste Index
D8A5 ')'
D8A9 ']'
D8AB 'Syntax error'
D8AE gennemlæs Blank
D8B2 istandsætte variabeltype
D8C4 Array slutning
D8C8 reserver plads i variabelområdet
D8D5 variabeltype
D8E3 10, Defaultværdi for Index
D92B 2 Bytes
D92D frigiv' plads i BASIC-Stack
****** læs variabelnavn
D935 fastsæt variabeltype
D93D er variabel allerede lagt
D93E nej
D941 gennemlæs navnenes bogstaver
D942 test Bit 7
D945 gennemlæs næste Blank
D94B sæt pointer af variabeltype
D988 variabeltype
D991 #05 + #09 => #0D
D995 40
D997 reserver plads i BASIC-Stack
D99B 41
D99D 40 tegn allerede ?
D99E ja, så 'Syntax error'
D9A1 læs næste tegn på navn
D9A3 forvandel små bogstaver til store bogstaver
D9A7 sidste tegn
D9A8 nej
D9AA sæt BASIC-Stackpointer
D9B0 gennemlæs næste Blank
****** fastslå variabeltype
D9B3
D9B6 lille #0B ?
D9B8 -#09, #0D => #05
D9BA '!', Real-variabel ?
D9BC sæt Type af 'Real'
D9BE 'Syntax error'
D9C0 '%', heltal-variabel ?
D9C2 eller '$', String ?
D9C4 nej, 'Syntax error'
D9C7 'Real'
D9C9 variabeltype mærkes
****** Arraytabel opdateres
D9CD slet tabel for Array

D9D0 Array slutning
D9D4 Araystart
D9D7 sammenlign HL <> DE
D9DA ingen Array
D9E5 beregn tabelposition for Array
****** BASIC-kommando ERASE
D9F4
D9F7 slet Array
D9FA følgende komma ?
D9FD ja, næste Array
****** slet Array
DA00 læs variabelnavn
DA04 variabeltype
DA07 beregn tabelposition for Array
DA0A søg Array
DA0D ikke fundet, 'Improper argument'
DA16 BC := HL - DE
DA1F Arraytabel opdataeres
DA33 6 Bytes
DA35 reserver plads i BASIC-Stack
DA71 reserver plads i BASIC-Stack
DA75 fastslå variabeltype
DA88 variabeltype
DA8D reserver plads i BASIC-Stack
DAAC 26 bogstaver, 'A'
DAB0 første bogstav i navn
DAB1 beregn tabelposition
DAB8 næste bogstav
DaB9 allerede alle bogstaver ?
DABD beregn tabelposition for Array
DAC8 Arraystart
DAE2 sammenlign HL<>BC
DB10 Spring efter (HL)
****** BASIC-kommando LINE
DB18 test næste tegn
DB1B 'INPUT'
DB1C hent kanalnummer
DB1F vis evt. dialogstring
DB22 søg variabel
DB25 Type 'String', er 'Type mismatch'
DB2A hent indgang fra aktiv redskab
DB2D indtag String i Descriptorstack
DB31 vis resultatet af variabel
****** hent indgang fra aktiv redskab
DB36
DB39 hent indgang fra diskette
DB42 ';'
****** BASIC-kommando INPUT
Db48 hent kanalnummer
DB4B hent indgang og forvandel den
DB4F søg variabel
DB59 følgende komma ?
DB5C ja, næste variabel
****** hent indgang og forvandel den
DB60
DB63 vis evt. dialogstring
******
DB7E '?Redo from start', LF,0
****** vis evt. dialogstring
DB90
DB91 ';'
DB93 Trenntegn mærkes
DB96 gennemlæs Blank
DB99 '"'
DB9B ingen String ?
DBA0 følgende komma ?
DBA3 ja
DBA4 test næste tegn
DBA7 ';'
DBAC '?'
DBAE vis
DBB1 ' '
DBB3 vis
DBC6 'Type mismatch'
DBCE vis fejlmelding
DBD1 'Type mismatch'
DBD5 hent variabelnavn og type
DBDF 'String'
DBE1 hent ja, Stringparametre
DBE5 følgende komma ?
DBEA nej
DBEC ','
DBFD test String
DC13 gennemlæs Blank, TAB og LF
DC22 indtag String i Descriptor
DC25 gennemlæs Blank, TAB og LF
DC28 '"'
DC2A læs String
DC39 vis fejlmelding
DC3C 'EOF met'
DC42 '"'
DC53 indgangsbufferens start
DC56 første tegn lig 0
DC59 '"'
DC5F 'EOF met'
DC64 indgangsbufferens start

```
DC6A   JP (DE)
DC7F   CR
DC82   '"'
DCA4   LF + CR
DCA7   CR ?
DCAA   LF ?
DCBE   ','
DCC1   CR
DCC4   ' '
DCC7   TAB
DCCA   LF
******  BASIC-kommando RESTORE
DCCD   ingen linienummer ?
DCCF   hent linienummer efter DE
DCD3   søg BASIC-linie DE
DCD7   sæt DATA-pointer
DCDA   programstart
DCDD   som DATA-pointer
******  BASIC-kommando READ
DCDF
DCE0   DATA-pointer
DCE3   hent næste DATA-element
DCE7   søg variabel
DCEC   ':'
DCF6   ','
DCFA   linieadresse mens READ-
       kommando
DCFD   sæt aktuel linieadresse
DD00   'Syntax error'
DD04   følgende komma ?
DD08   ja
DD0A   DATA-pointer
DD10   ','
DD13   gennemlæs resten af linien
DD16   linierne ?
DD17   nej
DD1A   linielængde
DD1C   0, programslutning ?
DD1E   'DATA exhausted'
DD20   vis fejlmelding
DD23   linieadresse mens READ-
       kommando
DD27   gennemlæs Blank
DD2A   'DATA'
dd2c   nej, søg videre
DD2F   fortegn mærkes
DD30   fremstil absolutværdien
******  overfør fortegn B
DD3C
DD41   resultatets fortegn
DD42   negativ, så skift fortegn
DD47   fortegnsbit vendes
******  heltal-addition HL:= DE +
       HL
DD4F   slet Carry-flag
DD50   addition
DD53   resultat positiv ?
DD54   sæt flag
******  heltals-subreaktion HL := DE
       - HL
DD57   ombyt operatør
DD58   slet Carry-flag
DD59   Subtraktion
DD5C   resultat positiv ?
DD5D   sæt flag
******  heltalsmultiplikation med
       fortegn
DD60   fortegn af resultatet
       bestemmes
DD63   fortegnsløs multiplikation
DD66   overfør fortegnet
******  fortegnet af resultatet
       bestemmes
DD6C   fortegn for HL
DD6D   og fortegnet for DE
DD6E   efter B
DD70   fremstil absolutværdien af DE
DD74   fremstil absolutvardien af HL
******  heltalsmultiplikation uden
       fortegn
DD77
******  heltals-division med fortegn
DDA1   Division HL := HL / DE
DDA4   fortegn overføres
******  heltal-MOD-beregning
DDA8   fortegn mærkes
DDA9   Division
DDAC   resten af HL
DDAD   hent fortegnet tilbage
DDAE   og overfør
******  Division HL := HL / DE, DE
       := Rest
DDB0   resultatets fortegn bestemmes
******  fremstil absolutværdi
DDEF   test fortegnet
DDF1   positiv, allerede færdig
******  heltals-fortegnsskifter
DDF2
```

****** SGN fortegn fra HL
DDFE
****** sammenlign HL<>DE
DE07 fortegn fra HL
DE08 og fortegn fra DE
DE0A tal med samme fortegn sammenlignes
****** test næste komma
DE1A ','
****** test af klammer af
DE1E '('
****** test af klammer til
DE22 ')'
****** test af lighedstegn
DE26 '='
****** test af næste tegn
DE2A hent tilbagespringsadresse
DE2B hent tegn
DE2D tilbagekald programpointer
DE2E tegn sammenligning
DE2F 'Syntax error'
****** gennemlæs Blank
DE31
DE33 ' '
DE35 test videre på Blank
DE37 slutning af Statement ?
****** test linierne, er 'Syntax error'
DE3C
DE40 'Syntax error'
****** test slutningen af Statement
DE42
****** test komma
DE46
DE47 gennemlæs Blank
DE4A ','
DE4D gennemlæs Blank
****** gennemlæs Blank, TAB og LF
DE52 hent tegn
DE53 forhøj pointer
DE54 ' '
DE58 TAB
DE5C LF
DE60 formindsk pointer
****** Interpretersløjfe
DE62 adresse på aktuel Statement
DE65 sæt adresse på aktuel Statement
DE68 KL POLL SYNCHRONOUS
DE6B Event-forarbejdning (AFTER /EVERY)
DE6E gennemlæs Blank
DE71 udfør BASIC-kommando
DE74 læs program tekst
DE75 ':', slutning på Statement?
DE77 ja
DE79 'Syntax error'
DE7D linielængde
DE7E lig 0 ?
DE80 ja, til END-kommando
DE82 aktuel linieadresse mærkes
DE86 TRACE-flag sat ?
DE8A nej
DE8C TRACE-rutine
DE8F til begyndelsen af interpretersløjfe
DE91 Til END-kommando
****** udfør BASIC-kommando
DE94 Token gange 2
DE95 test kommandosvar
DE9A ugyldig Token, 'Syntax error'
DE9F plus #DEE5 (tabeladresse)
DEA7 kommandoadresse Stack
DEA9 gennemlæs Blank, spring til kommando
DEAC 'Syntax error'
****** aktuel linieadresse af 0
DEAF 0
DEB2 som aktel linieadresse
****** load aktuel linieadresse
DEB6 aktuel linieadresse
****** Test Direktmode/hent linieadresse
DEBA aktuel linieadresse
DEBF 0, Direktemode
DEC2 linienummer efter HL
****** BASIC-kommando TRON
DEC6 sæt flag
****** BASIC-kommando TROFF
DECA slet flag
****** TRACE-rutine
DECF '['
DED1 udskriv
DED5 aktuel linieadresse
DED9 linienummer efter HL
DEDC linienummer udskrives
DEE0 ']'
DEE2 udskriv

******	Adresse på BASIC-kommando
DEE5	80, AFTER
DEE7	81, AUTO
DEE9	82, BORDER
DEEB	83, CALL
DEED	84, CAT
DEEF	85, CHAIN
DEF1	86, CLEAR
DEF3	87, CLG
DEF5	88, CLOSEIN
DEF7	89, CLOSEOUT
DEF9	8A, CLS
DEFB	8B, CONT
DEFD	8C, DATA
DEFF	8D, DEF
DF01	8E, DEFINT
DF03	8F, DEFREAL
DF04	90, DEFSTR
DF07	91, DEG
DE09	92, DELETE
DF0B	93, DIM
DF0D	94, DRAW
DF0F	95, DRAWR
DF11	96, EDIT
DF13	97, ELSE
DF15	98, END
DF17	99, ENT
DF19	9A, ENV
DF1B	9B, ERASE
DF1D	9C, ERROR
DF1F	9D, EVERY
DF21	9E, FOR
DF23	9F, GOSUB
DF25	A0, GOTO
DF27	A1, IF
DF29	A2, INK
DF2B	A3, INPUT
DF2D	A4, KEY
DF2F	A5, LET
DF31	A6, LINE
DF33	A7, LIST
DF33	A8, LOAD
DF37	A9, LOCATE
DF39	AA, MEMORY
DF3B	AB, MERGE
DF3D	AC, MID$
DF3F	AD, MODE
DF41	AE, MOVE
DF43	AF, MOVER
DF45	V0, NEXT
DF47	B1, NEW
DF49	B2, ON
DF4B	B3, ON BREAK
DF4D	B4, ON ERROR GOTO 0
DF4F	B5, ON SQ
DF51	B6, OPENIN
DF53	B7, OPENOUT
DF55	B8, ORIGIN
DF57	B9, OUT
DF59	BA, PAPER
DF5B	BB, PEN
DF5D	BC, PLOT
DF5F	BD, PLOTER
DF61	BE, POKE
DF63	BF, PRINT
DF65	C0, '
DF67	C1, RAD
DF69	C2, RANDOMIZE
DF6B	C3, READ
DF6D	C4, RELEASE
DF6F	C5, REM
DF71	C6, RENUM
DF73	C7, RESTORE
DF75	C8, RESUME
DF77	C9, RETURN
DF79	CA, RUN
DF7B	CB, SAVE
DF7D	CC, SOUND
DF7F	CD, SPEED
DF81	CE, STOP
DF83	CF, SYMBOL
DF85	D0, TAG
DF86	D1, TAGOFF
DF89	D2, TROFF
DF8B	D3, TRON
DF8D	D4, WAIT
DF8F	D5, WEND
DF91	D6, WHILE
DF93	D7, WIDTH
DF95	D8, WINDOW
DF97	D9, WRITE
DF99	DA, ZONE
DF9B	DB, DI
DF9D	DC, EI
DF9E	DD, FILL
DFA1	DE, FRAPHICS
DFA3	DF, MASK
DFA5	E0, FRAME

DFA7 E1, CURSOR
DFAA begyndelsen af fri RAM
DFB2 max. 300 tegn
DFB5 hent tegn for inputbuffer
DFB9 sidste tegn ?
DFBA nej
DFBE 301 tællerstand
DFC0 samme linielængde
DFC3 efter B
DFC6 3 x nul som afslutning
****** hent tegn fra inputbufferen
DFCD
DFCE sidste tegn ?
DFD0 bogstav ?
DFD3 ja
DFD5 numerisk ?
DFD8 ja
DFDB '&' ?
DFDD ja
DFE1 Token ?
DFE2 ja
DFE3 '!'
DFE8 ignorer overskydende Blank
DFEC ja
DFED ' '
DFEF skriv i bufferen
DFF5 'REM' ?
DFF7 ja
DFFB tabellens basisadresse
DFFE gennemsøg tabellen
E002 fundet, resten ikke konverteret
E005 'ELSE'
E009 skriv i buffer
E00D skriv tegn i buffer
E00E forhøj bufferviser
E00F nedsæt tæller
E011 tæller lig 0 ?
E012 nej
E013 send fejlmelding
E016 'Line too long'
****** Specielle Token
E017 'DATA'
E018 'DEFINT'
E019 'DEFSTR'
E01A 'DEFREAL'
E01B Slut på tabel
******
E01C Skriv i bufferen
E020 Er bufferens slutning nået ?
E022 ':'
E026 Forhøj bufferpointeren
E028 Token ?
E02B ' '
E02F ' '
E031 '"'
E03B Flag for slut på Statement
E044 Konverter små bogstaver til store bogstaver
E047 Beregn kommandoords adresser
E052 Test for bogstav eller tal
E058 'FN'
E05B Test for bogstav eller tal
E069 'Funktion'
E06B Skriv i buffer
E06E Funktions-Token
E06F skrives i buffer
E07C Test for bogstav eller tal
E086 Token for variabel
E08B Token for REAL-variabel
E090 skrives i buffer
E094 Test for bogstav eller tal
E097 og skriv i buffer
E09D Test for bogstav eller tal
E0A3 og skriv i buffer
E0B6 Tabellens basis-adresse
E0B9 Gennemsøg tabellen
****** Kommandoer med linienumre
E0C8 'RESTORE'
E0C9 'AUTO'
E0CA 'RENUM'
E0CB 'DELETE'
E0CC 'EDIT'
E0CD 'RESUME'
E0CE 'ERL'
E0CF 'ELSE'
E0D0 'RUN'
E0D1 'LIST'
E0D2 'GOTO'
E0D3 'THEN'
E0D4 'GOSUB'
E0D5 Tabellens slutning
E0D6 '!'
E0DA '&'
E0DD '$'
E0F9 Token for linienummer
E105 Test for STRING

E108	Token for Floating Point variabel
E112	Token for 2-byte tal
E119	10
E11D	Adder Offset
E121	Token for 1-byte tal
E123	Skriv i buffer
E128	Skriv i buffer
E12F	Skriv i buffer
E134	Sammenlign HL <> DE
E145	Token for binært tal
E14B	Skriv i buffer
E152	Hent variabeltype
E158	Skriv i buffer
E161	'"'
E165	'I', kommandoudvidelse
E16B	'?'
E16D	Token for 'PRINT'
E172	Adresse på BASIC-operatorer
E187	'''
E18B	Skriv i buffer
E190	'"'
E19A	Skriv i buffer
E1A1	'"'
E1A5	Skriv i buffer
******	Bearbejdning af kommandoudvidelse
E1A8	Skriv i buffer
E1AB	Nul
E1AF	Skriv i buffer
E1B2	Næste tegn
E1B3	Forhøj pointer
E1B4	Test for bogstav eller tal
E1B7	Ja, så til buffer
E1BA	Formindsk pointer med 1
E1BC	Sæt bit 7 ved sidste tegn
E1C3	Skriv i buffer
E1C6	'''
E1C8	Skriv i buffer
E1CB	Skriv tegn til EOL
E1CE	i buffer
******	BASIC-kommando: LIST
E1D2	Hent linienummer-område
E1D7	Hent kanalnummer
E1DA	Slut på Statement, ellers 'SYNTAX ERROR'
E1DD	Sæt aktuelt linienummer til 0
E1E2	List linier
E1E5	Til READY-mode
******	List BASIC-linier BC - DE
E1E8	
E1E9	Linienummer til DE
E1EB	Søg BASIC-linie DE
E1F0	Programslutning ?
E1F5	Færdig !
E1F6	Break via ESC
E1FA	Adder linielængde
E201	Næste linienummer til DE
E205	Sammenlign HL <> DE
E209	Er sidste linienummer højere ?
E20B	List BASIC-linie i buffer
E20E	Pointer på buffer
E211	Udskriv tegn
E214	Forhøj pointer
E215	Næste tegn
E217	Endnu ikke slut ?
E219	Udskriv LF
E21F	List næste linie
E222	Output-kanal < 8 ?
E225	Load tegn
E226	Ja, skærm-output
E228	Udskriv tegn
E22B	LF
E22E	CR
E232	Kontrol-tegn ?
E234	Udskriv som almindeligt tegn
E23A	Udskriv tegn
E249	Pointer på buffer
E26B	Kommando-Token ?
E27C	Udskriv konstanter
E27E	'I', kommandoudvidelse
E290	'''
E294	'ELSE'
E299	':'
E29D	'"'
E2A8	'"'
E2CF	Skriv tegn i buffer
E2D0	Forhøj bufferpointer
******	List kommandoudvidelse
E2D6	
E2D8	Skriv i buffer
E2DC	næste tegn
E2DD	Forhøj pointer
E2DE	EOL ?
E2DF	Nej
E2E1	Slet bit 7
E2E3	Skriv i buffer
E2E6	Sidste tegn ?

E2E8	Nej, det næste tegn
E2ED	' '
E2EF	Skriv i buffer
E302	Funktion ?
E324	Test for bogstav eller tal
E337	Floating Point tal ?
E33F	Binært tal ?
E343	Hextal ?
E347	Linieadresse ?
E34B	Linienummer ?
E34F	2-byte tal ?
E356	1-byte tal ?
E35B	Tal ?
E376	'X'
E38D	'&'
E398	Variabeltype 'REAL'
E39A	Hent tal
E3AE	'A'
E3B3	Plus #E41D, adresse på kommando-ord
E3BF	26 bogstaver
E3C1	Kommando-ords tabel
E3CA	Næste bogstav
E3CC	Tabel for BASIS-operatorer
E3D2	'SYNTAX ERROR'
E3F6	TAB
E3FA	' '
******	Adresser på kommando-ord
E41D	A
E41F	B
E421	C
E423	D
E425	E
E427	F
E429	G
E42B	H
E42D	I
E42F	J
E431	K
E433	L
E435	M
E437	N
E439	O
E43B	P
E43D	Q
E43F	R
E441	S
E443	T
E445	U
E447	V
E449	W
E44B	X
E44D	Y
E44F	Z
******	Tabel over BASIC-kommandoer
******	Bogstav Z
E451	DA ZONE
******	Bogstav Y
E456	48 YPOS
******	Bogstav X
E45B	47 XPOS
E45F	FD XOR
******	Bogstav W
E463	D9 WRITE
E468	D8 WINDOW
E46E	D7 WIDTH
E473	D6 WHILE
E478	D5 WEND
E47C	D4 WAIT
******	Bogstav V
E481	7F VPOS
E485	1D VAL
******	Bogstav U
E489	ED USING
E48E	1C UPPER$
E494	1B UNT
******	Bogstav T
E498	D3 TRON
E49C	D2 TROFF
E4A1	EC TO
E4A4	46 TIME
E4A7	EB THEN
E4AB	7D TESTR
E4B0	7C TEST
E4B4	1A TAN
E4B7	D1 TAGOFF
E4BD	D0 TAG
E4C0	EA TAB
******	Bogstav S
E4C4	CF SYMBOL
E4CA	E7 SWAP
E4CE	7B STRING$
E4D5	19 STR$
E4D9	CE STOP
E4DD	E6 STEP
E4E1	18 SQR
E4E3	17 SQ

E4E6 CD SPEED
E4EB E5 SPC
E4EE 16 SPACE$
E4F3 CC SOUND
E4F9 15 SIN
E4FC 14 SGN
E4FF CB SAVE
****** Bogstav R
E504 CA RUN
E507 7A ROUND
E50C 45 RND
E50F 79 RIGHT$
E515 C9 RETURN
E51B C8 RESUME
E521 C7 RESTORE
E528 C6 RENUM
E52D 13 REMAIN
E533 C5 REM
E536 C4 RELEASE
E53D C3 READ
E541 C2 RANDOMIZE
E549 C1 RAD
****** Bogstav Q
****** Bogstav P
E54F BF PRINT
E554 78 POS
E557 BE POKE
E55B BD PLOTR
E560 BC PLOT
E564 44 PI
E566 BB PEN
E569 12 PEEK
E56D BA PAPER
****** Bogstav O
E573 B9 OUT
E576 B8 ORIGIN
E57C FC OR
E57E B7 OPENOUT
E585 B6 OPENIN
E58B B5 ON SQ
E598 B4 ON ERROR GOTO 0
E5A0 B3 ON BREAK
E5A8 B2 ON
****** Bogstav N
E5AB FE NOT
E5AE B1 NEW
E5B1 B0 NEXT
****** Bogstav M
E5B6 AF MOVER
E5BB AE MOVE
E5BF AD MODE
E5C3 FB MOD
E5C6 77 MIN
E5C9 AC MID$
E5CD AB MERGE
E5D2 AA MEMORY
E5D8 76 MAX
E5DB DF MASK
****** Bogstav L
E5E0 11 LOWER$
E5E6 10 LOG 10
E5EB 0F LOG
E5EE A9 LOCATE
E5F4 A8 LOAD
E5F8 A7 LIST
E5FC A6 LINE
E600 A5 LET
E603 0E LEN
E606 75 LEFT$
****** Bogstav K
E60C A4 KEY
****** Bogstav J
E610 0D JOY
****** Bogstav I
E614 0C INT
E617 74 INSTR
E61C A3 INPUT
E621 0B INP
E624 43 INKEY$
E62A 0A INKEY
E62F A2 INK
E632 A1 IF
****** Bogstav H
E635 42 HIMEM
E63A 73 HEX$
****** Bogstav G
E63F DE GRAPHICS
E647 A0 GO TO
E64C 9F GO SUB
****** Bogstav F
E653 09 FRE
E656 E0 FRAME
E65B 9E FOR
E65E E4 FN
E660 08 FIX
E663 DD FILL
****** Bogstav E
E668 07 EXP

E66B 9D EVERY
E670 9C ERROR
E675 41 ERR
E678 E3 ERL
E67B 9B ERASE
E680 40 EOF
E683 9A ENV
E686 99 ENT
E689 98 END
E68C 97 ELSE
E690 DC EI
E692 96 EDIT
****** Bogstav D
E697 95 DRAWR
E69C 94 DRAW
E6A0 93 DIM
E6A3 DB DI
E6A5 49 DERR
E6A9 92 DELETE
E6AF 91 DEG
E6B2 90 DEFSTR
E6B8 8F DEFREAL
E6BF 8E DEFINT
E6C5 8D DEF
E6C8 72 DEC$
E6CC 8C DATA
****** Bogstav C
E6D1 E1 CURSOR
E6D7 06 CREAL
E6DC 05 COS
E6DF 7E COPYCHR$
E6E7 8B CONT
E6EB 8A CLS
E6EE 89 CLOSEOUT
E6F6 88 CLOSEIN
E6FD 87 CLG
E700 86 CLEAR
E705 04 CINT
E709 03 CHR$
E70D 85 CHAIN
E712 84 CAT
E715 83 CALL
****** Bogstav B
E71A 82 BORDER
E720 71 BIN$
****** Bogstav A
E725 81 AUTO
E729 02 ATN
E72C 01 ASC

E72F FA AND
E732 80 AFTER
E737 00 ABS
****** BASIC-operatorer og tilhørende Tokens
E73B F8 '↑'
E73D F9 'BACKSLASH'
E740 F0 '>='
E743 F0 '=>'
E747 EE '>'
E749 F2 '<>'
E74D F3 '<='
E751 F3 '=<'
E755 EF '='
E757 F1 '<'
E759 F7 '/'
E75B 01 ':'
E75D F6 '*'
E75F F5 '-'
E761 F4 '+'
E763 C0 '"'
****** Slet programpointer
E766
E767 Programstart
E76D 3 gange 0 til programslut
E770 Programslut
E77D Indsæt linienummer
****** Erstat linieadresse med linienummer
E78B Hent liniens næste element
E78E Slut på Statement ?
E790 Ja
E791 'Linieadresse' ?
E793 Nej
E79F Linienummer til DE
E7A2 'Linienummer'
E7A6 indsættes
****** BASIC-kommando: DELETE
E7F3
E7F6 Slut på Statement, ellers 'SYNTAX ERROR'
E802 til READY-mode
E805 Hent linienummer-område
E80A Søg BASIC-linie DE
E80F Søg BASIC-linie DE
****** Hent linieadresse
E82C
E82E Nummer eller adresse til DE
E830 'Linieadresse' ?

E832 Ja
E834 'Linienummer' ?
E836 'SYNTAX ERROR'
E83A Hent linienummer til HL
E83D Sammenlign HL <> DE
E840 hvis mindre, søg fra programstart
E845 Spring resten af linien over
E848 fra adresse (HL)
E849 Søg BASIC-linie DE
E84C hvis ikke fundet, søg fra programstart
E854 'Linieadresse'
E859 i programmet
E85B Linieadresse i programmet.
****** Søg BASIC-linie DE
E861
E865 Udskriv fejlmeddelelse
E868 'LINIE DOES NOT EXIST'
E869 Programstart
E86E Linielængde til BC
E872 Programslut ?
E873 Ikke fundet
E878 Linienummer til HL
E87C Sammenlign HL <> DE
E882 hvis større, ikke fundet
E883 ens, fundet
E884 Adder linielængde
E885 søg videre
****** Søg BASIC-linie DE
E887 Programstart
E88B Gem linieadresse
E88D Flyt liniens længde til BC
E891 Programslut ?
E893 Ja
E896 Linienummer til HL
E89A Sammenlign HL <> DE
E89F Er aktuelt linienummer større eller lig med ?
E8A0 Adder linielængde
E8A1 Søg videre
****** BASIC-kommando: RENUM
E8A3 10, Deafult for startværdi
E8A6 Hent linienummer til DE
E8AA 0, Default
E8AD Følger komma ?
E8B0 Hent linienummer til DE
E8B4 10, Default
E8B7 Følger komma ?
E8BD EOL, ellers 'SYNTAX ERROR'
E8C6 Søg BASIC-linie DE
E8CB Søg BASIC-linie DE
E8D0 Sammenlign HL <> DE
E8D3 'IMPROPER ARGUMENT'
E8F2 'IMPROPER ARGUMENT'
E967 'IF'
E974 'ELSE'
E980 '['
E984 '('
E98D '['
E991 '('
E995 ']'
E999 ')'
E9A1 'SYNTAX ERROR'
****** BASIC-kommando: DATA
E9A8
****** BASIC-kommando: REM og '
E9AC
****** BASIC-kommando: ELSE
E9B2
E9C2 Programstart
E9D1 Hop til (BC)
E9EC Udskriv fejlmeddelelse
E9FA 'ELSE'
E9FD 'THEN'
EA02 Spring over 'BLANKS'
EA0E '"'
EA12 'I'
EA16 ':'
EA1A 'REM'
EA1E Funktion
EA25 '"'
EA2C '"'
****** BASIC-kommando: RUN
EA7D Slut på Statement ?
EA80 Programstart som default
EA84 Ja
EA86 Linienummer ?
EA88 Ja
EA8A Linieadresse ?
EA94 MC BOOT PROGRAM
EA9A Programstart
EA9F Hent linieadresse
EAB7 til fortolker-løkke
****** BASIC-kommando: LOAD
EABA
EAC2 Til READY-mode

EACC DISK IN DIRECT
EAD6 Hent navne, åben fil
EAD9 Filtype
EAE7 følger komma ?
EAEA Ja, hent 16-bits værdi
EAED som startadresse
EAF1 Slut på statement, ellers 'SYNTAX ERROR'
EAF6 Startadresse
EAF9 DISK IN DIRECT
EAFD DISK IN CLOSE
****** BASIC-kommando: CHAIN
EB02 'MERGE'
EB04 Flag for MERGE
EB07 Skip BLANKS
EB0D Default-værdi NULL for startlinie
EB10 Følger komma ?
EB13 Nej
EB16 ','
EB18 Hent 16-bits værdi
EB1C Følger komma?
EB1F Nej
EB21 Test på efterfølgende tegn
EB24 'DELETE'
EB25 Slet linieområde
EB2A Slut på statement, ellers 'SYNTAX ERROR'
EB30 Garbage Collection
EB3D Hent startlinie
EB3E Programstart som default
EB42 Ingen startlinie
EB4C Flag for MERGE
****** BASIC-kommando: MERGE
EB59 Hent navne, åben fil
EB5C Slut på Statement, ellers 'SYNTAX ERROR'
EB5F Slet variabler
EB62 Test filtype
EB65 Til READY-mode
EB71 Programslut
EB75 Programstart
EB79 Programslut
EB7D BD := HL-DE
EBA5 Sammenlign HL <> DE
EBBA Programslut
EBCD Sammenlign HL <> DE
EBE9 Programslut
EBF1 Programslut
EBFD 'MEMORY FULL'
EBFF Udskriv fejlmeddelelse
EC01 DISK IN CHAR
EC05 CTRL Z
EC09 Diskettefejl
EC0E Diskettefejl
EC14 'EOF MET'
EC1B Udskriv fejlmeddelelse
EC24 Programslut
EC2A Programslut
EC32 Programslut
EC4B Programslut
EC67 Filtype
EC6E ASCII-fil ?
EC70 Nej
EC72 Filtype
EC75 ASCII-fil
EC77 Ja
EC79 Slet bit 0 (beskyttet fil)
EC7D Udskriv fejlmeddelelse
EC80 'FILE TYPE ERROR'
EC87 Programstart
EC9A Sammenlign HL <> DE
ECA2 Programslut
ECA5 Filtype
ECA8 Test bit 0
ECAA Sæt flag for beskyttet fil
ECAF DISK IN DIRECT
ECB2 'EOF MET'
ECD8 'DIRECT COMMAND FOUND'
ECDC 'OVERFLOW'
ECDE Udskriv fejlmeddelelse
****** BASIC-kommando: SAVE
ECE1
ECE4 OPENOUT
ECE7 Filtype 0, BASIC-program
ECE9 Følger komma?
ECEC Nej
ECEE Test på efterfølgende tegn
ECF1 Normal variabel
ECF4 Variabelnavn
ECF5 Konverter små bogstaver til store bogstaver
ECF7 'SYNTAX ERROR'
ECFB Tabellens basisadresse
ECFE Gennemsøg tabellen
ED01 Flyt adresse fra tabel til STACK

ED02	Skip BLANKS
ED05	Antallet af filer
ED06	ikke fundet, 'SYNTAX ERROR'
ED08	'A'
ED0B	'B'
ED0E	'P'
******	SAVE, P
ED11	Filtype 1, protected
ED13	Slut på Statement, ellers 'SYNTAX ERROR'
ED1E	Programstart
ED23	Programslut
ED27	HL := HL - DE
******	SAVE, B
ED30	Test for ','
ED33	Hent 16-bits værdi
ED36	Gem den
ED37	Test for ','
ED3A	Hent 16-bits værdi
ED3D	Gem den
ED3E	Følger der komma ?
ED41	0, default for tilgangsadresse
ED44	Ja, hent 16-bits værdi
ED47	Gem den
ED48	Slut på Statement, ellers 'SYNTAX ERROR'
ED4B	Filtype 2, binær
ED4D	Tilgangsadresse
ED4E	Slutadresse
ED4F	Startadresse
ED50	DISK OUT DIRECT
ED53	BREAK via ESC
ED56	CLOSEOUT
******	SAVE, A
ED58	Slut på Statement, ellers 'SYNTAX ERROR'
ED5C	9
ED5E	Output på kanal 9, Diskette
ED62	1 til
ED65	65535
ED68	linier listes
ED6C	Sæt output til default igen
ED6F	CLOSEOUT
ED79	Slet BLANK, TAB og LF
ED7E	'&'
ED82	Test for numerisk
ED87	type for integer
ED8A	Slet variabel
ED9A	'&'
EDA3	Flyt integer-tal til HL
EDB0	'.'
EDB2	Slet BLANK, TAB og LF
EDB5	Test på tal
EDBC	'.'
EDC1	Type for Integer
EDCE	'.'
EDD5	Type for Real
EDEF	Test for streng
EDF2	Nej
EDFF	4-byte integer*256 til Floating Point
EE03	Multiplicer tallet med 10↑A
EE07	Sæt variabeltype til flydende komma
EE0A	Kopier variabel fra DE til HL
EE14	Slet BLANK, TAB og LF
EE18	-1
EE1A	'-'
EE1D	0
EE1E	'+'
EE24	Slet BLANK, TAB og LF
EE28	Test for tal
EE2E	Konverter små bogstaver til store bogstaver
EE33	'0'
EE47	'E'
EE52	Slet BLANK, TAB og LF
EE55	Test for tal
EE60	Sæt type til Real
EE6B	100
EE99	Skip BLANK, TAB og LF
EEC0	gange 10
EEC4	plus det næste ciffer
EEDA	Flyt integertallet til HL
EEED	Skip BLANK, TAB og LF
EEF0	Konverter små bogstaver til store bogstaver
EEF3	Basis 2, binær
EEF5	'X'
EEF9	basis 16, Hex
EEFB	'H'
EF00	Skip BLANK, TAB og LF
EF05	Basis 10, Decimalt
EF08	Konverter hextal til binært
EF12	Konverter hextal til binært
EF1C	Talsystemets basis

EF1D Integer-multiplikation uden fortegn
****** Konverter Hex-tal til binært
EF31 Hent tegn
EF32 Forhøj pointeren
EF33 Test for tal
EF36 Ja
EF38 Konverter små bogstaver til store bogstaver
EF3B 'A'
EF3E Lille 'A', Fejl
EF40 'A'-('9'+1)
EF42 '0'
EF45 Ingen fejl
EF47 Slet carry
****** Udskriv integertal HL
EF49 Konverter integertal til ASCII
EF4C Udskriv streng
****** Konverter integertal til ASCII
EF4F
EF51 Flyt integertal til HL
EF61 Nul
EF62 Konverter tal til formateret streng
EF68 ' '
EF98 '%'
F02F '.'
F034 'I'
F03D '+E'
F047 '-'
F04C '0'-1
F04F 10
F053 '9'+1
F079 ','
F099 '0'
F0A8 '5'
F0B1 'I'
F0C0 '9'
F0C3 '0'
F0DA '0'
F0E8 '0'
F128 ','
F135 '0'
F146 '0'
F156 '-'
F162 '+'
F166 ' '
F181 '*'
F185 ' '
F1CF '0'
F1DE '0'
****** BASIC-funktion: PEEK
F20D UNT
F210 READ RAM, LD A,(HL)
F211 Overtag AKK-indhold som integertal
****** BASIC-kommando: POKE
F214 Hent 16-bits adresse
F218 Test komma og hent 8-bits værdi
F21C Skriv værdien i adressen
****** BASIC-funktion: INP
F21E CINT
F221 Flyt portadresse til BC
F223 Læs port
F225 Overtag indeholdet i AKK som integerværdi
****** BASIC-kommando: OUT
F228 Hent adresse og værdi
F22B Udskriv
****** BASIC-kommando: OUT
F22E Hent adresse og værdi
F231 Flyt 8-bits værdi til D
F232 3. parameter Nul
F234 Hent evt. 3. parameter
F237 Flyt 3. parameter til E
F238 Læs porten
F23B Kombiner
F23C og vent
****** Hent 16-bits/8-bits værdi
F23F Hent 16-bits værdi
F243 og flyt det til BC
F244 Test for ','
F247 og hent 8-bits værdi
****** Kommandoudvidelse
F24A Forhøj programpointeren
F24C Følger der en Nul-byte ?
F24F Ja, KL FIND COMMAND
F252 Flyt kommandoadresse til DE
F254 Hvis ikke fundet, 'UNKNOWN COMMAND'
F256 Hent tegn
F257 Skip kommando-ord
F258 Er bit 7 sat ?
F259 Nej, læs videre
F25B Til CALL-kommando
F25D Udskriv fejlmeddelelse
F260 'UNKNOWN COMMAND'

******	BASIC-kommando: CALL
F261	Hent 16-bits værdi
F264	#FF = RAM valgt
F266	Adresse til #AE55
F26B	Flyt konfigurationsbyte til #AE57
F26E	Gem STACK-pointer
F272	Maksimalt 32 parametre
F274	følger komma ?
F277	Nej
F27A	Hent udtryk og
F27E	flyt adresse til STACK
F27F	Næste parameter
F281	Slut på Statement, ellers 'SYNTAX ERROR'
F284	Gem HL
F289	Antallet af parametre i AKK
F28E	Flyt parameterblokkens adresse til IX
F290	Udfør rutine
F293	Hent STACK-pointeren tilbage
F297	Initialiser descriptor-blok
F29A	Hent HL tilbage
******	BASIC-kommando: ZONE
F2A2	Hent 8-bits værdi, forskellig fra 0
F2A5	som tabulatorvidde
******	BASIC-kommando: PRINT
F2A9	Kanalnummer
F2AC	Slut på Statement ?
F2AF	Ja, udskriv LF
F2B2	'USING'
F2B8	Tabellens basisadresse
F2BB	Gennemsøg tabellen
F2BF	JP (DE), udfør funktionen
F2C2	Slut på Statement ?
F2C5	Nej, fortsæt
F2C8	Antallet af tabel-elementer
F2C9	Returadresse, hvis tallet ikke findes
F2CB	','
F2CE	'SPC'
F2D1	'TAB'
F2D4	';'
F2D5	Skip BLANKS
******	PRINT
F2D7	Hent udtrykket
F2DC	Test for streng
F2DF	Ja
F2E1	Konverter tallet til en ASCII-streng
F2E4	Hent streng-parameter
F2E7	Efterstil ' ' blank
F2E9	Hent stringdescriptor
F2EC	Forhøj længden
F2F0	Hent stringdescriptor
F2F3	Længde
F2FF	' '
F302	Kontroltegn ?
F30F	Valg af output
******	PRINT
F31E	Skip BLANKS
F321	Tabulatorvidde
******	PRINT SPC
F339	Hent 8-bits værdi i parantes
******	PRINT TAB
F342	Hent 8-bits værdi i parantes
******	Hent 8-bits værdi i parantes
F362	Skip BLANKS
F365	Test for '('
F368	Hent 8-bits værdi
F36B	Test for ')'
******	PRINT USING
F383	Skip BLANKS
F386	Hent strengudtryk
F389	Test for efterfølgende tegn
F38C	';'
F392	Hent udtryk
F3A9	Slut på Statement ?
F3AC	Ja
F3B4	Hent udtryk
F3D7	'UNDERLINE'
F3F4	';'
F3F6	Skip BLANKS
F3F9	Test for ','
F3FF	'&'
F404	'!'
F408	'BACKSLASH'
F413	'BACKSLASH'
F417	' '
F436	Test for formateringstegn
F443	Formater tal
F446	Udskriv streng
******	Test for formateringstegn
F44D	
F454	'+'
F460	'.'
F464	'#'

F47A ' '
F47C '*'
F489 '#'
F49C '$'
F4B0 'IMPROPER ARGUMENT'
F4B8 '.'
F4BC '#'
F4C0 ','
F4D0 '#'
F4D6 '↑'
F4F9 '-'
F4FD '+'
F507 '$'
****** BASIC-kommando: WRITE
F50D
F510 Slut på Statement ?
F513 Ja
F516 Hent udstryk
F51B Test for streng
F51E Ja
F520 Konverter tal til ASCII
F523 og udskriv
F528 '"'
F52A udskrives
F52D Udskriv streng
F530 '"'
F532 udskrives
F537 Slut på Statement ?
F53D ','
F53F udskrives
F542 Fortsæt
****** Konfigurer hukommelsen
F544 Plads fra DE til HL
F547 Sammenlign HL med BC
F54A Er højeste adresse < #AC00 ?
F54B HIMEM
F54E Slut på strengen
F551 Slut på den frie RAM
F555 Start på den frie RAM
F558 Plus 303
F55D angiver programstarten
****** BASIC-kommando: MEMORY
F570 Hent 16-bits værdi
F577 Sammenlign HL <> DE
******
F58F TXT GET M TABLE
F5F7 Sammenlign HL <> DE
****** Beregn strengområdets størrelse
F5FD
F5FF Start på streng
F603 Slut på streng
F606 BC := HL - DE
****** Forhøj program- og variabelpointer
F60C Programslut
F610 Programslut
F618 Variabelstart
F61C Variabelstart
F61F Arraystart
F623 Arraystart
F626 Arrayslut
F62A Arrayslut
F633 Programslut
F63E Programslut
F645 BC := HL - DE
****** Initialisering af BASIC-stack
F652 Start på STACK
F655 BASIC-stackpointer
F658 Reserver plads i BASIC-stack
F65A til 1 byte og gem den
F65D Flyt Nul til STACK
F65F Forhøj stackpointeren
F660 og gem den
****** Frigiv plads i BASIC-stack
F665 Stackpointer
F669 Fratræk AKK-indhold
F671 Gem BASIC-stackpointerens nye værdi
****** Reserver plads i stackpointeren
F675 BASIC-stackpointer
F67A Adder indeholdet i AKK
F67E BASIC-stackpointer
F683 Giver plus #4F94 overflow ?
F686 Så er stackpointeren > #B06C
F689 Initialiser BASIC-stack
F68C 'MEMORY FULL'
F68F Slut på strengen
F692 Start på strengen
****** Reserver plads til strengen
F696
F69C Start på strengen
F6A4 Sammenlign HL <> DE
F6AE Udskriv fejlmeddelelse

F6B1 'STRING SPACE FULL'
F6B2 Start på strengen
F6BF Programslut
F6D2 Sammenlign HL <> DE
F6EA Blocktransfer LDDR
F6F9 BC := HL - DE
F6FE Blocktransfer LDIR
F705 BC := HL - D
F70C Start på streng
F717 Slut på streng
****** BASIC-kommando: SYMBOL
F784 'AFTER'
F788 Hent 8-bits værdi
F78C Test for ','
F78F 8 værdier
F792 følger komma ?
F796 Ja, hent 8-bits værdi
F79B Allerede 8 argumenter ?
F79F TXT GET MATRIX
F7A2 Matrix ikke i RAM, 'IMPROPER ARGUMENT'
F7A5 8
F7A8 plus matrixadresse
F7A9 Byte fra STACK
F7AB i matrixtabellen
F7AD Næste byte
****** SYMBOL AFTER
F7B1 Skip BLANKS
F7B4 Hent integerværdi med fortegn
F7B8 256
F7BB Sammenlign HL <> DE
F7BE 'IMPROPER ARGUMENT'
F7C2 TXT GET M TABLE
F7C6 Er matrix endnu ikke defineret ?
F7D3 'IMPROPER ARGUMENT'
F7DD 256
F7E0 TXT SET M TABLE
F7FD 'MEMORY FULL'
F805 TXT SET M TABLE
F815 Sammenlign HL <> DE
F818 'MEMORY FULL'
F833 Slut på strengen
F83E Blocktransfer LDDR
F844 Start på streng
F851 Blocktransfer LDIR
F857 Slut på streng
F85B Start på streng
F865 Programslut
F868 Sammenlign HL <> DE
F875 Udskriv fejlmeddelelse
F878 "MEMORY FULL'
****** Læs streng
F879
F87E '"'
F880 Skip BLANKS
F89F ','
F8AD JP (DE)
F8BE ' '
F8C2 TAB
F8C6 CR
F8CA LF
****** Udskriv streng
F8D0 Hent strengparametre
F8D3 Tom streng ?
F8D4 Hent tegn
F8D5 Forhøj pointer
F8D6 Udskriv tegn
F8D9 Næste tegn
****** BASIC-funktion: LOWER$
F8EC Konverter store bogstaver til små bogstaver
****** Konverter store bogstaver til små bogstaver
F8F1 'A'
F8F4 'Z'+1
F8F7 'a'-'A'
****** BASIC-funktion: UPPER$
F8FA Konverter små bogstaver til store bogstaver
F915 Hop til (BC)
****** Streng-addition
F91D Flyt pointer til 2. streng
F921 Adder
F922 længden
F923 Ingen overflow
F925 Udskriv fejlmeddelelse
F928 'STRING TOO LONG'
****** BASIC-funktion: BIN$
F964
****** BASIC-funktion: HEX$
F969
F96D Hent udtryk
F975 Følger komma ?
F978 0 som default-værdi
F979 Ja, hent 8-bits værdi
F97C Større/lig 17 ?

```
F97E   Ja, 'IMPROPER
       ARGUMENT'
F982   Test for ')'
F98A   Konverter tal til streng
****** BASIC-funktion: DEC$
F98F   Hent udtryk
F992   Test for ','
F995   Lægges på BASIC-stack
F998   Hent strengudtryk og
       parametre
F99B   Test for ')'
F99F   Længde
F9A0   Frigiv plads i BASIC-stack
F9A4   Længde
F9A5   Overtag variabel
F9AB   og test for formateringstegn
F9AE   'IMPROPER ARGUMENT'
F9B3   'IMPROPER ARGUMENT'
F9B7   Formater tal
F9BA   Overtag streng
****** BASIC-funktion STR$
F9BC
F9BD   Konverter tal til streng
F9C1   Sæt tæller for strengens
       længde til -1
F9C3   Nul
F9C4   Forhøj tælleren
F9C5   Slut på streng ?
F9C6   Forhøj pointeren
F9C7   Nej, næste tegn
F9CA   Strenglængde
F9CB   Reserver plads, opbyg
       strengdescriptor
****** BASIC-funktion: LEFT$
F9D3   Hent streng og 8-bits værdi
****** BASIC-funktion: RIGHT$
F9D8   Hent streng og 8-bits værdi
F9DB   Strenglængde
F9DC   minus parametre
****** BASIC-funktion: MID$
F9E2   Test for '('
F9E5   Hent streng og 8-bits værdi
F9E8   Nul, 'IMPROPER
       ARGUMENT'
F9EB   255
F9EC   som default
F9ED   Hent det 3. argument
F9F0   Test for ')'
****** BASIC-kommando: MID$
FA07   Test for '('
FA0A   Hent variabel
FA0D   Type = streng, ellers 'TYPE
       MISMATCH'
FA19   'IMPROPER ARGUMENT'
FA1C   255
FA1D   som default
FA1E   Hent det 3. argument
FA21   og test for ')'
FA24   Test for '='
FA28   Hent strengudtryk og -
       parameter
FA3E   Blocktransfer LDIR
FA43   Hent strengudtryk
****** Hent det 3. argument for
       MID$
FA4F   Default 255
FA52   ')'
FA56   Test for ','
FA59   Hent 8-bits værdi
****** BASIC-funktion: LEN
FA69   Hent strengparametre, flyt
       længde til AKK
FA6C   Overtag indholdet i AKK som
       integerværdi
****** BASIC-funktion ASC
FA6E   Første tegns ASCII-kode
FA71   Overtag indholdet i AKK som
       integerværdi
****** BASIC-funktion CHR$
FA74   CINT, <256
FA77   ASCII-kode i AKK
FA7A   Længde 1
FA7C   Anlæg streng med længden
       AKK
****** BASIC-funktion INKEY$
FA7E   KM READ CHR
FA81   Ingen tast trykket ?
FA83   'ESC'
FA85   Tom streng
FA87   'ESC'
FA89   Tom streng
FA8B   Overtag tegn i streng
****** BASIC-funktion STRING$
FA8D   Hent 8-bits værdi, længde
FA91   Test for ','
FA94   Hent udtryk
FA97   TEst for ')'
FA9F   Anlæg streng med længden A
```

FAA1 Test for streng
FAA4 Nej
FAA6 hent strengparametre
FAA9 Tom streng, 'IMPROPER ARGUMENT'
FAAB Hent ASCII-kode
****** BASIC-funktion: SPACE$
FAAD
FAB0 ' '
****** BASIC-funktion: VAL
FABE Hent strengparameter
FAC1 Overtag indholdet i AKK som integerværdi
FACE Konverter streng til værdi
FAD5 Udskriv fejlmeddelelse
FAD8 'TYPE MISMATCH'
FAE2 'IMPROPER ARGUMENT'
****** BASIC-funktion: INSTR
FAE5 Hent udtryk
FAE8 Test for streng
FAEB Default startposition 1
FAED Ja
FAEF CINT, < 256
FAF3 'IMPROPER ARGUMENT'
FAF7 Test for ','
FAFA Hent strengudtryk
FAFD Test for ','
FB05 Hent strengudtryk og -parameter
FB08 Test for ')'
FB48 Overtag indholdet i AKK som integerværdi
FB65 Start på streng
FB68 Sammenlign HL <> BC
FB6D Slut på streng
FB70 Sammenlign HL <> BC
FB9E Programstart
FBA1 Sammenlign HL <> DE
FBA4 Slut på streng
FBA8 Sammenlign HL <> DE
FBAD Programslut
FBB1 Sammenlign HL <> DE
FBC4 Blocktransfer LDIR
****** Initialisering af descriptorstack
FBCC
FBCF Pointer i descriptorstack for strenge
FBD7 Streng
FBD9 som variabeltype
FBDC Pointer i descriptorstack
FBE2 Strengdescriptor
FBE5 Sammenlign HL <> DE
FBE8 'STRING EXPRESSION TOO COMPLEX'
FBEA Udskriv fejlmeddelelse
FBF0 Pointer i descriptorstack
FBF6 Type streng, ellers 'TYPE MISMATCH'
FC0A Start på streng
FC10 Sammenlign HL <> DE
FC1B Start på streng
FC27 Sammenlign HL <> DE
****** BASIC-funktion: FRE
FC53 Test for streng
FC56 Nej
FC5B Garbage Collection
FC5E Beregn fri RAM
****** Garbage Collection
FC64
FC7C Sammenlign HL <> DE
FC87 Slut på streng
FC8B Start på streng
FCA9 Sammenlign HL <> BC
FCAF Start på streng
FCB3 BC := HL - DE
FCB7 Sammenlign HL <> DE
FCBC Blocktransfer LDDR
FCC0 Start på streng
FCD9 Sammenlign HL <> DE
FCE6 Sammenlign HL <> DE
FCF3 Hent det numeriske resultat
FD03 UNT
****** BASIC-operator '+'
FD0C Test operand-type
FD0F Flydende komma ?
FD11 Integer-addition HL := HL + DE
FD14 Ikke overflow, overtag resultat i HL
FD17 konverter til flydende komma tal
FD1A Flydende komma-addition
FD1D Ikke overflow, så ok
FD1E 'OVERFLOW'
****** BASIC-operand '-'
FD21 Test operand-type
FD24 Flydende komma ?

FD26 Integer-subtraktion HL := DE - HL
FD29 Ikke overflow, overtag resultat i HL
FD2C konverter til flydende komma-tal
FD2F Flydende komma-subtraktion
FD32 Ikke overflow, ok
FD33 'OVERFLOW'
****** BASIC-operand '*'
FD35 Test operand-type
FD38 Flydende komma ?
FD3A Integer-multiplikation med fortegn
FD3D Ikke overflow, overtag resultat i HL
FD40 konverter til flydende komma-tal
FD43 Flydende komma-multiplikation
FD46 Ikke overflow, ok
FD47 'OVERFLOW'
****** Aritmetisk sammenligning
FD49 Test operand-type
FD4C Integersammenligning
FD4F Flydende komma-sammenligning
****** BASIC-operand '/'
FD52
FD57 Flydende komma division
FD5B 5 bytes
FD5E overfør resultat
FD60 ok ?
FD61 'DIVISION BY ZERO'
FD64 'OVERFLOW'
****** BASIC-operand 'BACKSLASH'
FD67
FD6B Integer division med fortegn
FD6E Overflyt resultat til HL
FD71 'DIVISION BY ZERO'
****** BASIC-operand 'MOD'
FD79
FD7D MOD-beregning
FD80 Overflyt resultat til HL
FD83 Udskriv fejlmeddelelse
FD86 'DIVISION BY ZERO'
****** BASIC-operand 'AND'
FD87
FD8C HL := HL AND DE
FD8F Overtag integertal HL
****** BASIC-operand 'OR'
FD92
FD97 HL := HL ORD DE
FD9A Overtag integertal HL
****** BASIC-operand 'XOR'
FD9C
FDA1 HL := HL XOR DE
FDA4 Overtag integertal HL
****** BASIC-operand 'NOT'
FDA6 CINT
FDAC Komplementer HL
FDAE Overtag integertal HL
****** BASIC-funktion: ABS
FDB0 SGN
FDB3 Postivt fortegn
****** Skift fortegn
FDB4 Hent numerisk resultat
FDB7 Fortegnsskift, flydende komma
FDBA Fortegnsskift, integer
FDBD Gem resultat
FDC0 Overflow, konverter tal efter flydende komma
****** Bestemmelse af fortegn
FDC4
FDC6 SGN
****** Bestemmelse af fortegn
FDCC Hent numerisk resultat
FDCF Integer SGN
FDD2 SGN
****** Afrund tal
FDD5
FDD7 Overtag variabeltype- og værdi
FDDB Hent numerisk resultat
FDDE Afrundingspositioner ?
FDDF Flydende komma værdi ?
FDE2 Afrunding efter komma ?
FDE3 Konverter integerværdi til flydende komma værdi
FDE6 Afrund tallet
FDE9 CINT
FDEC Afrundingspositioner
FDED Hvis forskellig fra 0, så afrund
FDEF Konverter flydende komma til integer
FDF2 Udfør funktion
FDF6 Afrundingspositioner

FDF7 Multiplicer flydende komma tal med 10↑A
FDFA Konverter flydende komma til integer
FE02 Konverter integer til flydende komma værdi
FE05 Inverter afrundingspositioner
FE06 svarende til division
FE07 Multiplicer flydende komma tal med 10↑A
****** BASIC-funktion: FIX
FE0E FIX-funktion
****** BASIC-funktion: INT
FE13 INT-funktion
FE16 Hent numerisk resultat
FE19 Integer ?
FE1A JP (DE), udfør funktion
FE1E Variabeltype
FE25 Variabeltype til C, pointer til HL
FE29 Konverter integer til flydende komma værdi
FE2D Streng ?
FE34 Hvis positiv, overtag fortegn fra B
FE38 Overtag resultat i HL
FE3C Streng ?
FE3E Ja, 'TYPE MISMATCH'
FE40 Variabeltype
FE43 Streng ?
FE45 Ja, 'TYPE MISMATCH'
FE4D Variabeltype
FE50 gemmes
FE52 Konverter integertal til flydende komma værdi
FE58 Konverter integertal til flydende komma værdi
FE5D Pointer på variabel
FE68 Variabel
FE6D 'TYPE MISMATCH'
FE70 Variabeltype
FE73 sammenlignes
FE74 Integer ?
FE76 Nej
****** Integer-operander til flydende komma
FE78 Første operand
FE7B konverteres
FE7E BASIC-stackpointer, 2. operand
FE81 konverteres
FE84 og flyttes til DE
****** Konvertering af integertal til flydende komma værdier
FE8D Flyt tal til DE
****** Konvertering af integertal til flydende komma værdier
FE95 Variabeltype
FEl8 til 'REAL'
FE9E Hvis negativ, så skift integer-fortegn
FEA2 Konvering af integertal til flydende komma værdier
****** Konvertering af 4-bytes tal til flydende komma
FEA5 Lo-word
FEA9 Hi-word
FEAC Variabeltype
FEAF Real
FEB1 Flyt pointer til 4-byte værdi
FEB3 Konvertering af tal til flydende komma
****** BASIC-funktion: CINT
FEB6
FEBA 'OVERFLOW'
FEBF Resultat
FECC Overflow
FECE Flyt pointer til variabeltype
FED1 Load variabeltype
FED2 Sammenlign integertype med streng
FED5
FED9 'TYPE MISMATCH'
FEDD Konverter flydende komma værdi til integertal
FEE1 Flyt fortegn B til HL-integertal
****** Flyt integerværdi (HL) til (HL)
FEE6
****** BASIC-funktion: UNT
FEEB Hent numerisk resultat
FEEE Integer ?
FEEF Konverter flydende komma til integer
FEF2 'OVERFLOW'
FEF5 Overfør fortegn B til integertal
FEF8 Overfør integer i HL

FEFB Udskriv fejlmeddelelse
FEFE 'OVERFLOW'
FF02 Variabeltyper
FF05 sammenlignes
FF06 Forskellige ?
FF0F CINT
FF11 Type = streng, ellers 'TYPE MISMATCH'
****** BASIC-funktion: CREAL
FF14 Hent numerisk resultat
FF17 Hvis integer, så konverter
****** Sæt flydende komma til 0
FF1B
****** BASIC-funktion: SGN
FF2A SGN
****** Overfør indhold i AKK som integer
FF32 Lo-byte
FF33 slet Hi-byte
****** Overfør integer i HL
FF35 Gem værdien
FF38 Type for integer
FF3A og gem
****** Variabeltype for flydende komma
FF3E Pointer på flydende komma
FF41 Type for REAL
****** Hent variabeltype, HL pointer på variablen
FF45 Pointer på variabel
FF48 type til C
FF49 HL pointer på variabel
****** Hent variabeltype
FF4B Variabeltype i AKK
****** Hent numerisk resultat
FF4F Variabeltype
FF52 Streng ?
FF54 Ja, 'TYPE MISMATCH'
FF56 Load integerværdi
FF59 Ingen flydende komma, færdig
FF5A Adresse på flydende komma tal
FF5E Test for streng
FF61 Ja, ok
FF62 Udskriv fejlmeddelelse
FF65 'TYPE MISMATCH'
****** Test for streng
FF66 Variabeltype
FF69 streng ?
FF6C Sæt variabeltype
FF6F Flyt adressen til DE
****** Læg resultatet i BASIC-stack
FF74
FF76 Variabeltype
FF79 svarende til stack
FF7A Reserver plads i BASIC-stack
FF7D Læg resultatet i STACK
****** Kopier variablen til (HL)
FF83 Flyt måladressen til DE
FF84 Kildeadresse
FF88 Variabeltype
FF8B svarende til forskydningstæller
FF8C Slet Hi-byte
FF8E Forskyd
****** Test for bogstaver
FF92 Konverter små bogstaver til store bogstaver
FF95 'A'
FF99 'Z'+1
****** Test for alfanumerisk tegn
FF9C Test for bogstav
FF9F ja
FFA0 '.'
FFA4 '0'
FFA8 '9'+1
****** Konverter små bogstaver til store bogstaver
FFAB 'a'
FFAE 'z'+1
FFB1 'a'-'A'
****** Gennemsøg efterfølgende tabel
FFB4
FFB6 Load tabellængde
FFB8 Returadresse ved negativ søgning
FFBA Flyt pointer til næste tabel-element
FFBB Sammenlign tegn
FFBC Forhøj pointer
FFBD Fundet ?
FFBF Er tabellen endnu ikke gennemsøgt ?
FFC1 Load returadresse
FFC5 Flyt adresse til HL
****** Gennemsøg hukommelsesområde (HL)
FFCA
FFCC A til C

```
FFCE   Nul
FFD2   svarende til oprindelig AKK
FFD4   Sæt carry
******  Sammenlign HL <> DE
FFD8
FFD9   H - D
FFDB   L - C
******  Sammenligning HL <> BC
FFDE
FFDF   H - B
FFE1   L - C
******  BC := HL - DE
FFE4
FFE6   HL := HL - DE
FFE9   BC := HL
******  Blocktransfer LDIR, antal i A
FFEC   Antal til C
FFED   Sæt Hi-byte til 0
FFF0   Er tæller BC = 0 ?
FFF1   Ja, så færdig
FFF2   Blocktransfer
******  Blocktransfer LDDR
FFF5
FFF6   Er tæller BC = 0 ?
FFF7   Ja, så færdig
FFF8   Blocktransfer;
******  Hop til (HL)
FFFB
******  Hop til (BC)
FFFC
******  Hop til (DE)
FFFE
```

4. TILLÆG

4.1. OPERATIVSYSTEMETS RUTINER

Vi har her udlistet operativsystemets rutiner og tabeller i det omfang, de er os bekendte.

ADVARSEL: Forsøg aldrig at lave tilgang til de i det følgende viste adresser, dersom man ikke er fortrolig med mekanismen til omstilling af hukommelseskonfigurationerne.

Benyt da hellere de i kapitel 2.1 anførte vektorer.

Opstillingen tjener først og fremmest til at give et hurtigt overblik over operativsystemet. Derfor har vi listet CPC 6128-operativsystemets rutiner (se kapitel 2.5). Der gælder en lignende konfiguration for CPC 664, med undtagelse af ganske få adresser.

KERNEL

0000	RST 0 RESET ENTRY
0008	RST 1 LOW JUMP
0010	RST 2 SIDE CALL
0018	RST 3 FAR CALL
0020	RST 4 RAM LAM
0028	RST 5 FIRM JUMP
0030	RST 6 USER RESTART
0038	RST 7 INTERRUPT ENTRY
0040	Hertil kopieres til RAM
0044	Restore High Kernel Jumps
005C	KL CHOKE OFF
0099	KL TIME PLEASE
00A3	KL TIME SET
00B1	Scan Events
0153	Kick Event
0163	KL NEW FRAME FLY
016A	KL ADD FRAME FLY
0170	KL DEL FRAME FLY
0176	KL NEW FAST TICKER
017D	KL ADD FAST TICKER
0183	KL DEL FAST TICKER
0189	Ticker Chain bearbejdning
01B3	KL ADD TICKER
01C5	KL DEL TICKER
01D2	KL INIT EVENT
01E2	KL EVENT
0219	KL DO SYNC
0227	KL SYNC RESET
022E	Sync Event implementering
0255	KL NEXT SYNC
0276	KL DONE SYNC
0284	KL DEL SYNCHRONOUS
028D	KL DISARM EVENT
0294	KL EVENT DISABLE
029A	KL EVENT ENABLE
02A0	KL LOG EXT
02B1	KL FIND COMMAND
0326	KL ROM WALK
0330	KL INIT BACK
0379	Add Event
0388	Delete Event
0397	KL SÆT RAM-KONFIGURATION
03C7	KL POLL SYNCHRONOUS
03E7	RST 7 INTERRUPT ENTRY CONT'D
041E	KL EXT INTERRUPT ENTRY
042A	KL LOW PCHL CONT'D
0430	RST 1 LOW JUMP CONT'D
045F	KL FAR PCHL CONT'D
0467	KL FAR ICALL CONT'D
046D	RST 3 LOW FAR CALL CONT'D
04BD	KL SIDE PCHL CONT'D
04C3	RST 2 LOW SIDE CALL CONT'D
04DB	RST 5 FIRM JUMP CONT'D

04F7	KL L ROM ENABLE CONT'D
04FE	KL L ROM DISABLE CONT'D
0505	KL U ROM ENABLE CONT'D
050C	KL U ROM DISABLE CONT'D
0516	KL ROM RESTORE CONT'D
051F	KL ROM SELECT CONT'D
0524	KL PROBE ROM CONT'D
052D	KL ROM DESELECT CONT'D
0543	KL CURR SELECTION CONT'D
0547	KL LDIR CONT'D
054D	KL LDDR CONT'D
0553	KL ROM OFF & KONFIG. SAVE
056C	RST 4 RAM LAM CONT'D
057D	KL RAM LAM (IX)

MACHINE PACK

0591	Reset Cont'd
05C5	Tabelle 60Hz
05D5	Tabelle 50Hz
05ED	MC BOOT PROGRAM
061C	MC START PROGRAM
0677	Koldstart
0688	Opstartsmelding
06FC	Udskriv meddelelse
0705	Load-Error-Message
0738	Firma
0776	MC SET MODE
0786	MC CLEAR INKS
078C	MC SET INKS
07AA	COLOR IMAGE
07B4	MC WAIT FLYBACK
07C0	MC SCREEN OFFSET
07E0	MC RESET PRINTER
07F7	KONVERTERING FOR NATIONALE TEGN
080C	MC TEGNTILORDNING
081B	MC PRINT CHAR
0835	MC WAIT PRINTER
0844	MC SEND PRINTER
0858	MC BUSY PRINTER
0863	MC SOUND REGISTER
0883	Scan Keyboard

JUMP RESTORE

08BD	JUMP RESTORE
08DE	Main Jump Adress
0A72	BASIC Jump Adr.
0AB4	Move (hl+3) nach ((hl+1)),cnt=(hl)

SCREEN PACK

0ABF	SCR INITIALISE
0AD0	SCR RESET
0AE9	SCR SET MODE
0B0C	SCR GET MODE
0B17	SCR CLEAR
0B37	SCR SET OFFSET
0B3C	SCR SET BASE
0B45	SCR ÆNDRING AF SCREEN-START
0B56	SCR GET LOCATION
0B5D	SCR CHAR LIMITS
0B6A	SCR CHAR POSTION
0BAF	SCR DOT POSITION
0C05	SCR NEXT BYTE
0C11	SCR PREV BYTE
0C1F	SCR NEXT LINE
0C39	SCR PREV LINE
0C55	SCR ACCESS
0C71	SCR WRITE
0C74	SCR PIXELS
0C7A	XOR Mode
0C7F	AND Mode
0C85	OR Mode
0C8A	SCR READ
0C8E	SCR INK ENCODE
0CA7	SCR INK DECODE
0CD8	Reset colors
0CEA	SCR SET FLASHING
0CEE	SCR GET FLASHING
0CF2	SCR SET INK
0CF7	SCR SET BORDER
0CF8	Set Colour
0D10	GET COLOUR ELEMENTS
0D1A	SCR GET INK
0D1F	SCR GET BORDER
0D20	Get Colour
0D35	Hent INK-adresse

0D61 Set Inks on Frame Fly
0D73 Flash Inks
0D87 Parametre for aktuel farvevalg
0D99 Farvematrix
0DB9 SCR FILL BOX
0DBD SCR FLOOD BOX
0DE5 SCR CHAR INVERT
0DF8 Adressering af farvehukommelsen
0E00 SCR HW ROLL
0E44 SCR SW ROLL
0EF9 SCR UNPACK
0F2A SCR REPACK
0F93 SCR HORIZONTAL
0F9B SCR VERTICAL
1052 Default Colours

TEXT SCREEN

1074 TXT INITIALISE
1084 TXT RESET
109F Reset Params (alle vinduer)
10E4 TXT STR SELECT
1103 TXT SWAP STREAMS
111E ldir cnt=15
1126 Adr. for vinduesparametre til DE
1139 Sæt default parametre
115A TXT SET COLUMN
1165 TXT SET ROW
1170 TXT SET CURSOR
117C TXT GET CURSOR
1186 Aktuel vindue øverst,venstre + HL
1193 Aktuel vindue øverst,venstre - HL
11A4 Move Cursor
11CA TXT VALIDATE
11D6 HL indenfor vinduets afgrænsninger
1208 TXT WIN ENABLE
1252 TXT GET WINDOW
125F TXT DRAW/UNDRAW CURSOR
1265 TXT PLACE/REMOVE CURSOR
1276 TXT CUR ON
127E TXT CUR OFF
1286 TXT CUR ENABLE
1288 Cur Enable Cont'd
1297 TXT CUR DISABLE
1299 Cur Disable Cont'd
12A6 TXT SET PEN
12AB TXT SET PAPER
12BA TXT GET PEN
12C0 TXT GET PAPER
12C6 TXT INVERSE
12D4 TXT GET MATRIX
12F2 TXT SET MATRIX
12FE TXT SET M TABLE
132B TXT GET M TABLE
1335 TXT WR CHAR
134B TXT WRITE CHAR
137B TXT SET BACK
1388 TXT GET BACK
13A8 TXT SET GRAPHIC
13AC TXT RD CHAR
13BE TXT UNWRITE CHAR
13FE TXT OUTPUT
140A TXT OUT ACTION
1452 TXT VDU DISABLE
1459 TXT VDU ENABLE
1460 AKTUEL CURSOR-FLAG TIL AKK
1464 Kopier default styretegn
1474 Kopier default styretegn
14D4 TXT GET CONTROLS
14E1 Bell
14EC Transparentmode on/off
14F1 INK-kommando
14FA BORDER-kommando
1501 Definering af vindue
150D SYMBOL-kommando
1519 CRSR Left
151E CRSR Right
1523 CRSR Down
1528 CRSR Up
1539 CRSR Home
153F CRSR til start af linie
1547 LOCATE-kommando
154F TXT CLEAR WINDOW
155E Slet tegn på cursor-position
1565 Slet vindue fra cursor-position
1578 Slet vindue til cursor-position
158F Slet linie fra cursorposition
1599 Slet linie til cursor-position

GRAPHICS SCREEN

15A8	GRA INITIALISE
15D7	GRA RESET
15EC	NN
15FB	GRA MOVE RELATIVE
15FE	GRA MOVE ABSOLUTE
1606	GRA ASK CURSOR
160E	GRA SET ORIGIN
161C	GRA GET ORIGIN
1624	Hent fysisk startposition
1627	Hent fysisk startposition + placering af cursor
162A	GRA KOORD. KONVERTERES
165D	Adresse for løbende koordinat + relativ koordinat
16A5	GRA WIN WIDTH
16EA	GRA WIN HEIGHT
1717	GRA GET W WIDTH
172D	GRA GET W HEIGHT
1736	GRA CLEAR WINDOW
1767	GRA SET PEN
176E	GRA SET PAPER
1775	GRA GET PEN
177A	GRA GET PAPER
1780	GRA PLOT RELATIVE
1783	GRA PLOT ABSOLUTE
1786	GRA PLOT
1794	GRA TEST RELATIVE
1797	GRA TEST ABSOLUTE
179A	GRA TEST
17A6	GRA LINE RELATIVE
17A9	GRA LINE ABSOLUTE
17AC	GRA MASK PARAM SAVE
17B0	GRA MASK PARAM SAVE
17B4	GRA LINE
1940	GRA WR CHAR
19D5	GRA PARAM SAVE
19D9	GRA FILL

KEYBOARD MANAGER

1B5C	KM INITIALISE
1B98	KM RESET
1BBF	KM WAIT CHAR
1BC5	KM READ CHAR
1BFA	KM CHAR RETURN
1C04	KM EXP BUFFER
1C0A	Exp Buffer Cont'd
1C3C	Default Exp String
1C46	KM SET EXPAND
1C6A	Exp Buffer CLEAR
1CA7	Plads til ny streng?
1CB3	KM GET EXPAND
1CC3	Adresse for aux-string til DE
1CDB	KM WAIT KEY
1CE1	KM READ KEY
1D38	KM GET STATE
1D3C	Set State
1D40	KM UPDATE KEY STATE MAP
1DB8	KM TEST BREAK
1DE5	KM GET JOYSTICK
1DF2	KM GET DELAY
1DF6	KM SET DELAY
1DFA	KM ARM BREAK
1E0B	KM DISARM BREAK
1E19	KM BREAK EVENT
1E2F	KM GET REPEAT
1E34	KM SET REPEAT
1E45	KM TEST KEY
1E55	Hent bit svarende til KEY#
1E6D	Bit Mask
1EC4	KM GET TRANSLATE
1EC9	KM GET SHIFT
1ECE	KM GET CONTROL
1ED1	Get Key Table
1ED8	KM SET TRANSLATE
1EDD	KM SET SHIFT
1EE2	KM SET CONTROL
1EE5	Set Key Table
1EEF	Key Translation Table
1F3F	Key SHIFT Table
1F8F	Key CTRL Table

SOUND MANAGER

1FE9	SOUND RESET
2050	SOUND HOLD
206B	SOUND CONTINUE
208B	Sound Event
20D7	Scan Sound Queues
2114	SOUND QUEUE
21AC	SOUND RELEASE
21CE	SOUND CHECK
21EB	SOUND ARM EVENT
23DB	SET VOLUME

2495 SOUND AMPL ENVELOPE
249A SOUND TONE ENVELOPE
249D COPY ENVELOPE
24A6 SOUND A ADRESS
24AB SOUND T ADRESS
24AE GET ENVELOPE ADRESS

CASSETTE MANAGER

24BC CAS INITIALISE
24CE CAS SET SPEED
24E1 CAS NOISY
24E5 CAS IN OPEN
24FE CAS OUT OPEN
2502 Cass. Open
2550 CAS IN CLOSE
2557 CAS IN ABANDON
257F CAS OUT CLOSE
2599 CAS OUT ABANDON
25A0 CAS IN CHAR
25C6 CAS OUT CHAR
25F6 Check Input Buffer Status
25F9 Check Buffer Status
2603 CAS TEST EOF
2607 CAS RETURN
2618 CAS IN DIRECT
2653 CAS OUT DIRECT
2692 CAS CATALOG
26AC READ FILE HEADER
2891 DISPLAY CASS MESSAGE (# in B)
28F0 DISPLAY CASS MESSAGE (1 CHAR)
2935 CASS MESSAGES
29A6 CAS READ
29AF CAS WRITE
29C1 CAS CHECK
29E3 MOTOR ON & OPEN KEYBOARD
2B3D Cass. Input RD DATA & Test ESC
2BA7 Cass. Output WR DATA
2BBB CAS START MOTOR
2BBF CAS STOP MOTOR
2BC1 CAS RESTORE MOTOR

SCREEN EDITOR

2C02 EDIT
2C42 EDIT JUMP
2C72 EDIT JUMPTABLE #1
2CAE EDIT JUMPTABLE #2
2CBD CRSR UP
2CC1 CRSR DWN
2CC5 CRSR RGHT
2CC9 CRSR LEFT
2CD0 ESC
2CEA BREAK-MESSAGE
2CF1 ENTER
2CFE BELL
2D02 CRSR RGHT (buffer)
2D0A CRSR DWN (buffer)
2D14 CTRL & CRSR RGHT
2D1D CTRL & CRSR DWN
2D34 CRSR LEFT (buffer)
2D3C CRSR UP (buffer)
2D45 CTRL & CRSR LEFT
2D4F CTRL & CRSR UP
2D81 CTRL & TAB (Filp Insert)
2D8A INDENT
2DC3 DEL
2DCD CLR
2E17 SHFT & CRSR RGHT
2E1C SHFT & CRSR LEFT
2E21 SHFT & CRSR UP
2E26 SHFT & CRSR DWN
2E65 COPY
2F56 CHAR FROM KEYBOARD

ARITHMETIK

2F73 FLO PI
2F91 COPY FLO VARIABLE FROM (DE) TO (HL)
2F9F FLO INTEGER TO FLOATING POINT
2FC8 FLO 4-BYTE-VALUE TO FLOATING POINT
2FD1 FLO 4-BYTE-VALUE MULTIPLIED BY 256 TO INTEGER
2FD9 FLO FLOATING POINT TO INTEGER
3001 FLO FLOATING POINT TO INTEGER
3014 FLO FIX
3055 FLO INT
305F FLO

30C6	FLO NUMBER MULTIPLIED BY 10↑A
3136	FLO RND INIT
3143	FLO SET RANDOM SEED
3159	FLO RND
3188	FLO GET LAST RND-VALUE
31B1	FLO LOG10
31B6	FLO LOG
322F	FLO EXP
32AC	FLO SQR
32AF	FLO RAISING POWER
3345	FLO DEG/RAD
3349	FLO COS
3353	FLO SIN
33C8	FLO TAN
33D8	FLO ATN
349E	FLO SUBTRACTION
34A2	FLO ADDITION
3577	FLO MULITIPLICATION
3604	FLO DIVISION
36DF	FLO EQUALITY
3727	FLO SGN
3731	FLO CHANGE SIGN

CHARACTERS

3800-3FFF CHARACTERS

4.2. REFERENCER TIL SYSTEM-RAM

I det følgende findes der til hver RAM-adresse, dersom disse kan findes i operativsystemets område, referencer til de steder, de benyttes. Dette kan være en stor hjælp ved manipulering af indholdet i RAM via egne programmer og rutiner. Her drejer det sig ligeledes om en CPC 6128-listning.

B100:	0638							
B101:	063B							
B114:	2DA5	2DBB	2DDE	2DEA				
B115:	2C24	2D81	2D85	2D8D				
B116:	2DF3	2DFA	2E13	2E41	2EC1			
B117:	2DF6							
B118:	24E1	2807	28D2					
B119:	280C	290F						
B11A:	24E5	2550	2557	25F6	2692	26E0	271B	292F
B11B:	263C	269C	26EF					
B11D:	25BC	25C1	260F	2613	26F2			
B11F:	2743	274E	2760					
B12F:	26FC							
B130:	26AC							
B131:	24FA							
B132:	25AA	25B5	25B9	2608	260C	2629	263F	270C
B134:	24F2	261F	2626	26DD				
B136:	2706							
B137:	24F6							
B15F:	24FE	257F	2599	25CA	2656	27D9		
B160:	266E	2685	279E					
B162:	25EA	25EF	27A1					
B164:	2790	27A8						

B174: 27CD
B175: 258B 27BF
B176: 2663
B177: 25D4 25E3 25E7 2671 267E 27B6 27CA
B179: 27A4
B17B: 2796 27D2
B17C: 2666
B17E: 266A
B1A4: 26BB 274B 2763 2804
B1B5: 2700
B1B7: 26D9 2709
B1B9: 2022 2072 2094 20BE 2122 214D 21B9
B1BB: 273D
B1BC: 21D1
B1BE: 20E9 2637
B1D5: 21EF
B1E4: 2564 27E5
B1E5: 29E3 2ACD 2AE3
B1E6: 2AC6 2B23
B1E7: 24DC
B1E8: 2B78 2B8B
B1E9: 24D9
B1EA: 2B7C
B1EB: 2B00 2B12 2B16
B1ED: 1FE9 206B
B1EE: 2050 208D 20B7 20D7 2258 2286
B1F0: 201D 20D1 210C 2147 21B4
B1F8: 2000 2296
B237: 229E 22C0
B276: 22A6 22B8
B2A6: 2303 2495 24A6
B2B5: 1FFD 23EF
B396: 249A 24AB
B590: 1B9E
B5D6: 1B6E
B628: 1BCF 1BF0
B629: 1C38
B62A: 1BC6 1BFA
B62B: 1C17 1CC9
B62D: 1C13
B62F: 1C35 1C96 1CA1 1CA7
B630: 1CAC
B631: 1B68 1D12 1D2B 1D38 1D3C
B632: 1CFB
B633: 1D9E 1DF2 1DF6
B634: 1DD8
B635: 1B8A 1D57 1D86 1E4D
B637: 1D4F 1E46
B63B: 1DE5

B63D: 1DB8
B63E: 1DEB
B63F: 1B8D 1D43
B649: 1D40 1D54
B64B: 1D49
B653: 1D7A 1D92 1DA1
B654: 1D7F 1DAC
B655: 1B63
B656: 1E0D 1E19
B657: 1DFD
B686: 1E76 1E86 1EAE
B688: 1E97 1E9D
B68A: 1D96 1E93 1EAA
B68B: 1EC4 1ED8
B68D: 1EC9 1EDD
B68F: 1ECE 1EE2
B691: 1D8B 1E2F 1E37
B692: 1B71
B693: 160E 161C 1640
B695: 1612 1620 1655
B697: 15FE 1606 165E
B699: 1602 160A 1664
B69B: 166A 16C9 1717 1753 1910
B69D: 1673 16CD 171B 1906
B69F: 1680 16FB 172D 174A 18B9 1AE8
B6A1: 1689 16FF 1731 1746 18C3 1B18
B6A3: 0FA5 0FAE 0FB1 0FFF 101C 176A 1775 178D 19C4 1B34
B6A4: 0FF3 1027 175D 1771 177A 19CE
B6A5: 17BD 188F 18C8 18DA 18E6 18EF 18FA 18FF 19D9 1A19
1A44 1AAC 1AC1
B6A7: 17CC 1893 18A2 18AD 18B2 1915 1928 1934 19DF 1A25
1A2C 1A9F 1AA9
B6A9: 1802 1861 19FE 1A4B 1AC6
B6AA: 19E6 1B3A
B6AB: 17EC 1846 1A0B 1A21 1ABD 1AD7 1ADB 1ADF
B6AC: 1A50 1A79
B6AD: 17C4 17E8 1812 181B 18D2 18DD
B6AE: 17D3 17E2 191F 1A5D 1A66 1A94
B6AF: 17DF 1828 1898
B6B0: 17F9 1868 1876 1880 1A76 1A97
B6B2: 17B0 17F2 1820
B6B3: 0FA9 0FB4 0FBA 1012 104C 17AC
B6B4: 0FF7 1021 19C9 19D5
B6B5: 10AF 10B3 10E6 1103 110C
B6B6: 10A1
B726: 10A4 1135 115F 116A 1176 117C 11A7 11AD 1340 1555
156F 1582
B728: 123A 1259
B729: 1166 1186 1193 11EF 1229 1252 1539 1552 1568 157B

B72A: 115B 118C 119B 11DD 11E2 1542 159E
B72B: 11F7 122C 1255 1558 156B
B72C: 11D6 11EA 157E 1593
B72D: 1182 11B2
B72E: 113C 125F 128E 129F 1336 143B 1460
B72F: 10CA 10DA 126B 12A6 12BA 12C9 12CF 1392 13A0 13DB
B730: 11BD 12AB 12C0 13BE 1589
B731: 1377 1384 1388
B733: 13A8 140B
B734: 1321 132B
B735: 1078
B736: 1326 1331
B738: 134F 13C1 13E7
B758: 1413 144E 1465
B759: 142C 1446
B763: 146B
B7C3: 0B0C 0B31
B7C4: 0B3C 0B51 0B56 0B8A 0E2A 0E3D
B7C5: 0B20
B7C6: 0AC7 0B37 0B47 0B59 0B93 0BED 0E32
B7C7: 0C6A 0C71
B7C8: 0C6D
B7D2: 0CEA 0CEE 0D95
B7D3: 0D8E
B7D4: 0CDB 0D92
B7E5: 0D38 0D87
B7F6: 0CE4 0D7C 0D8A
B7F7: 0D0C 0D83
B7F8: 0D61 0D73
B7F9: 0D42 0D55
B802: 0FA1 0FBD
B804: 07E3 0812
B82D: 0066 00F2 011D 0127
B82E: 00EC
B82F: 00F5 00FE 0102
B831: 00E2 00F8 0114 0132 0142 03FE
B832: 010A 014E
B8B4: 009E 00AC 00B1 010E
B8B6: 009A 00A8
B8B8: 00A5
B8B9: 00BF 016A 0170
B8BB: 00C7 017D 0183
B8BD: 00DC 0189 01BF 01C5
B8BF: 00D2 03D0
B8C0: 0256 026E 0287 03D6
B8C1: 022A 03C7
B8C2: 0263 026B 0276 0294 029A 03E0
B8C3: 0230 02B1 0307
B8D3: 02A1 02A5 02BE

B8D5:	0399					
B8D6:	0080	0351	0484	04B5	0539	0543
B8D7:	0060	0086				
B8D9:	005D	0083	0330	04D5		
B8DA:	034E					

00	Zeilenende	93	DIM
01	':', Ende des Statements	94	DRAW
02	Integervariable '%'	95	DRAWR
03	Stringvariable '$'	96	EDIT
04	Realvariable '!'	97	ELSE
0D	Variable ohne Kennzeichen	98	END
0E	Konstante 0	99	ENT
0F	Konstante 1	9A	ENV
10	Konstante 2	9B	ERASE
11	Konstante 3	9C	ERROR
12	Konstante 4	9D	EVERY
13	Konstante 5	9E	FOR
14	Konstante 6	9F	GOSUB
15	Konstante 7	A0	GOTO
16	Konstante 8	A1	IF
17	Konstante 9	A2	INK
19	Ein-Byte-Wert	A3	INPUT
1A	Zwei-Byte-Wert, dezimal	A4	KEY
1B	Zwei-Byte-Wert, binär	A5	LET
1C	Zwei-Byte-Wert, hex	A6	LINE
1D	Zeilenadresse	A7	LIST
1E	Zeilennummer	A8	LOAD
1F	Fließkommawert	A9	LOCATE
80	AFTER	AA	MEMORY
81	AUTO	AB	MERGE
82	BORDER	AC	MID$
83	CALL	AD	MODE
84	CAT	AE	MOVE
85	CHAIN	AF	MOVER
86	CLEAR	B0	NEXT
87	CLG	B1	NEW
88	CLOSEIN	B2	ON
89	CLOSEOUT	B3	ON BREAK
8A	CLS	B4	ON ERROR GOTO 0
8B	CONT	B5	ON SQ
8C	DATA	B6	OPENIN
8D	DEF	B7	OPENOUT
8E	DEFINT	B8	ORIGIN
8F	DEFREAL	B9	OUT
90	DEFSTR	BA	PAPER
91	DEG	BB	PEN
92	DELETE	BC	PLOT

BD	PLOTR
BE	POKE
BF	PRINT
C0	'
C1	RAD
C2	RANDOMIZE
C3	READ
C4	RELEASE
C5	REM
C6	RENUM
C7	RESTORE
C8	RESUME
C9	RETURN
CA	RUN
CB	SAVE
CC	SOUND
CD	SPEED
CE	STOP
CF	SYMBOL
D0	TAG
D1	TAGOFF
D2	TRON
D3	TROFF
D4	WAIT
D5	WEND
D6	WHILE
D7	WIDTH
D8	WINDOW
D9	ZONE
DA	WRITE
DB	DI
DC	EI
DD	FILL
DE	GRAPHICS
DF	MASK
E0	FRAME
E1	CURSOR
E3	ERL
E4	FN
E5	SPC
E6	STEP
E7	SWAP

EA	TAB
EB	THEN
EC	TO
ED	USING
EE	>
EF	=
F0	>=
F1	<
F2	<>
F3	<=
F4	+
F5	-
F6	*
F7	/
F8	^
F9	'Backslash'
FA	AND
FB	MOD
FC	OR
FD	XOR
FE	NOT
FF	Funktion

Token &FF står for een funktion, hvorefter de følgende tokens kan bruges.

00	ABS	71	BIN$
01	ASC	72	DEC$
02	ATN	73	HEX$
03	CHR$	74	INSTR
04	CINT	75	LEFT$
05	COS	76	MAX
06	CREAL	77	MIN
07	EXP	78	POS
08	FIX	79	RIGHT$
09	FRE	7A	ROUND
0A	INKEY	7B	STRING$
0B	INP	7C	TEST
0C	INT	7D	TESTR
0D	JOY	7E	COPYCHR$
0E	LEN	7F	VPOS
0F	LOG		
10	LOG10		
11	LOWER$		
12	PEEK		
13	REMAIN		
14	SGN		
15	SIN		
16	SPACE$		
17	SQ		
18	SQR		
19	STR$		
1A	TAN		
1B	UNT		
1C	UPPER$		
1D	VAL		
40	EOF		
41	ERR		
42	HIMEM		
43	INKEY$		
44	PI		
45	RND		
46	TIME		
47	XPOS		
48	YPOS		
49	DERR		

MONITOR

Vi kan næsten forestille os, hvorledes læserne brænder efter at finde ud af, hvad der gemmer sig bag hver enkelt ROM-listning, der jo kun giver et symbolsk billede på, hvad der kunne tænkes at skjule sig i operativsystemet. Men dersom man ikke selv ejer en MONITOR, så er der faktisk ikke andet at gøre, end at gå i krig med at taste det i det følgende offentliggjorte program.

På nær to små maskinkoderutiner til henholdsvis læsning af en byte fra hukommelsen og til hentning af en byte fra en fil, er programmet skrevet komplet i BASIC. Der er dog en ulempe ved det, idet bestemte kommandoer af typen (IX+xx) ikke kan understøttes. Dukker der en sådan op, vil programmet straks melde at der er tale om en specialkommando. Alt efter behov, må man så selv sammensætte kommandoen ud af bitmønsteret.

Komandoernes format svarer ikke helt til Z80-standarden, hvilket bl.a. betyder at immediate-values kendetegnes med et foranstillet dobbeltkryds (numerisk tegn). Dobbeltbyte-værdier, der angives uden, er alle adresser.

Der er mulighed for at disassemblere i RAM, ROM eller filer, hvor sidstnævnte byder på særlige muligheder, der ikke i samme enkelthed, kan fås på andre måder.

Bemærk at programmet SKAL kaldes "MIMO.BAS" for at OPENIN kan finde sin fil.

Men nu til kommando-oversigten, hvor der gælder den grundlæggende syntaks, at det er HEX-værdier, der angives umiddelbart efter kommandoen, f.eks. vil m48 ‹ENTER› sætter den aktuelle adresse til $0048.

d Disassemblering fra aktuel adresse. Funktionen standses ved tryk på en vilkårlig tast.

f Efter bogstavet skal det fuldstændige filnavn placeres. Den følgende angivelse sætter den relative adresse, hvorfra filen skal vises på skærmen. Der læses fra starten af filen. Filmode, afbrydes ved funktionen m.

i Skriv bytes i hukommelsen. Kommandoen kræver ikke yderligere parametre. Funktionen standses ved indtasting af en tom streng.

o Indstilling af Output-fil, hvor 0 er standard og giver alle output i mode 1. 1 sætter skærmen i mode 2 og inddeler samtidigt skærmen, så den øverste trediedel bruges til enkel display og resten til disassembleren. 8 sender output til printeren.

m Angiver den løbende/aktuelle adresse, hvorpå alle efterflg. kommandoer rettes.

b Indstiller hukommelseskonfigurationen. Den krævede byte har en opbygning som det er beskrevet andre steder i nærværende bog. FE vælger eksempelvis begge indbyggede ROMs og den mellemliggende RAM. FE vælger kun RAM.

$ Konverterer decimale parametre til HEX.

% Konverterer HEX-tal (max. 4 positioner) til decimaltal.

x Afslutter programmer og genskaber tidligere hukommelse.

? Udfører en varmstart og viser kommando-oversigten.

‹ENTER› lister hukommelsesindholdet i HEX og ASCII.

Held og lykke med indtastningen!

```
10 top=HIMEM
20 ON ERROR GOTO 40
30 OPENIN "mimo.bas"
40 RESUME NEXT
50 MEMORY HIMEM-1
60 CLOSEIN
70 him=HIMEM-256
80 ZONE 8:lf=0
90 mpb=him-20:MEMORY mpb-1
100 GOSUB 1350:ms=&FE
110 CLS:INK 3,6:b0=1:b1=24:b2=22:b3=0
120 DIM l%(4,255),mn$(4,255),pu%(15)
130 GOSUB 1010:a=0
140 bs$=STRING$(32,8):bl$=SPACE$(30)
150 IF plf=1 THEN lf=0:plf=0
160 MODE 1:PRINT:PRINT: PRINT"c = Call Maschinenprogramm"
170 PRINT"d = Diassemblieren"
180 PRINT"f = File"
190 PRINT"i = Insert Bytes"
200 PRINT"o = Output-lf#"
210 PRINT"m = Memoryadress"
220 PRINT"b = Bank-select
230 PRINT"$ = Dezimal -> Hex"
240 PRINT"% = Hex -> Dezimal"
250 PRINT"x = Ende"
260 PRINT"? = Warmstart"
270 PRINT:GOTO 290
280 IF lf=0 OR lf>7 THEN MODE 1
290 BORDER b0:INK 0,b0:INK 1,b1:PRINT:PRINT"bank= ";HEX$(ms,2):PRINT "mem = ";
    HEX$(a,4):i=a:PRINT"lf# =";lf:PRINT
300 INPUT">",h$:hl$=LEFT$(h$,1)
310 IF h$="?" THEN GOTO 150
320 IF h$="x" THEN MEMORY top:MODE 1:END
330 IF hl$<>"o" THEN 370
340 lf=VAL(RIGHT$(h$,1)):IF lf=0 OR lf>7 THEN plf=0:GOTO 280
350 IF plf=0 THEN MODE 2:WINDOW #0,1,80,25,25:WINDOW #1,1,80,1,8:
    WINDOW #2,1,80,9,25:plf=1
360 GOTO 290
```

```
370 IF hl$="$" THEN PRINT HEX$(VAL(RIGHT$(h$,LEN(h$)-1))):GOTO 290
380 IF hl$<>"%" THEN 410
390 xx=(VAL("&"+RIGHT$(h$,LEN(h$)-1))):IF xx<0 THEN xx=xx+65536
400 PRINT xx:GOTO 290
410 IF hl$<>"m" THEN 460
420 IF file=1 THEN file=0:CLOSEIN
430 IF LEN(h$)=1 THEN 280
440 a=VAL("&"+RIGHT$(h$,LEN(h$)-1)):IF a<0 THEN a=a+65536
450 padp=a-1:GOTO 280
460 IF hl$<>"b" THEN 490
470 re=VAL("&"+RIGHT$(h$,LEN(h$)-1)):IF re>255 OR re<0 THEN
    PRINT"2-Byte Hexwert verlangt":GOTO 280
480 ms=re:GOTO 280
490 IF hl$<>"f" THEN 570
500 IF file=1 THEN CLOSEIN
510 ON ERROR GOTO 530
520 OPENIN MID$(h$,2)
530 RESUME NEXT
540 INPUT"basis (hex) ";h$
550 h$="m"+h$
560 file=1:GOTO 440
570 REM
580 IF hl$="d" THEN i=a:GOTO 810
590 IF hl$="c" THEN CALL a:GOTO 280
600 IF hl$="i" THEN 780
610 IF LEN(h$)<2 THEN h$="00"
620 bis=VAL("&"+RIGHT$(h$,LEN(h$)-1)):IF bis<1 THEN bis=bis+65536
630 IF plf=0 THEN MODE 2 ELSE lf=1
640 BORDER b2:INK 0,b2:INK 1,b3
650 ON file GOTO 670
660 a=INT(a/16)*16
670 FOR i=a TO bis  STEP 16
680 PRINT#lf,HEX$(i,4);":";:FOR j=0 TO 15
690 pad=i+j:GOSUB 1520:PRINT#lf," ";HEX$(mv,2);
700 NEXT j:PRINT#lf,TAB(60);
710 FOR j=0 TO 15:pad=i+j:GOSUB 1520:he=(mv AND 127)
720 IF he<32 OR he=127 THEN he=46
730 PRINT#lf,CHR$(he);:NEXT j:PRINT#lf
740 IF INKEY$<>"" THEN a=i:i=65535:ELSE a=i+16
```

```
750 NEXT
760 IF lf<>8 THEN INPUT ">ENTER< druecken, wenn fertig";re$
770 GOTO 280
780 i=a
790 PRINT HEX$(i,4);": ";:INPUT"",d$:IF d$="" THEN 280
800 POKE i,VAL("&"+d$):i=i+1:GOTO 790
810 IF plf=1 THEN lf=2:PRINT#lf,CHR$(11);
820 IF LEN(h$)=1 THEN h$="00"
830 bis=VAL("&"+RIGHT$(h$,LEN(h$)-1)):IF bis<1 THEN bis=bis+65536
840 pa=a
850 PAPER 0:IF INKEY$ <>"" THEN a=pa:PRINT#lf:GOTO 280
860 IF pa>bis THEN a=pa:PRINT#lf:GOTO 760
870 pad=pa:GOSUB 1520:op=mv:ad=pa:pa=pa+1
880 IF lf=8 THEN PRINT#lf,LEFT$(bl$,10);
890 PRINT#lf,HEX$(ad,4);"  ";:xx=0
900 PRINT#lf,HEX$(op,2);
910 se=0:GOSUB 1700:IF LEFT$(mn$,1)="?" THEN 1070
920 se=xx:GOSUB 1700:IF mn$="" THEN PAPER 3:PRINT#lf,"        ????":
    PAPER 0:GOTO 850
930 ON l%(xx,op) GOTO 980,970,960,950
940 ON l%(xx,op)-1 GOTO 980,970,960,950
950 pad=pa:GOSUB 1520:PRINT#lf,HEX$(mv,2);:pa=pa+1
960 pad=pa:GOSUB 1520:PRINT#lf,HEX$(mv,2);:pa=pa+1
970 pad=pa:GOSUB 1520:PRINT#lf,HEX$(mv,2);:pa=pa+1
980 PRINT#lf,LEFT$(bl$,(4-l%(xx,op))*2+2);
990 GOSUB 1090
1000 GOTO 850
1010 PRINT:PAPER 3:PRINT"bitte warten";:PAPER 0:PRINT:FOR i=0 TO 4:
     FOR j=0 TO 255
1020 READ a:l%(i,j)=a
1030 NEXT j,i
1040 FOR i=0 TO 4:FOR j=0 TO 255
1050 READ mn$:mn$(i,j)=mn$
1060 NEXT:NEXT:RETURN
1070 xx=l%(0,op):pad=pa:GOSUB 1520:op=mv:se=xx:GOSUB 1700:IF mn$="" THEN 920
1080 PRINT#lf,HEX$(op,2);:pa=pa+1:GOTO 940
1090 se=xx:GOSUB 1700:ln=LEN(mn$)
1100 IF mn$=pmn$ THEN PAPER 3
1110 pmn$=mn$:ppn=1
```

```
1120 IF MID$(mn$,ln-3,4)="+/-^" THEN mn$=LEFT$(mn$,ln-4):GOTO 1230
1130 pn=INSTR(mn$,"*"):IF pn<>0 THEN PRINT#lf,LEFT$(mn$,pn-1);:GOTO 1170
1140 pn=INSTR(ppn,mn$,"^"):IF pn<>0 THEN PRINT#lf,MID$(mn$,ppn,pn-ppn);:
     GOTO 1220
1150 PRINT#lf,mn$;
1160 PRINT#lf:RETURN
1170 pad=pa-2:GOSUB 1520:ar=mv:pn=pn+1
1180 IF pn>ln THEN xz=ar:PRINT#lf,HEX$(xz,2);:GOTO 1160
1190 ppn=pn:IF MID$(mn$,pn,1)<>"^" THEN xz=ar:PRINT#lf,HEX$(xz,2);:GOTO 1140
1200 pn=pn+1:pad=pa-1:GOSUB 1520:yy=256*mv+ar:PRINT#lf,HEX$(yy,4);
1210 PRINT#lf,MID$(mn$,pn):RETURN
1220 pn=pn+1:pad=pa-1:GOSUB 1520:ar=mv:xz=ar:PRINT#lf,HEX$(xz,2);:GOTO 1210
1230 PRINT#lf,mn$;
1240 pn=pn+1:pad=pa-1:GOSUB 1520:ar=mv:yy=ad+2+ar+(ar>127)*256:
     PRINT#lf,HEX$(yy,4);
1250 PRINT#lf:RETURN
1260 sp=1
1270 WHILE MID$(mn$,sp,1)<>" ": sp=sp+1:WEND
1280 WHILE MID$(mn$,sp,1)=" ": sp=sp+1:WEND
1290 ad=cn+VAL(RIGHT$(mn$,LEN(mn$)-sp+1))
1300 ha=INT(ad/256):la=ad-ha*256
1310 PRINT#lf," ($";HEX$(ha,2);HEX$(la,2);")":RETURN
1320 IF MID$(mn$,sp,1)="-" THEN 1340
1330 ad=cn+ar:GOTO 1300
1340 ad=cn+ar-256:GOTO 1300
1350 POKE mpb,&DF
1360 po=mpb+4:ph=INT(po/256):pl=po-ph*256
1370 POKE mpb+1,pl:POKE mpb+2,ph
1380 POKE mpb+3,&C9
1390 po=mpb+7:ph=INT(po/256):pl=po-ph*256
1400 POKE mpb+4,pl:POKE mpb+5,ph
1410 POKE mpb+7,&3A
1420 by=mpb+14:ph=INT(by/256):pl=by-ph*256
1430 POKE mpb+10,&32
1440 POKE mpb+11,pl:POKE mpb+12,ph
1450 POKE mpb+13,&C9
1460 DATA &c1,&d1,&f1,&e1,&f5,&d5,&c5,&cd,&80,&bc,&f5,&d1,&72,&23,&73,&c9
1470 FOR i=1 TO 16
1480 READ a
```

```
1490 mp$=mp$+CHR$(a)
1500 NEXT i
1510 RETURN
1520 IF pad>65535 THEN RETURN
1530 ON file GOTO 1600
1540 IF ms=255 THEN mv=PEEK(pad):RETURN
1550 ph=INT(pad/256):pl=pad-ph*256
1560 POKE mpb+8,pl:POKE mpb+9,ph
1570 POKE mpb+6,ms
1580 CALL mpb
1590 mv=PEEK(by):RETURN
1600 IF padp<pad THEN GOSUB 1630
1610 mv=pu%(pad MOD(16))
1620 RETURN
1630 ret%=0:mpp=@mp$
1640 getf=PEEK(mpp+1)+256*PEEK(mpp+2)
1650 CALL getf,@ret%
1660 mv=ret% AND 255: IF (ret% AND &100)=0 THEN mv=0
1670 padp=padp+1:pu%(padp MOD(16))=mv
1680 IF padp<pad GOTO 1650
1690 RETURN
1700 mn$=mn$(se,op):RETURN
1710 DATA 1 , 3 , 1 , 1 , 1 , 1 , 2 , 1
1720 DATA 1 , 1 , 1 , 1 , 1 , 1 , 2 , 1
1730 DATA 2 , 3 , 1 , 1 , 1 , 1 , 2 , 1
1740 DATA 2 , 1 , 1 , 1 , 1 , 1 , 2 , 1
1750 DATA 2 , 3 , 3 , 1 , 1 , 1 , 2 , 1
1760 DATA 2 , 1 , 3 , 1 , 1 , 1 , 2 , 1
1770 DATA 2 , 3 , 3 , 1 , 1 , 1 , 2 , 1
1780 DATA 2 , 1 , 3 , 1 , 1 , 1 , 2 , 1
1790 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1800 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1810 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1820 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1830 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1840 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1850 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1860 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1870 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
```

```
1880 DATA 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1890 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1900 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1910 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1920 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1930 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1940 DATA 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
1950 DATA 1 , 1 , 3 , 3 , 3 , 1 , 2 , 1
1960 DATA 1 , 1 , 3 , 4 , 3 , 3 , 2 , 1
1970 DATA 1 , 1 , 3 , 2 , 3 , 1 , 2 , 1
1980 DATA 1 , 1 , 3 , 2 , 3 , 2 , 2 , 1
1990 DATA 1 , 1 , 3 , 1 , 3 , 1 , 2 , 1
2000 DATA 1 , 1 , 3 , 1 , 3 , 1 , 2 , 1
2010 DATA 1 , 1 , 3 , 1 , 3 , 1 , 2 , 1
2020 DATA 1 , 1 , 3 , 1 , 3 , 3 , 2 , 1
2030 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2040 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2050 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2060 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2070 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2080 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2090 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2100 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2110 DATA 2 , 2 , 2 , 4 , 2 , 2 , 2 , 2
2120 DATA 2 , 2 , 2 , 4 , 0 , 2 , 0 , 2
2130 DATA 2 , 2 , 2 , 4 , 0 , 0 , 2 , 2
2140 DATA 2 , 2 , 2 , 4 , 0 , 0 , 2 , 2
2150 DATA 2 , 2 , 2 , 4 , 0 , 0 , 0 , 2
2160 DATA 2 , 2 , 2 , 4 , 0 , 0 , 0 , 2
2170 DATA 2 , 0 , 2 , 4 , 0 , 0 , 0 , 0
2180 DATA 2 , 2 , 2 , 4 , 0 , 0 , 0 , 0
2190 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2200 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2210 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2220 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2230 DATA 2 , 2 , 2 , 2 , 0 , 0 , 0 , 0
2240 DATA 2 , 2 , 2 , 2 , 0 , 0 , 0 , 0
2250 DATA 2 , 2 , 2 , 2 , 0 , 0 , 0 , 0
2260 DATA 2 , 2 , 2 , 2 , 0 , 0 , 0 , 0
```

```
2270 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2280 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2290 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2300 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2310 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2320 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2330 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2340 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2350 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2360 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2370 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2380 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2390 DATA 0 , 4 , 4 , 2 , 0 , 0 , 0 , 0
2400 DATA 0 , 2 , 4 , 2 , 0 , 0 , 0 , 0
2410 DATA 0 , 0 , 0 , 0 , 3 , 3 , 4 , 0
2420 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2430 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2440 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2450 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2460 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2470 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2480 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2490 DATA 3 , 3 , 3 , 3 , 3 , 3 , 0 , 3
2500 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2510 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2520 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2530 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2540 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2550 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2560 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2570 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2580 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2590 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2600 DATA 0 , 0 , 0 , 4 , 0 , 0 , 0 , 0
2610 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2620 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2630 DATA 0 , 2 , 0 , 2 , 0 , 2 , 0 , 0
2640 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2650 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
```

```
2660 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2670 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2680 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2690 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2700 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2710 DATA 0 , 4 , 4 , 2 , 0 , 0 , 0 , 0
2720 DATA 0 , 2 , 4 , 2 , 0 , 0 , 0 , 0
2730 DATA 0 , 0 , 0 , 0 , 3 , 3 , 4 , 0
2740 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2750 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2760 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2770 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2780 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2790 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2800 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2810 DATA 3 , 3 , 3 , 3 , 3 , 3 , 0 , 3
2820 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2830 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2840 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2850 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2860 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2870 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2880 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2890 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2900 DATA 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0
2910 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2920 DATA 0 , 0 , 0 , 4 , 0 , 0 , 0 , 0
2930 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2940 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2950 DATA 0 , 2 , 0 , 2 , 0 , 2 , 0 , 0
2960 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2970 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
2980 DATA 0 , 2 , 0 , 0 , 0 , 0 , 0 , 0
2990 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3000 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3010 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3020 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3030 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3040 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
```

```
3050 DATA 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
3060 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3070 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3080 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3090 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3100 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3110 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3120 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3130 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3140 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3150 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3160 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3170 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3180 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3190 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3200 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3210 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3220 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3230 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3240 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3250 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3260 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3270 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3280 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3290 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3300 DATA 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
3310 DATA "nop     ","ld      bc,#*^","ld      (bc),a","inc     bc",
     "inc     b","dec     b","ld      b,#^","rlca    "
3320 DATA "ex      af,af'","add     hl,bc","ld      a,(bc)","dec     bc",
     "inc     c","dec     c","ld      c,#^","rrca    "
3330 DATA "djnz    +/-^","ld      de,#*^","ld      (de),a","inc     de",
     "inc     d","dec     d","ld      d,#^","rla     "
3340 DATA "jr      +/-^","add     hl,de","ld      a,(de)","dec     de",
     "inc     e","dec     e","ld      e,#^","rra     "
3350 DATA "jr      nz,+/-^","ld      hl,#*^","ld      *^,hl","inc     hl",
     "inc     h","dec     h","ld      h,#^","daa     "
3360 DATA "jr      z,+/-^","add     hl,hl","ld      hl,*^","dec     hl",
     "inc     l","dec     l","ld      l,#^","cpl     a"
3370 DATA "jr      nc,+/-^","ld      sp,#*^","ld      *^,a","inc     sp",
```

```
    "inc     (hl)","dec     (hl)","ld      (hl),#^","scf     "
3380 DATA "jr      c,+/-^","add     hl,sp","ld      a,*^","dec     sp",
    "inc     a","dec     a","ld      a,#^","ccf     "
3390 DATA "ld      b,b","ld      b,c","ld      b,d","ld      b,e",
    "ld      b,h","ld      b,l","ld      b,(hl)","ld      b,a"
3400 DATA "ld      c,b","ld      c,c","ld      c,d","ld      c,e",
    "ld      c,h","ld      c,l","ld      c,(hl)","ld      c,a"
3410 DATA "ld      d,b","ld      d,c","ld      d,d","ld      d,e",
    "ld      d,h","ld      d,l","ld      d,(hl)","ld      d,a"
3420 DATA "ld      e,b","ld      e,c","ld      e,d","ld      e,e",
    "ld      e,h","ld      e,l","ld      e,(hl)","ld      e,a"
3430 DATA "ld      h,b","ld      h,c","ld      h,d","ld      h,e",
    "ld      h,h","ld      h,l","ld      h,(hl)","ld      h,a"
3440 DATA "ld      l,b","ld      l,c","ld      l,d","ld      l,e",
    "ld      l,h","ld      l,l","ld      l,(hl)","ld      l,a"
3450 DATA "ld      (hl),b","ld      (hl),c","ld      (hl),d","ld      (hl),e",
    "ld      (hl),h","ld      (hl),l","halt    ","ld      (hl),a"
3460 DATA "ld      a,b","ld      a,c","ld      a,d","ld      a,e",
    "ld      a,h","ld      a,l","ld      a,(hl)","ld      a,a"
3470 DATA "add     a,b","add     a,c","add     a,d","add     a,e",
    "add     a,h","add     a,l","add     a,(hl)","add     a,a"
3480 DATA "","adc     a,c","adc     a,d","adc     a,e","adc     a,h",
    "adc     a,l","adc     a,(hl)","adc     a,a"
3490 DATA "sub     a,b","sub     a,c","sub     a,d","sub     a,e",
    "sub     a,h","sub     a,l","sub     a,(hl)","sub     a,a"
3500 DATA "sbc     a,b","sbc     a,c","sbc     a,d","sbc     a,e",
    "sbc     a,h","sbc     a,l","sbc     a,(hl)","sbc     a,a"
3510 DATA "and     a,b","and     a,c","and     a,d","and     a,e",
    "and     a,h","and     a,l","and     a,(hl)","and     a,a"
3520 DATA "xor     a,b","xor     a,c","xor     a,d","xor     a,e",
    "xor     a,h","xor     a,l","xor     a,(hl)","xor     a,a"
3530 DATA "or      a,b","or      a,c","or      a,d","or      a,e",
    "or      a,h","or      a,l","or      a,(hl)","or      a,a"
3540 DATA "cp      a,b","cp      a,c","cp      a,d","cp      a,e",
    "cp      a,h","cp      a,l","cp      a,(hl)","cp      a,a"
3550 DATA "ret     nz","pop     bc","jp      nz,*^","jp      *^",
    "call    nz,*^","push    bc","add     a,#^","rst     0"
3560 DATA "ret     z","ret     ","jp      z,*^","?","call    z,*^",
    "call    *^","adc     a,#^","rst     1"
```

```
3570 DATA "ret     nc","pop     de","jp      nc,*^","out     (^),a",
     "call    nc,*^","push    de","sub     a,#^","rst     2"
3580 DATA "ret     c","exx     ","jp      c,*^","in      a,(^)","call    c,*^",
     "?","sbc     a,#^","rst     3"
3590 DATA "ret     po","pop     hl","jp      po,*^","ex      (sp),hl",
     "call    po,*^","push    hl","and     a,#^","rst     4"
3600 DATA "ret     pe","jp      (hl)","jp      pe,*^","ex      de,hl",
     "call    pe,*^","?","xor     a,#^","rst     5"
3610 DATA "ret     p","pop     af","jp      p,*^","di      ","call    p,*^",
     "push    af","or      a,#^","rst     6"
3620 DATA "ret     m","ld      sp,hl","jp      m,*^","ei      ","call    m,*^",
     "?","cp      a,#^","rst     7"
3630 DATA "","","","","","","",""
3640 DATA "","","","","","","",""
3650 DATA "","","","","","","",""
3660 DATA "","","","","","","",""
3670 DATA "","","","","","","",""
3680 DATA "","","","","","","",""
3690 DATA "","","","","","","",""
3700 DATA "","","","","","","",""
3710 DATA "in      b,(c)","out     (c),b","sbc     hl,bc","ld      *^,bc",
     "neg     a","retn    ","im      0","ld      i,a"
3720 DATA "in      c,(c)","out     (c),c","adc     hl,bc","ld      bc,*^","",
     "reti    ","","ld      r,a"
3730 DATA "in      d,(c)","out     (c),d","sbc     hl,de","ld      *^,de","",
     "","im      1","ld      a,i"
3740 DATA "in      e,(c)","out     (c),e","adc     hl,de","ld      de,*^","",
     "","im      2","ld      a,r"
3750 DATA "in      h,(c)","out     (c),h","sbc     hl,hl","ld      *^,hl","",
     "","","rld     a,(hl)"
3760 DATA "in      l,(c)","out     (c),l","adc     hl,hl","ld      hl,*^","",
     "","","rld     a,(hl)"
3770 DATA "in      f,(c)","","sbc     hl,sp","ld      *^,sp","","","",""
3780 DATA "in      a,(c)","out     (c),a","adc     hl,sp","ld      sp,*^","",
     "","",""
3790 DATA "","","","","","","",""
3800 DATA "","","","","","","",""
3810 DATA "","","","","","","",""
3820 DATA "","","","","","","",""
```

```
3830 DATA "ldi     (de),(hl)","cpi     a,(hl)","ini     (hl),(c)",
     "outi    (c),(hl)","","","",""
3840 DATA "ldd     (de),(hl)","cpd     a,(hl)","ind     (hl),(c)",
     "outd    (c),(hl)","","","",""
3850 DATA "ldir    (de),(hl)","cpir    a,(hl)","inir    (hl),(c)",
     "otir    (c),(hl)","","","",""
3860 DATA "lddr    (de),(hl)","cpdr    a,(hl)","indr    (hl),(c)",
     "otdr    (c),(hl)","","","",""
3870 DATA "","","","","","","","",""
3880 DATA "","","","","","","","",""
3890 DATA "","","","","","","","",""
3900 DATA "","","","","","","","",""
3910 DATA "","","","","","","","",""
3920 DATA "","","","","","","","",""
3930 DATA "","","","","","","","",""
3940 DATA "","","","","","","","",""
3950 DATA "","","","","","","","",""
3960 DATA "","add     ix,bc","","","","","","",""
3970 DATA "","","","","","","","",""
3980 DATA "","add     ix,de","","","","","","",""
3990 DATA "","ld      ix,#*^","ld      *^,ix","inc     ix","","","","",""
4000 DATA "","add     ix,ix","ld      ix,*^","dec     ix","","","","",""
4010 DATA "","","","","inc     (ix+^)","dec     (ix+^)","ld      (ix+*),#^",""
4020 DATA "","add     ix,sp","","","","","","",""
4030 DATA "","","","","","","","ld      b,(ix+^)",""
4040 DATA "","","","","","","","ld      c,(ix+^)",""
4050 DATA "","","","","","","","ld      d,(ix+^)",""
4060 DATA "","","","","","","","ld      e,(ix+^)",""
4070 DATA "","","","","","","","ld      h,(ix+^)",""
4080 DATA "","","","","","","","ld      l,(ix+^)",""
4090 DATA "ld      (ix+^),b","ld      (ix+^),c","ld      (ix+^),d",
     "ld      (ix+^),e","ld      (ix+^),h","ld      (ix+^),l","",
     "ld      (ix+^),a"
4100 DATA "","","","","","","","ld      a,(ix+^)",""
4110 DATA "","","","","","","","add     a,(ix+^)",""
4120 DATA "","","","","","","","adc     a,(ix+^)",""
4130 DATA "","","","","","","","sub     a,(ix+^)",""
4140 DATA "","","","","","","","",""
4150 DATA "","","","","","","","and     a,(ix+^)",""
```

```
4160 DATA "","","","","","","xor     a,(ix+^)",""
4170 DATA "","","","","","","or      a,(ix+^)",""
4180 DATA "","","","","","","cp      a,(ix+^)",""
4190 DATA "","","","","","","","",""
4200 DATA "","","","!!!     spezialbefehl mit (ix+*)","","","","",""
4210 DATA "","","","","","","","",""
4220 DATA "","","","","","","","",""
4230 DATA "","pop     ix","","ex      (sp),ix","","push    ix","","",""
4240 DATA "","jmp     (ix)","","","","","","",""
4250 DATA "","","","","","","","",""
4260 DATA "","ld      sp,ix","","","","","","",""
4270 DATA "","","","","","","","",""
4280 DATA "","add     iy,bc","","","","","","",""
4290 DATA "","","","","","","","",""
4300 DATA "","add     iy,de","","","","","","",""
4310 DATA "","ld      iy,#*^","ld      *^,iy","inc     iy","","","","",""
4320 DATA "","add     iy,iy","ld      iy,*^","dec     iy","","","","",""
4330 DATA "","","","","inc     (iy+^)","dec     (iy+^)","ld      (iy+*),#'",'
4340 DATA "","add     iy,sp","","","","","","",""
4350 DATA "","","","","","","ld      b,(iy+^)",""
4360 DATA "","","","","","","ld      c,(iy+^)",""
4370 DATA "","","","","","","ld      d,(iy+^)",""
4380 DATA "","","","","","","ld      e,(iy+^)",""
4390 DATA "","","","","","","ld      h,(iy+^)",""
4400 DATA "","","","","","","ld      l,(iy+^)",""
4410 DATA "ld      (iy+^),b","ld      (iy+^),c","ld      (iy+^),d",
     "ld      (iy+^),e","ld      (iy+^),h","ld      (iy+^),l","",
     "ld      (iy+^),a"
4420 DATA "","","","","","","ld      a,(iy+^)",""
4430 DATA "","","","","","","add     a,(iy+^)",""
4440 DATA "","","","","","","adc     a,(iy+^)",""
4450 DATA "","","","","","","sub     a,(iy+^)",""
4460 DATA "","","","","","","sbc     a,(iy+^)",""
4470 DATA "","","","","","","and     a,(iy+^)",""
4480 DATA "","","","","","","xor     a,(iy+^)",""
4490 DATA "","","","","","","or      a,(iy+^)",""
4500 DATA "","","","","","","cp      a,(iy+^)",""
4510 DATA "","","","","","","","",""
4520 DATA "","","","!!!     spezialbefehl mit (iy+*)","","","","",""
```

```
4530 DATA "","","","","","","",""
4540 DATA "","","","","","","",""
4550 DATA "","pop     iy","","ex      (sp),iy","","push    iy","",""
4560 DATA "","jmp     (iy)","","","","","",""
4570 DATA "","","","","","","",""
4580 DATA "","ld      sp,iy","","","","","",""
4590 DATA "rlc     b","rlc     c","rlc     d","rlc     e","rlc     h",
     "rlc     l","rlc     (hl)","rlc     a"
4600 DATA "rrc     b","rrc     c","rrc     d","rrc     e","rrc     h",
     "rrc     l","rrc     (hl)","rrc     a"
4610 DATA "rl      b","rl      c","rl      d","rl      e","rl      h",
     "rl      l","rl      (hl)","rl      a"
4620 DATA "rr      b","rr      c","rr      d","rr      e","rr      h",
     "rr      l","rr      (hl)","rr      a"
4630 DATA "sla     b","sla     c","sla     d","sla     e","sla     h",
     "sla     l","sla     (hl)","sla     a"
4640 DATA "sra     b","sra     c","sra     d","sra     e","sra     h",
     "sra     l","sra     (hl)","sra     a"
4650 DATA "","","","","","","",""
4660 DATA "srl     b","srl     c","srl     d","srl     e","srl     h",
     "srl     l","srl     (hl)","srl     a"
4670 DATA "bit     0,b","bit     0,c","bit     0,d","bit     0,e",
     "bit     0,h","bit     0,l","bit     0,(hl)","bit     0,a"
4680 DATA "bit     1,b","bit     1,c","bit     1,d","bit     1,e",
     "bit     1,h","bit     1,l","bit     1,(hl)","bit     1,a"
4690 DATA "bit     2,b","bit     2,c","bit     2,d","bit     2,e",
     "bit     2,h","bit     2,l","bit     2,(hl)","bit     2,a"
4700 DATA "bit     3,b","bit     3,c","bit     3,d","bit     3,e",
     "bit     3,h","bit     3,l","bit     3,(hl)","bit     3,a"
4710 DATA "bit     4,b","bit     4,c","bit     4,d","bit     4,e",
     "bit     4,h","bit     4,l","bit     4,(hl)","bit     4,a"
4720 DATA "bit     5,b","bit     5,c","bit     5,d","bit     5,e",
     "bit     5,h","bit     5,l","bit     5,(hl)","bit     5,a"
4730 DATA "bit     6,b","bit     6,c","bit     6,d","bit     6,e",
     "bit     6,h","bit     6,l","bit     6,(hl)","bit     6,a"
4740 DATA "bit     7,b","bit     7,c","bit     7,d","bit     7,e",
     "bit     7,h","bit     7,l","bit     7,(hl)","bit     7,a"
4750 DATA "res     0,b","res     0,c","res     0,d","res     0,e",
     "res     0,h","res     0,l","res     0,(hl)","res     0,a"
```

```
4760 DATA "res 1,b","res 1,c","res 1,d","res 1,e",
     "res 1,h","res 1,l","res 1,(hl)","res 1,a"
4770 DATA "res 2,b","res 2,c","res 2,d","res 2,e",
     "res 2,h","res 2,l","res 2,(hl)","res 2,a"
4780 DATA "res 3,b","res 3,c","res 3,d","res 3,e",
     "res 3,h","res 3,l","res 3,(hl)","res 3,a"
4790 DATA "res 4,b","res 4,c","res 4,d","res 4,e",
     "res 4,h","res 4,l","res 4,(hl)","res 4,a"
4800 DATA "res 5,b","res 5,c","res 5,d","res 5,e",
     "res 5,h","res 5,l","res 5,(hl)","res 5,a"
4810 DATA "res 6,b","res 6,c","res 6,d","res 6,e",
     "res 6,h","res 6,l","res 6,(hl)","res 6,a"
4820 DATA "res 7,b","res 7,c","res 7,d","res 7,e",
     "res 7,h","res 7,l","res 7,(hl)","res 7,a"
4830 DATA "set 0,b","set 0,c","set 0,d","set 0,e",
     "set 0,h","set 0,l","set 0,(hl)","set 0,a"
4840 DATA "set 1,b","set 1,c","set 1,d","set 1,e",
     "set 1,h","set 1,l","set 1,(hl)","set 1,a"
4850 DATA "set 2,b","set 2,c","set 2,d","set 2,e",
     "set 2,h","set 2,l","set 2,(hl)","set 2,a"
4860 DATA "set 3,b","set 3,c","set 3,d","set 3,e",
     "set 3,h","set 3,l","set 3,(hl)","set 3,a"
4870 DATA "set 4,b","set 4,c","set 4,d","set 4,e",
     "set 4,h","set 4,l","set 4,(hl)","set 4,a"
4880 DATA "set 5,b","set 5,c","set 5,d","set 5,e",
     "set 5,h","set 5,l","set 5,(hl)","set 5,a"
4890 DATA "set 6,b","set 6,c","set 6,d","set 6,e",
     "set 6,h","set 6,l","set 6,(hl)","set 6,a"
4900 DATA "set 7,b","set 7,c","set 7,d","set 7,e",
     "set 7,h","set 7,l","set 7,(hl)","set 7,a"
```

DATAMAT AMSTRAD

DATAMAT er et komfortabelt dataforvaltningsprogram til AMSTRAD.
DATAMAT kan indeholde op til 512 tegn pr. »kort« og bruger ethvert felt som index (søge) felt. De kan søge, sortere og udvælge data efter alle kriterier og på alle felter, efter Deres ønske.
DATAMAT kan overføre data til TEXTOMAT således at disse bruges til f.eks. personlige serie-breve m.v.
DATAMAT er extrem hurtig da den er skrevet i 100% maskinkode.

DATAMAT i stikord

- ***Fuld menustyrring giver en hurtig og nemmere betjening.***
- ***Fri definerbar indgangsmaske.***
- ***512 tegn pr. datablok, max. 50 felter pr. blok.***
- ***Arbejder sammen med TEXTOMAT.***
- ***Arbejder sammen med 1 eller 2 diskettestationer.***
- ***Udprintning af lister og labels i fri format.***
- ***Data kan sorteres og selekteres.***
- ***Printer kan også tilsluttes gennem RS 232 porten.***

Kun kr. 498,-

AMSTRAD TEKSTBEHANDLINGS-PROGRAMMET

TEXTOMAT

TEXTOMAT er et menustyret tekstbehandlingsprogram. Alle printere kan faktisk tilsluttes via de indbyggede printertilpasningsrutiner. 10 frit definerbare styretegn til printer for max. udnyttelse af printerens muligheder, såsom fed, bred og grafik skrift. Teksten kan blokopdeles og flyttes rundt, som det nu passer bedst. Højre og venstre margin kan justeres ligesom overskrifter m.m. kan centreres.

Med data fra DATAMAT kan TEXTOMAT fremstille seriebreve med personligt tilsnit. Teksten formateres direkte på skærmen, således at det færdige resultat har lige højre og venstre margin.

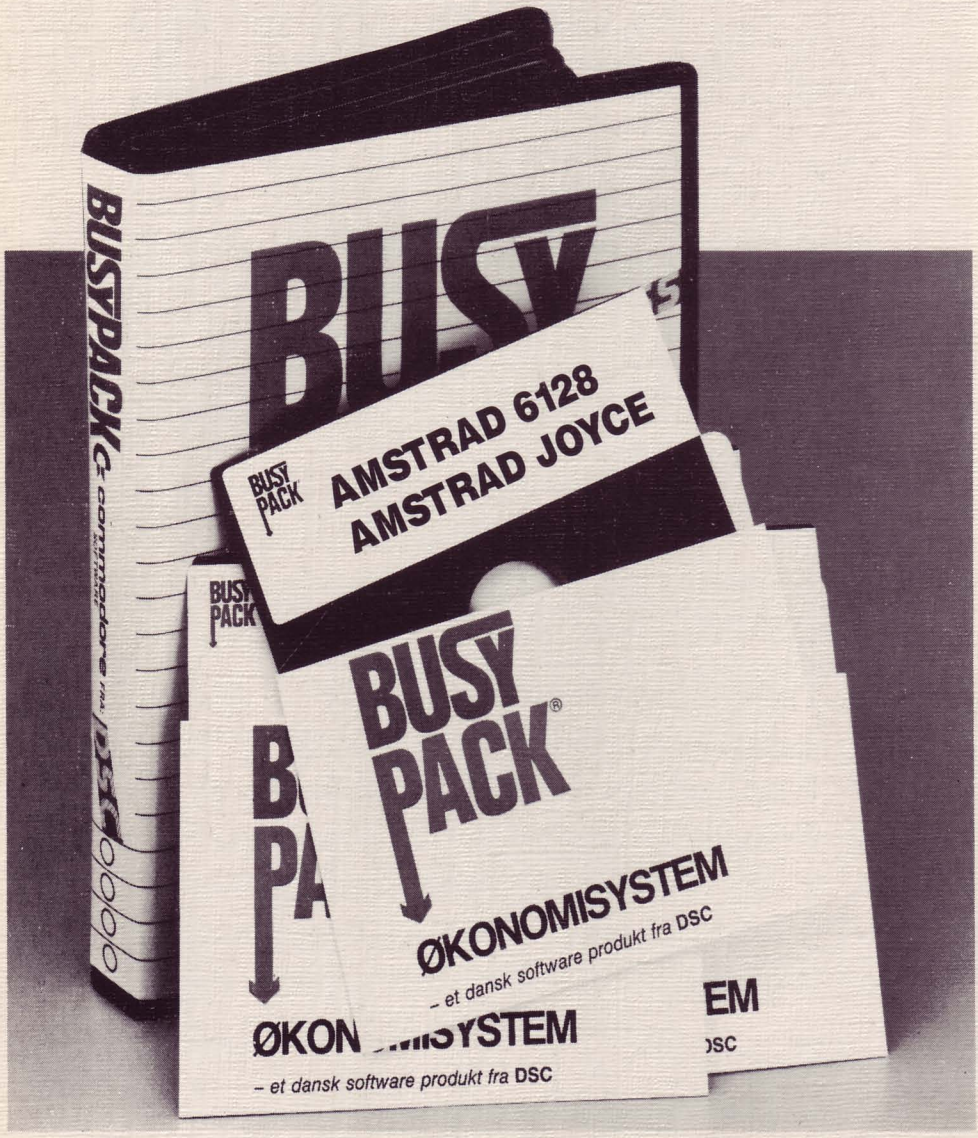
TEXTOMAT i stikord

- ***Fuld menustyring.***
- ***Regnefunktioner kan bruges i teksten.***
- ***40 eller 80 tegn pr. linie.***
- ***Kan køre med 2 diskettestationer.***
- ***Formattere teksten direkte på skærmen.***
- ***Næsten alle printere kan tilsluttes.***

Kun kr. 498,-

ADMINISTRATIVE PROGRAMMER TIL AMSTRAD 6128 OG JOYCE

Finans-bogholderi
Debitor- og kreditor-bogholderi
Fakturering
Lagerstyring



SE DET HOS DIN FORHANDLER ELLER KOM DIREKTE!
Nordic Computer Software, Smedegade 7, 6950 Ringkøbing

ISBN 87-7283-007-7