ABC KLUBBEN Publikationer m.m.

ABC-klubben har en publikationsserie döpt till **ABC-Rapporter.**

ABC-Rapport nr 1, den s k disassemblern, innehåller en listning av programvaran i ABC-80 med kommentarer, en verklig guldgruva för den som vill dyka djupare. Den är på c:a 320 sidor och kostar 100 Skr inkl frakt.

ABC-Rapport nr 2, Fig-FORTH on ABC-80 implemented by Robert Johnsen, juni 1982. Detta är en rapport på s:a 60 sidor skriven på engelska och innehåller bl a installations-manual för ABC-80 med köranvisningar och screenbeskrivningar samt utförlig referenslista. Dessutom innehåller rapporten utdrag i form av Glossary (verb-beskrivningar) och beskrivning av Fig-FORTH editorn. Rapporten kostar 60 Skr inkl frakt.

ABC-bladet, Samlingsnummer 1980-81, inkl ABC-kassetterna 1 och 2 är en sammanställning av ABC-bladets åtta nummer vilka utkom under klubbens första två år. Detta omfattar c:a 150 sidor och kostar 100 Skr inkl frakt.

ABC-bladet 1982, inkl ABC-kassetterna 3,4,5,6,7 och 8 är tillgänglig med alla sex kassetterna för 125 Skr.

ABC-klubben rabatterar för nya medlemmar vid köp av båda dessa samlingsnummer och tar då 210 Skr. Av praktiska skäl betraktas varje samlingsnummer för sig som en helhet. Om du är intresserad av någon av ABC-klubbens publikationer får Du dem enklast genom att sätta in beloppet på ABC-klubbens postgirokonto 62 93 00 - 5 och ange vad Du vill ha samt medlemsnummer och **en tydlig och läslig leveransadress.**

Ingen tillkommande moms

Postgironummer för ABC-klubbens publikationer

pg 62 93 00 - 5

Vid beställning över PG och rätta summan inget postförskottstillägg.

ABC-klubben har träffat avtal med Stockholms Datamaskinscentral QZ att medlemmarna skall kunna få köra på QZ:s DEC-10 dator till reducerad taxa under kvällstid då denna dator har viss överkapacitet. **Denna verksamhet kallas Q-Zentralen.** ABC-klubben har två olika kundnummer som medlemmarna får utnyttja, ett till lågtaxa där man endast kan köra **KOM** och då få tillgång till programbanken och ett kundnummer till en något högre taxa där man tämligen fritt får utnyttja DEC-10. Man betalar endast sin egen " session cost " till ABC-klubben som i sin tur står som garant gentemot QZ. Tills vidare är det fråga om en försöksverksamhet, men styrelsen verkar för att verksamheten skall bli permanent. Om Du vill utnyttja denna möjlighet att köra på QZ förfar Du på följande sätt: Betala in 50 kr i inträdesavgift på ABC-klubbens speciella postgirokonto för Q-Zentralen, **nummer 43 51 74 - 8.** Då får Du Dig tillsänt en ansvarsförbindelse, en manual, några inbetalningskort samt en kassett med erforderliga program och exempel på KOM-trafik. Så snart den ifyllda ansvarsförbindelsen kommit till klubben få Du password och sen är det bara att koppla upp och köra.

Gå med i ABC-klubben för att få ökat utbyte av Din dator och få tillgång till den stora erfarenhet som vi gemensamt bygger upp.

Medlem blir Du enklast genom att betala in medlemsavgiften på ABC-klubbens postgirokonto 15 33 36 - 3. Medlemsavgiften är för 1983, **125 Skr för seniorer och 70 Skr för juniorer** (junior är man t o m det kalenderår man fyller 18). Beställer Du någon av publikationerna, är vi tacksamma om Du använder särskillt inbetalningskort för detta.

ABC-klubben
Vidängsvägen 1
161 33 Bromma

Telefon:
08-80 15 22 (automatisk telefonsvarare)
08-80 15 23 (modem med ABC-Monitor)

Postgirokonton:

Medlemsavgifter	**: 15 33 36 - 3**
Publikationer	**: 62 93 00 - 5**
Q-Zentralen	**: 43 51 74 - 8**

Jr datorns synpunkt fungerar ABC800 mot kassettbandspelaren lite annorlunda än ABC80. Om man skriver LOAD CAS:FIL.BAC så läser datorn igenom hela kassetten tills den hittar den filen! Om den inte hittar den så stannar den bara när bandet tar slut. Detta är anledningen till att programmet frågar efter sista fil, annars står bara datorn och går på tomgång på satsen OPEN 'CAS:' AS .. när programmet inte hittar fler filer.

Också när man skriver på kassetten så stannar bandet om man slutar skriva. Detta gör också att om man läser ut en fil med ett program så lagras filen blockvis med en start- och stopp-sträcka mellan varje block. Däremot om man lagrar samma BAC-fil med ett SAVE-kommando så läggs filen ut utan blockmellanrum. Detta gör att programmet CASDISK.800 inte kan läsa längre BAC-filer som har lagrats med ett SAVE-kommando. Däremot kan programmet läsa ABC800-BAC filer som har lagrats med COPYLIB (både med ABC80-version och mot rande version för ABC800!). Jag har gjor en motsvarighet till COPYLIB för ABC800 som heter DISKCAS.800.

För övrigt finns ett mycket bra program för ABC800 i Monitorn som heter COPYFAST.800. Detta programmet kopierar en ett rent fysiskt rakt upp och ner, med älp av lite assember.

Programmet CASDISK kopierar alltså även randomfiler enligt ABC-kassett nr 7. Men för att programmet som läser in denna radomfil skall fungera, så måste det konverteras enligt Luxors manual BASIC II, sid 91, punkt 5.

Jag vill sluta med några synpunkter på bildskärmarna för ABC800. Först prisskillnaden, den är faktiskt 2500 kronor inklusive moms! Nu kommer det en ny färgskärm från Luxor. Men denna skärm är egentligen en skärm till ABC M(onokorom) som kan ge högupplösningsgrafik i färg.

Man kan tydligen inte ordna en dator med teckengenerator för teledata och som även kan kopplas om till 80 tecken. Även om man kan få högupplösningsgrafik i den nya färgskärmen så kan man ju inte få text i färg. Text i färg är faktiskt inte så dumt, när an har vant sig vid att använda den. Det blir ju en ny dimension på informationen. Nåväl detta är inte sagt som ett försvar för för 40 tecken. Det är ganska kasst med 40 tecken, men om man mest skriver program och inte text så kan man stå ut med det. Om man skall köra som ter al mot en annan dator då är det natu gtvis väldigt dumt med 40 tecken.

Om jag skall byta till en 80 teckens skärm, då måste jag förutom mera bildminne skaffa ett nytt rom som hanterar utskriften på skärmen, samt ett nytt VU-kort.

Bo Kullmar <1789>

VARNING ABC800 !

Om flexskive-DOS version 6.13 används för RAM-floppy skriver denna version på flera adresser i poke-arean (65024 - 65279) som skall vara ledig för egna rutiner.

Detta kan ställa till problem vid assemblerrutiner!

Lennart Jansson <620>

Autopilot på ABC800

av Björn Gustavsson

I ABC-bladet nr 2 1982 kunde man läsa om hur man kunde sätta autopilot på ABC80. Detta är mycket lätt att göra på ABC800, betydligt lättare än på ABC80. De som har gjort BASIC II har nämligen varit snälla nog att lägga vissa hopptabeller och pekare i RAM i stället för i ROM. Jag hoppas att kunna beskriva några fler användningsområden i kommande artiklar.
De intressanta adresserna i det här fallet är 255:144-146. (Adresser enligt Stockmans notation: 255*256+144.) Vid RESET lägger BASIC dit JP xxxx, där xxxx är adressen till rutinen i BASIC-ROMet som läser in ett tecken från tangentbordet. (Adressen är versionsberoende. På ABC800 M kan den vara 3:56.)
Det är då enkelt att ändra adressen så att den hoppar till en egen assemblerrutin. I assemblerrutinen härintill har jag gjort så att den helt enkelt hämtar ett tecken i taget från en text som ligger lagrad på ett lämpligt ställe i minnet.

Jag har gjort ett enkelt BASIC-program som laddar assembler-programmet, sköter lagringen av texten samt uppstart av autopiloten. Programmet består av 3 funktioner. Det är lätt att skriva egna program som anropar dessa. Man kan då t ex hämta texten från en fil.

I demoprogrammet härintill finns dessa funktioner före rad 200. På rad 200 börjar huvudprogrammet som läser in kommandon direkt från tangentbordet. Du kan prova att mata in följande:

```
NEW
10 ; CHR$(12%)"Demonstration av autopilot"
RUN
SAVE TEST
NEW
LOAD TEST
RUN
*
```

"*" betyder slut på texten och autopiloten sätter då igång att utföra ett kommando i taget.

När du skriver egna styrprogram används funktionerna så här (i samma ordning):

```
110 Z%=FNI%(<adress>)
```

" adress " är adressen där du vill ha texten lagrad. I mitt program används 63488 (dosbuffert 3). Måste användas före FNL%.

```
150 Z%=FNL%("<text>")
```

" text " är texten som skall lagras i textbufferten. FNL% kan användas obegränsat antal gånger. Observera dock att funktionen inte kontrollerar var texten hamnar. Är du oförsiktig kan t ex systemvariabler skrivas över.

```
210 Z%=FNS%
```

Detta kopplar in autopiloten. Nästa tecken som BASIC-tolken försöker läsa in hämtas från den inmatade texten. Observera dock att så länge BASIC-programmet exekveras försöker BASIC-tolken inte att läsa in några tecken (om programmet inte använder INPUT eller liknande).

Användningsområden: Jag har använt autopiloten när jag velat lista flera filer i en följd. Med hjälp av autopiloten angav jag 15 filer samtidigt och sysslade med annat när listaprogrammet arbetade. (Jag har ett program som listar BAS-filer i ett snyggare format än LIST PR: med rubrik, datum och sidnummer överst på varje sida. Med autopilotens hjälp är det också lätt att omvandla en BAC-fil till BAS-fil.)

```
*** ASSEMBLERPROGRAMMET ***

BEGIN   EQU 65024 ; Poke-arean
GCHJMP  EQU 65425
        ORG BEGIN
BEGTX   DEFW 63488 ; DOSBUF 3
START   LD HL,(GCHJMP)
        LD (SAVE),HL
        LD HL,GETTX
        LD (GCHJMP),HL
        LD HL,(BEGTX)
        LD (NEXT),HL
        RET
        ;
GETTX   PUSH HL
        LD HL,(NEXT)
        LD A,(HL)
        INC HL
        LD (NEXT),HL
        POP HL
        AND A
        RET NZ
        PUSH HL
        LD HL,(SAVE)
        LD (GCHJMP),HL
        EX (SP),HL
        RET
NEXT    DEFW BEGTX
SAVE    DEFW 0
        END
```

Assemblerprogrammet

```
 10 POKE -512%,0%,248%,42%,145%,
    255%,34%,44%,254%,33%,21%,25
    4%,34%,145%,255%
 20 POKE -498%,42%,0%,254%,34%,4
    2%,254%,201%,229%,42%,42%,25
    4%,126%,35%
 30 POKE -485%,34%,42%,254%,225%
    ,167%,192%,229%,42%,44%,254%
    ,34%,145%,255%
 40 POKE -472%,227%,201%,0%,254%
    ,0%,0%
100 DEF FNI%(X%)
110   Ö9%=X% : POKE -512%,Ö9%,SW
    AP%(Ö9%) : RETURN 0%
120 FNEND
130 DEF FNL%(A$)
140   FOR N%=1% TO LEN(A$) : POK
    E Ö9%,ASCII(MID$(A$,N%,1%))
    : Ö9%=Ö9%+1% : NEXT N%
150   RETURN 0%
160 FNEND
170 DEF FNS%
180   POKE Ö9%,0% : RETURN CALL(
    -510%)
190 FNEND
200 ; CHR$(12%) "*** ABC800 AUTO
    PILOT DEMO ***"
210 ;
220 Z%=FNI%(63488%) ! Lägg texte
    n i DOSBUF 3
230 INPUT LINE A$ : ; : A$=LEFT$
    (A$,LEN(A$)-1%)
240 IF A$<>"/"+CHR$(13%) Z%=FNL%
    (A$) : GOTO 230 ELSE Z%=FNS%
```

ABC-KASSETTERNA 7 & 8 UTGIVNA UNDER VERKSAMHETSÅRET 1982

Kassett sju

CASDISK2.BAS 14
Detta är en ny version av CASDISK som används för att kopiera filer med speciellt filslut enl ABC-klubbens standard. Detta slutmärke används vid kopiering av random-filer till kassett. En random-fil kan ej återläsas från kassett med det gamla CASDISK.

GODJUL82.BAS 3
Kan någon förklara det här fenomenet. Skriv då till ABC- Bladet, några av svaren kommer att belönas med trevlig present.

[illegible]AKOUT.BAS 16
[illegible]program som kan köras på grundutrustning.

SOLITÄR .BAS 24
Spelprogram som kan köras på grundutrustning.

[illegible]TTE .BAS 10
Spelprogram som kan köras på grundutrustning.

LUFTVÄRN.BAS 20
Spelprogram som kan köras på grundutrustning.

RUNONLY .BAS 7
Enkelt programmskydd. Programmet kan bara köras.

NOTONLY .BAS 6
Upphäver verkan av ovanstående program.

DEVELOP .TXT 10
I denna fil finns källkoden till maskinspråket i nedanstående program.

DEVELOP .BAS 9
Detta program är ett hjälpmedel vid programmering. Det är inte alls så avancerat som HJÄLPAREN på kassett 8. Men det är ändå intressant att studera. DEVELOP arbetar med kontrolltecken som kommandon. [illegible] kan t ex styra TRACE under körning [illegible] få grafiska tecken listade.

HÖGPREC .INF 4
Kort information om nedanstående 15 filer, vilka ingår i paketet med högprecisionsmatematik.

[illegible]OG1 .BAS 54
Huvudprogram i högprecisionsmatematiken, laddar in maskinspråk.

Denna fil innehåller maskinspråk till matematikpaketet. Filen är en random-fil, vilket innebär att den inte har vanligt filslutmärke. Filen har försetts med ett extra block som innehåller slutmärke enligt klubbens standard. Filer med slutmärke enl denna standard måste läsas in till skiva med CASDISK2.

CASFORTH.REM 16
Kort information om kassettversionen av FORTH. För mer utförlig information hänvisar vi till ABC-Rapport 2, som kostar 60:-.

FORTH i kassettversion
På denna ABC-kassett ligger CASFORTH .ABS, som kan köras på en ABC80 utan floppy. Den fungerar med eller utan 16 K extra minne. Vid inladdningen känner programmet av om man har utbyggt minne.

Kassettversionen utnyttjar delar av RAM-minnet till screens. I grundversionen finns det plats för 5 screens, med 16 K extra minne utökas detta antal till 26. Screens till FORTH har distribuerats som textfilerna SCREEN.TXT och SCREEN2.TXT på ABC-kassett nr 4 resp 5. SCREEN.C på denna kassett innehåller de screens man kanske i första hand vill ha för kassettbruk. De har hämtats från SCREEN.TXT och SCREEN2 .TXT i något fall har obetydlig ändring gjorts.

F n kan man inte ladda in screens medan man kör Forth, man måste därför först läsa in de screens man vill ha och sedan ladda in Forth. Det går också att tillfälligt lämna Forth för att läsa fil kan man läsa vidare från nästa fil. Svarar man RETURN på frågan om filnamn avslutar programmet med att nollställa ej ianspråktagna screenbuffertar.

Inladdning av CASFORTH sker med hjälp av CMDINT.CAS. Om inläsningen gått bra får man texten **Z80 FIG-FORTH 1.1**
Om Forth av någon anledning spårar ur så att man kommer tillbaka till Basic kan man återinträda genom att skriva ;CALL(53082) för kallstart och ;CALL(53086) för varmstart. Kallstart innebär detsamma som efter ny inladdning av Forth. Vid varmstart behåller man variabelvärden och Forthord som lagts till extra.

Man kan tillfälligt lämna Forth genom att ge kommandot $BAS (Sol BAS) och får då texten **Welcome back to FORTH.** Man kan nu ladda in ytterliggare screens med programmet INSCREEN och genom varmstarten göra återinträde i Forth. Screens man editerat kan sparas som textfil på kassett. Då lämnar man Forth med BYE och laddar sedan **utan att göra RESET** in programmet CASCRUT, det står för CAssett SCReen UT.

CASCREEN.BAS 10
Program som läser in screens från t ex SCREEN.C.

INSCREEN.BAS 6

SCREEN .C 38

CMDINT .CAS 5
Detta program gör det möjligt att köra ABS-filer på grundutrustningen.

CASFORTH.ABS 32
Kassettversion av FORTH - en unik möjlighet för den som bara har grundutrustning.

CASCRUT .BAS 7
Lägger screens på kassett t ex när man gjort egna nya screens. Avståndet mellan CASCRUT och föregående fil är något knappt beroende på utrymmesbrist på mastern.

	Postgirokonton:
Medlemsavgifter	: 15 33 36 - 3
Publikationer	: 62 93 00 - 5
Q-Zentralen	: 43 51 74 - 8

Kassett åtta

CASDISK2.BAS 14
Det nya program för överföring av filer från kassett, som fortsättningsvis kommer att inleda ABC-kassetten.

GENESIS .BAS 33
Den sanna sagan om hur det gick till när ABC80 föddes.

ROADRACE.BAC 63
Biltävling för alla, ingen risk för olyckor! Programmet är skrivet av Nils Häggblom en av våra medlemmar i Finland. Programmet kan trots sin storlek köras med 16K minne.

GODJUL2 .BAS 11
Stämmningsfullt grafikprogram med anknytning till en viss helg.

Kort information om CAS80.

CAS80 .BAC 12
Program som skapar två nya enheter; CS1: och CS2:.

Enheten CS1: fungerar som CAS: medan CS2: ger utskrift på kassett med blockgap om ca 2 sekunder. CAS80 kan användas tillsammans med TV-editorn för att ge en fil på kassett med blockgap. Vissa program t ex ASMCAS klarar inte av att bearbeta en vanlig kassettfil. Detta inträffar när programmets behandlingstid för ett block är så lång att kassettbufferterna inte hinner bli klara för inläsning innan ett nytt kassettblock kommer. Med större avstånd mellan blocken kan detta avhjälpas.

Programmet CAS80 länkar in två nya kassett enheter med namnen CS1: och CS2:. Det som skiljer dessa båda nya enheter från ABC80's egen (CAS:) är att vid öppning, läsning, skrivning och stängning så kommer det INTE att skickas ut något "skräp" på V24 kontakten.

CS1: fungerar exakt som CAS:. CS2: däremot läser och skriver med pauser mellan blocken och med start och stop av motorn. (Se Avanserad programmering på ABC80 Sid 50-53)

CS2: är därför mycket användbar tillsammans med T80PRT(ABCV24) och ABCTRANS (ej CASMINI). Vid mottagning av filer är det inget problem men vid sändning blir det lite annorlunda. Man måste nämligen först göra i ordning en kassett med de filer man vill sända.

Gör så här:

1 Kör CAS80.

2 Ladda in programmet som du vill sända.

3 Sätt i en ny kassett.

4 Gör LIST CS2:NNNNNNNN.EEE och programmet är nu inspelat med pauser mellan blocken.

OBS! När ABCTRANS frågar efter "Filnamn här" glöm då ej att skriva CS2: före filnamnet.

Johan Hedin < 525>
0758/40159

HJÄLPARE.REM 19
Bruksanvisning till HJÄLPAREN, programmet som först kom på kassett nr 4, men då endast för checksumma 11273. Nu kan vi presentera den utbyggda versionen för alla checksummor. Dessutom i ett synnerligen kompakt format. Detta lagringsformat för maskinkod har åstadkommits med hjälp av ASSTRAN-<G,H,L>. Ett programmpaket på denna kassett. De fyra HJÄLPARE-filerna nedan har fått korta namn, där A står för checksumma 11273 och B för checksummorna 10042 & 9913. För vidare information se ABC-bladet 3, 1982 sid 24.

HLPA16 .BAC 14
HJÄLPAREN för CS:11273 och 16K minne.

HLPA32 .BAC 14
JÄLPAREN för CS:11273 och 32K minne.

HLPB16 .BAC 14
HJÄLPAREN för CS:10042 & 9913 och 16K minne.

HLPB32 .BAC 14
'JÄLPAREN för CS:10042 & 9913 och 2K minne.

INFO .BAS 5
Detta program inleder en variant av matematikpaketet på kassett nr 7. Vad sägs om en ' fickräknare ' med 4000 siffrors noggrannhet? Programmet kräver dock 32K minne.

MASK .BAS 56
enna fil laddas in av INFO med CHAIN, måste ligga i BAS-format. MASK laddar in maskinspråk i minnet.

RPN .BAC 48
Denna fil hör ihop med INFO och MASK och laddas in efter MASK. RPN.BAC simulerar en kalkylator.

ASSTRANG.BAC 8
Gör om maskinspråk lagrat i ramminnet till kompakta BASIC-rader se någon av HLP-filerna på denna kassett.

ASSTRANH.BAC 12
i variant av ASSTRAN

ASSTRANL.BAC 12

GOLV .BAC 2
Program som höjer BOFA fem sektorer. Kan läggas i början av program, som ska ladda in maskinspråk i början av minnet.

SÄNK .BAC 2
Sänker taket genom att flytta stackpekaren nedåt.

SÄNKIF .BAC 2
Sänker taket men känner först efter om det finns en floppy i systemet.

GOLVTAK .ASM 14
Källkod till programmen GOLV, SÄNK och SÄNKIF.

ERRORSYS.BAC 13
Detta program används för att skapa programmet BASICERR.SYS med egna personliga felmedelanden.

BYECHAIN.BAS 20
Autostart av program, du kan få start av valfritt program med kommandot BYE. BYCHAIN lägger in namnet på valfritt program i CMDINT.SYS, som alltså måste finnas på skivan när du kör BYECHAIN.

En .ABS-fil är i själva verket ett maskinspråksprogram. Den reserverade filen CMDINT .SYS skiljer sig inte till strukturen från en vanlig .ABS-fil, men vid kommandot BYE börjar DOSet att söka efter filen, oberoende hurudant maskinspråksprogram den innehåller. Det är alltså fullt möjligt att döpa om t.ex. filen FORTH.ABS till CMDINT.SYS och få den att exekveras vid BYE.

Om man läser en .ABS-fil record för record kommer de 7 första bytena i recordet att vara information för DOSet. Dessutom finns en checksumma för den del av koden som ligger i det aktuella recordet.

Om informationen eller checksumman är galna tar maskinens självbevarelsekänsla vid och gör RST 00 på stället.

Tabell 3

Byte 1 : =0 => Läs information som följer
=255 => Till följande record
Byte 2 : Antal byte maskinspråk efter byte 8
Byte 3 : Oanvänd (=0)
Byte 4 : High-byte av adressen dit maskinspråksstumpen placeras
Byte 5 : Byte 4 XOR 255 (Kontroll)
Byte 6 : Low-byte
Byte 7 : Byte 6 XOR 255 (Kontroll)
Byte 8 : Maskin kod ....
.
.
.
Byte (8+Byte 2) : Checksumma av maskinkoden

Strukturen ovan upprepas inom recordet tills byte 1=255.
När antal byte=0 blir adressen strax efter startadressen för rutinen! Hopp dit sker omedelbarbums!!!

SCREEN3 .TXT 42
Nya screens till FORTH. Innehåller bland annat ett program som läser biblioteket på vanliga ABC80-disketter och hjälper dig att få ordning på dina disketter.

1983-01-15 kl 23.00

Inlägg som finns på ABC-Monitorn som GÖTEBORG.FRÅ

Hej! Mina telefonräkningar har blivit oerhörda sedan jag skaffade ett modem. Finns det inte en möjlighet att få en monitor i Göteborgstrakten?

< Insänt av 1298 >

Svaret vi lade in som GÖTEBORG.SVR:

I Göteborg planeras att ordna en lokalavdelning med ett konstituerande möte onsdagen den 9 februari 1983 kl 18.00 på Poetgatan 16, i Hisings Backa Klubbhuset Brf Göteborgshus 36

Kontaktman Carl Gunnar Hillefors <552>
tel 031-58 14 47
träffas säkrast jämn vecka
kvällstid fram till kl 21.00

Närmare besked kommer i ABC-bladet 1,1983

Avsikten är bl a att lösa Monitor- frågan för Västsverige och annan "lokal" klubbverksamhet.

Ulf Sjöstrand

< Insänt av 1208 >

FILELIB

```
* ABC-klubben Databank Library *

* 1983-01-15 kl 23.30          *

Drive 0

GÖTEBORG.SVA 4     GÖTEBORG.FRÅ 4
DOS      .SVR 4    HJÄLP    .DIG 4
DOS      .FRÅ 4    TEXTED   .FRÅ 8
FR2702   .800 4    COL80    .FRÅ 32
PD68     .   4     VEDIT    .FRT 12
ABCMÖTEN.INF 12    EXSCREEN.REM 8
EXSCREEN.BAS 8     GODJUL80.TKN 8
ASMEDIT2.SVR 4     FILTRANS.800 48
EGENTKN  .FRÅ 4    ABCMINI  .800 8
TEXTED   .SVR 8    PROTALL  .800 12
CASDISK  .800 40   BOK800   .TXT 8
ACKER    .FUN 8    RITA     .SUB 12
ELFASYST.FEL 4     ASMEDIT  .SVR 4
TILL1306.MSG 8     ABCTRANS.800 24
MODEMBUG.INF 8     KOLLDIM  .800 12
STARTRK2.SVR 4     UTSKRIFT.FEL 4
STARTREK.GAM 76    STARTREK.INF 4
TKN80LIB.SVR 8     STARTREK.SVR 8
TKN80LIB.FRÅ 4     ASMERR   .SVR 8
ARTREK.FRÅ 4       VÄSTMÖTE.INF 4
VARNING  .800 4    VÄSTUPP  .ROP 4
MENY     .800 16   COPYFAST.800 16
GÖRCAS80.BUG 4     GÖRCAS80.FEL 4
ASMEDIT  .FRÅ 4    ASMCAS   .SVR 8
V24      .SVR 8    DATABANK.MER 4
GÖTEBORG.SVR 4     FIGF     .SVR 4
GÖRCAS80.REM 8     GÖRCAS80.BAS 24
PRONLINE.SVR 8     ASMCAS   .FRÅ 4
HARDCOPY.800 8     DIVERSE  .FRÅ 4
PROMPROG.SLJ 4     LIB      .FRT 4
CLOSOPEN.REM 8     KASSABOK.REM 8
OPEN     .ASM 32   CLOSE    .ASM 16
QARITM   .FRT 8    FLEXENH  .KÖP 4
STACKMAN.FRT 8     MAZECPM  .PAS 16
ELFASYST.FRÅ 4     ORDER    .INF 4
ORDER    .BAS 12   PRLISTA  .BAS 16
PROTALL  .BAS 12   CHANGE   .UTL 12
SNABELA  .SVR 4    DATUM    .BAS 12
FIGFORTH.FRÅ 4     MX80     .FRÅ 4
DIMFIX   .SUB 12   EDITORN  .INF 8
ABC800   .FRÅ 4    KASSA    .CPM 44
PROMPROG.SÄL 4     DATABANK.INF 12
ELFASYST.INF 32    ABC800   .SVR 4
FILELIB  .HLP 4    ABCV24   .INF 4
BANDSPEL.800 8     FILNAMN  .SVR 4
LOGOUT   .SVR 4    HELP     .TXT 4
BILL32K  .TXT 4    SKÖNSKRI.FRÅ 4
LOGOUT   .INF 8    STYRELSE.SVR 8
FELEN    .INF 8    FILTRANS.80K 32
STORTEXT.GRF 20    STORTEXT.REM 8
EXTRAKN  .ÄCK 4    FORTHBOK.SVR 4
MORSETEK.BAS 64    TILL455  .MSG 8
DOSCREEN.BUG 4     SCREEN   .31  4
GÖRITH   .BAS 8    GÖR      .ABS 8
EPSON    .UPG 4    MERAKUL  .1X2 8
UTSKRIFT.BAS 8     INFO1X27.PGM 8
SPAR     .FTN 24   TSM      .FTN 24
UM2      .FTN 24   LOTTO576.VAL 28
STRYK12X.123 32    L8RADER  .ENK 24
LOPOFLEX.BAS 12    TK2      .INF 8
LIBLIST  .UTL 12   LISTKOD  .BAS 8
QZANVISN.TXT 16    QZANSVAR.TXT 12
LÄSDISK2.REM 12    LÄSDISK2.BAS 48
TEST2    .BAS 48   TKN80LIB.BAS 12
LOGAIN2  .REM 12   LOGAIN2  .BAS 24
CHECKSUM.FRT 8     NOT      .ZZZ 4
QZFRÅGA  .SVR 4    QZENTRAL.INF 12
FILELIB  .INF 8    QZKOM    .INF 8
CURSOR   .INF 12   ABCSPRÅK.TST 8
DIKT     .TXT 8    SAFT     .BAS 24
NYTT     .ZZZ 8    CASTRANS.800 12
WRITE    .YES 4    SYSDIR   .SYS 4
```

Figur 3

```
Cas ñ 5
SOLFLÖJT (Gunilla von Bahr)

BELLMAN IN BRITAIN (Martin Best)

     SOLFLÖJT 2 (Gunilla von Bahr)
     VIVALDI      Piccolo Conderto P78
     BACH         Air for Suite No. 3
     MOZART       Andante KV 315
     GRIEG        Våren/ The Spring
     SOMMERFELDT Spring Tunes
     RACHMANINOV Vocalise
     von KOCH     Cantilena
     ++++++++++++++++++++++++++++++++++++
     BELLMAN IN BRITAIN (Martin Best)
     Systrar, hören min musik * Vila vid
     denna källa * Solen glimmar blank
     och trind * Hade jag sextusende
     daler * Gråt, fader Berg, och spela
     Se, god dag, min vän, min frände *
     Vår Ulla låg i säng och sov
     Portugal, Spanien
```

Ctrl-W	Cursor uppåt
Ctrl-Z	Cursor nedåt
Ctrl-A	Cursor vänster
Ctrl-S	Cursor höger
RETURN	Start ny rad
<-	Ta bort tecken (delete ch)
->	Stoppa in tecken (insert ch)
Ctrl-C	Ta bort rad (delete line)
Ctrl-B	Stoppa in rad (insert line)
Crtl-É	Gå ur editorn (exit)(Ctrl + ASCII 94 eller 126)
Ctrl-<	Gå ur samt spara på diskett (exit+flush)

Här följer de tre editor-screenerna, knappa in dem och var noga med att skriva alla tecken rätt. Sedan är det dags för provkörning.

Skriver man dem som en textfil kan den läsas in med DOSCREEN. Denna textfil kommer att läggas in på Monitorn under namnet VEDIT.FRT.

Lycka till!

Ulf Johnsen

< 674 >
Ludvika

För FORTH-fantomer

När ABC-kassetten med FORTH på ramlade ner i postlådan var det säkert många av oss som nyfiket kastade oss över den för att lära något om FORTH. Små programsnuttar skrivs, provkörs, ändras, provkörs o.s.v. Det som man mest konfronteras med är Editorn.

Med ABC-FORTH följde tre editorer, fig-editorn, poly-editorn och VEDIT. De två första är rad-orienterade, VEDIT är skärmorienterad. De radorienterade editorerna kan knäcka den mest entusiastiske programmerare. Man måste hålla reda på radnummer, cursorrad och annat som rätteligen datorn borde klara av själv. De är helt enkelt hopplöst omoderna. Den skärmorienterade VEDIT är ett steg i rätt riktning men den saknar några viktiga funktioner som måste finnas för att göra användandet riktigt trevligt. Jag har därför kompletterat VEDIT med några användbara kommandon.

Tyvärr är mina kunskaper i FORTH inte särskilt djupa, det är för att bättra på detta som editorn behövs, så ta inte programmet som exempel på bra programmeringsteknik. Det går säkert att göra editorn på fiffigare sätt och med flera finesser. Möjlighet att kopiera screener med olika nummer är en bra finess. Men om man belastar editorn med för många olika kommandon blir det lätt svåröverskådligt och problem med att komma ihåg alla kommandon. Har man 80-teckens skärm kan man ju utnyttja skärmens högra halva för en meny eller för listning av valfri annan screen som man vill ha till hands för jämförelse t ex.

Det går åt tre screener för editorn, numren spelar ingen roll, men ta tre i följd som är lediga. I exemplet används screen 70,71 och 72. Skriv in texten, kör FLUSH för att spara dem på diskett och skriv sedan "screennummer" LOAD (t.ex. 70 LOAD). Förhoppningsvis svarar datorn ok och editorn är klar att användas. Välj ut en screen, t.ex. 74, att öva på och skriv 74 ED. Screen 74 kommer då upp på skärmen och följande kommandon kan användas :

```
ñ 70
( SCREEN-ORIENTED EDITOR 'ED'  ñ1 )
: (CASE) OVER = IF DROP 1 ELSE 0
  THEN ;
: CASE COMPILE (CASE) /COMPILE/ IF ;
  IMMEDIATE
0 VARIABLE CUR
: YXCUR  ( x y -- )
  YCUR C" XCUR C" ;
: .CUR ( -- )
  Rñ ' C/L /MOD 2+ SWAP 3 + SWAP
  YXCUR ;
: "CUR ( n -- )
  0 MAX 755 MIN Rñ " ;
: +CUR ( n -- ) Rñ ' + "CUR ;
: +.CUR ( n-- ) +CUR .CUR ;
: +LIN ( -- )
  Rñ ' C/L / 1+ C/L * "CUR ;
: HOME ( -- ) 0 Rñ " ;
: LIMITS ( -- ) 21 0 DO C/L 3 +
  I 2+ YXCUR 127 EMIT LOOP ;
: BUFF SCR ' BLOCK ; ( n - adr) -->

ñ 71
( EDITOR 'ED'                  ñ2 )
: "BLK BUFF Rñ ' + C" UPDATE 1 +.CUR
  ; ( n -- )
: ENDLIN Rñ ' C/L / 1 + C/L * 2 - ;
  ( -- "adr" )
: IC Rñ ' 1 - ENDLIN DO BUFF I + '
  DUP I 1+ Rñ " .CUR EMIT "BLK -1
  .CUR +LOOP Rñ ' 1 - Rñ " -1
  +.CUR 32 EMIT 32 "BLK -1 +.CUR ;
: DC ENDLIN 1 + Rñ ' DO BUFF I + 1+
  ' DUP I Rñ " .CUR EMIT "BLK 1 .CUR
  LOOP ENDLIN 1 + Rñ " .CUR 32 EMIT
  32 "BLK CUR ' Rñ " .CUR ;
: IL ENDLIN 2 + C/L - BUFF + DUP C/L
  + B/BUF ENDLIN 3 + - <CMOVE ENDLIN
  2 + C/L - BUFF + C/L BLANKS PAGE
  SCR ' LIST LIMITS .CUR ;
: DL ENDLIN 2 + BUFF + DUP C/L -
  B/BUF ENDLIN 3 + - CMOVE 756 C/L
  - BUFF + C/L BLANKS PAGE SCR '
  LIST LIMITS .CUR ;          -->

ñ 72
( EDITOR 'ED'                  ñ3 )
: ED ( scr -- )
  PAGE LIST LIMITS HOME .CUR BEGIN
KEY 0 CASE 0 23 YXCUR QUIT ELSE
    9 CASE IC ELSE ( insert char )
    1 CASE -1 +.CUR ELSE ( left)
   26 CASE C/L +.CUR ELSE ( down )
   23 CASE C/L MINUS +.CUR ELSE ( u)
   19 CASE 1 +.CUR ELSE ( right )
   13 CASE 1 +LIN .CUR ELSE ( cr )
    8 CASE Rñ ' CUR " DC ELSE ( d c)
    2 CASE IL ELSE ( ins line )
    3 CASE DL ELSE ( del line )
  127 CASE 0 23 YXCUR FLUSH QUIT
      ELSE  ( Exit with flush )
      DUP DUP 32 < SWAP 127 > +
      IF DROP 32 THEN DUP EMIT "BLK
 THEN THEN THEN THEN THEN THEN THEN
 THEN THEN THEN THEN AGAIN ; ;S
** 'ED' Improved version of VEDIT **
** 82-12-12 by Ulf Johnsen  x674x **
```

Postgirokonton:

Medlemsavgifter	**: 15 33 36 - 3**
Publikationer	**: 62 93 00 - 5**
Q-Zentralen	**: 43 51 74 - 8**

Här finns programmen

Registerprogrammen finns på ABC-kassett nr 8 i form av screens i filen SCREEN3.TXT. Använd DOSCREEN för att flytta över dessa screens till en Forth systemskiva. Reservera plats för minst 97 screens (= det högsta nummer som finns i SCREEN3 .TXT). Musikkassettprogrammen ligger på screens nr 85 och 86. LIBRARY-programmet finns på screens 89, 90, 91, 92, 93, 94 och 95, samt utnyttjar även nr 31, 63, 87 och 88 samt ASM (assembler). Paketet laddas genom 89 LOAD. Den screen 31 som användes är en modifierad version skriven i Forth assembler. Genom ett förbiseende kom denna inte med i filen SCREEN3.TXT varför den återges härintill. Den finns även på Monitorn under filnamnet SCREEN.31.

Gunnar Tidner,<1306>

```
ñ 31
( READ- WRITE- SECTOR                )
 ASM
CODE CALL.READ D POP B PUSH 24678
  CALL B POP NEXT JMP END-CODE
CODE CALL.WRITE D POP B PUSH 24675
  CALL B POP NEXT JMP END-CODE
: INIT
  OFFSET ' IF 1 ELSE 0 THEN DRIVE "
  SET-DRIVE 8 * + 32 * ;
: READ.SECTOR
  ( sector, track -- buff.addr     )
  INIT CALL.READ 64786 ' 64789 C'
  DISK-ERROR " ?DISK ;
: WRITE.SECTOR
( addr, sector, track --           )
  INIT SWAP 64786 ' 256 CMOVE
   CALL.WRITE 64789 C'
  DISK-ERROR " ?DISK ;
;S
```

FORTH HANDY REFERENCE

Stack inputs and outputs are shown; top of stack on right
This card follows usage of the Forth Interest Group (S.F. Bay Area); usage aligned with the *Forth 78* International Standard.
For more info: Forth Interest Group
P.O. Box 1105
San Carlos, CA 94070.

Operand key:

n, n1, ...	16-bit signed numbers
d, d1, ...	32-bit signed numbers
u	16-bit unsigned number
addr	address
b	8-bit byte
c	7-bit ascii character value
f	boolean flag

STACK MANIPULATION

DUP	(n → n n)	Duplicate top of stack.
DROP	(n →)	Throw away top of stack.
SWAP	(n1 n2 → n2 n1)	Reverse top two stack items.
OVER	(n1 n2 → n1 n2 n1)	Make copy of second item on top.
ROT	(n1 n2 n3 → n2 n3 n1)	Rotate third item to top
-DUP	(n → n ?)	Duplicate only if non-zero.
>R	(n →)	Move top item to "return stack" for temporary storage (use caution).
R>	(→ n)	Retrieve item from return stack.
R	(→ n)	Copy top of return stack onto stack.

NUMBER BASES

DECIMAL	(→)	Set decimal base.
HEX	(→)	Set hexadecimal base
BASE	(→ addr)	System variable containing number base.

ARITHMETIC AND LOGICAL

+	(n1 n2 → sum)	Add.
D+	(d1 d2 → sum)	Add double-precision numbers.
-	(n1 n2 → diff)	Subtract (n1-n2).
*	(n1 n2 → prod)	Multiply.
/	(n1 n2 → quot)	Divide (n1/n2).
MOD	(n1 n2 → rem)	Modulo (*i.e.*, remainder from division).
/MOD	(n1 n2 → rem quot)	Divide, giving remainder and quotient.
*/MOD	(n1 n2 n3 → rem quot)	Multiply, then divide (n1*n2/n3), with double-precision intermediate.
*/	(n1 n2 n3 → quot)	Like */MOD, but give quotient only.
MAX	(n1 n2 → max)	Maximum.
MIN	(n1 n2 → min)	Minimum.
ABS	(n → absolute)	Absolute value.
DABS	(d → absolute)	Absolute value of double-precision number.
MINUS	(n → -n)	Change sign.
DMINUS	(d → -d)	Change sign of double-precision number.
AND	(n1 n2 → and)	Logical AND (bitwise).
OR	(n1 n2 → or)	Logical OR (bitwise).
XOR	(n1 n2 → xor)	Logical exclusive OR (bitwise).

COMPARISON

<	(n1 n2 → f)	True if n1 less than n2.
>	(n1 n2 → f)	True if n1 greater than n2.
=	(n1 n2 → f)	True if top two numbers are equal.
0<	(n → f)	True if top number negative.
0=	(n → f)	True if top number zero (*i.e.*, reverses truth value).

MEMORY

@	(addr → n)	Replace word address by contents.
!	(n addr →)	Store second word at address on top.
C@	(addr → b)	Fetch one byte only.
C!	(b addr →)	Store one byte only.
?	(addr →)	Print contents of address.
+!	(n addr →)	Add second number on stack to contents of address on top.
CMOVE	(from to u →)	Move u bytes in memory.
FILL	(addr u b →)	Fill u bytes in memory with b, beginning at address.
ERASE	(addr u →)	Fill u bytes in memory with zeroes, beginning at address.
BLANKS	(addr u →)	Fill u bytes in memory with blanks, beginning at address.

CONTROL STRUCTURES

DO ... LOOP	do: (end+1 start →)	Set up loop, given index range.
I	(→ index)	Place current index value on stack.
LEAVE	(→)	Terminate loop at next LOOP or +LOOP.
DO ... +LOOP	do: (end+1 start →) +loop: (n →)	Like DO ... LOOP, but adds stack value (instead of always '1') to index
IF ... (true) ... ENDIF	if: (f →)	If top of stack true (non-zero), execute. [Note: *Forth 78* uses IF ... THEN]
IF ... (true) ... ELSE ... (false) ... ENDIF	if: (f →)	Same, but if false, execute ELSE clause. [Note: *Forth 78* uses IF ... ELSE ... THEN]
BEGIN ... UNTIL	until: (f →)	Loop back to BEGIN until true at UNTIL. [Note *Forth 78* uses BEGIN ... END]
BEGIN ... WHILE ... REPEAT	while: (f →)	Loop while true at WHILE; REPEAT loops unconditionally to BEGIN [Note *Forth 78* uses BEGIN ... IF ... AGAIN]

TERMINAL INPUT-OUTPUT

.	(n →)	Print number.
.R	(n fieldwidth →)	Print number, right-justified in field.
D.	(d →)	Print double-precision number
D.R	(d fieldwidth →)	Print double-precision number, right-justified in field.
CR	(→)	Do a carriage return
SPACE	(→)	Type one space.
SPACES	(n →)	Type n spaces
."	(→)	Print message (terminated by ")
DUMP	(addr u →)	Dump u words starting at address.
TYPE	(addr u →)	Type string of u characters starting at address
COUNT	(addr → addr+1 u)	Change length-byte string to TYPE form.
?TERMINAL	(→ f)	True if terminal break request present
KEY	(→ c)	Read key, put ascii value on stack.
EMIT	(c →)	Type ascii value from stack
EXPECT	(addr n →)	Read n characters (or until carriage return) from input to address
WORD	(c →)	Read one word from input stream, using given character (usually blank) as delimiter

INPUT-OUTPUT FORMATTING

NUMBER	(addr → d)	Convert string at address to double-precision number.
<#	(→)	Start output string.
#	(d → d)	Convert next digit of double-precision number and add character to output string.
#S	(d → 0 0)	Convert all significant digits of double-precision number to output string.
SIGN	(n d → d)	Insert sign of n into output string.
#>	(d → addr u)	Terminate output string (ready for TYPE)
HOLD	(c →)	Insert ascii character into output string.

DISK HANDLING

LIST	(screen →)	List a disk screen.
LOAD	(screen →)	Load disk screen (compile or execute)
BLOCK	(block → addr)	Read disk block to memory address.
B/BUF	(→ n)	System constant giving disk block size in bytes.
BLK	(→ addr)	System variable containing current block number
SCR	(→ addr)	System variable containing current screen number.
UPDATE	(→)	Mark last buffer accessed as updated.
FLUSH	(→)	Write all updated buffers to disk.
EMPTY-BUFFERS	(→)	Erase all buffers.

DEFINING WORDS

: xxx	(→)	Begin colon definition of xxx.
;	(→)	End colon definition.
VARIABLE xxx	(n →) xxx: (→ addr)	Create a variable named xxx with initial value n, returns address when executed.
CONSTANT xxx	(n →) xxx: (→ n)	Create a constant named xxx with value n; returns value when executed.
CODE xxx	(→)	Begin definition of assembly-language primitive operation named xxx.
;CODE	(→)	Used to create a new defining word, with execution-time "code routine" for this data type in assembly.
<BUILDS . . . DOES>	does: (→ addr)	Used to create a new defining word, with execution-time routine for this data type in higher-level Forth.

VOCABULARIES

CONTEXT	(→ addr)	Returns address of pointer to context vocabulary (searched first).
CURRENT	(→ addr)	Returns address of pointer to current vocabulary (where new definitions are put)
FORTH	(→)	Main Forth vocabulary (execution of FORTH sets CONTEXT vocabulary).
EDITOR	(→)	Editor vocabulary; sets CONTEXT
ASSEMBLER	(→)	Assembler vocabulary; sets CONTEXT
DEFINITIONS	(→)	Sets CURRENT vocabulary to CONTEXT
VOCABULARY xxx	(→)	Create new vocabulary named xxx.
VLIST	(→)	Print names of all words in CONTEXT vocabulary

MISCELLANEOUS AND SYSTEM

(	(→)	Begin comment terminated by right paren on same line, space after (
FORGET xxx	(→)	Forget all definitions back to and including xxx
ABORT	(→)	Error termination of operation
'xxx	(→ addr)	Find the address of xxx in the dictionary, if used in definition, compile address
HERE	(→ addr)	Returns address of next unused byte in the dictionary
PAD	(→ addr)	Returns address of scratch area (usually 68 bytes beyond HERE)
IN	(→ addr)	System variable containing offset into input buffer used, e.g., by WORD
SP@	(→ addr)	Returns address of top stack item
ALLOT	(n →)	Leave a gap of n bytes in the dictionary
,	(n →)	Compile a number into the dictionary

FORTH-79 HANDY REFERENCE

Stack inputs and outputs are shown; top of stack on right. See operand key at bottom.

STACK MANIPULATION

DUP	(n → n n)	Duplicate top of stack.
DROP	(n →)	Discard top of stack.
SWAP	(n1 n2 → n2 n1)	Exchange top two stack items.
OVER	(n1 n2 → n1 n2 n1)	Make copy of second item on top.
ROT	(n1 n2 n3 → n2 n3 n1)	Rotate third item to top. "rote"
PICK	(n1 → n2)	Copy n1-th item to top. (Thus 1 PICK = DUP , 2 PICK = OVER)
ROLL	(n →)	Rotate n-th item to top. (Thus 2 ROLL = SWAP , 3 ROLL = ROT)
?DUP	(n → n (n))	Duplicate only if non-zero. "query-dup"
>R	(n →)	Move top item to "return stack" for temporary storage (use caution). "to-r"
R>	(→ n)	Retrieve item from return stack. "r-from"
R@	(→ n)	Copy top of return stack onto stack. "r-fetch"
DEPTH	(→ n)	Count number of items on stack.

COMPARISON

<	(n1 n2 → flag)	True if n1 less than n2. "less-than"
=	(n1 n2 → flag)	True if top two numbers are equal. "equals"
>	(n1 n2 → flag)	True if n1 greater than n2. "greater-than"
0<	(n → flag)	True if top number negative. "zero-less"
0=	(n → flag)	True if top number zero. (Equivalent to NOT) "zero-equals"
0>	(n → flag)	True if top number greater than zero. "zero-greater"
D<	(d1 d2 → flag)	True if d1 less than d2. "d-less-than"
U<	(un1 un2 → flag)	Compare top two items as unsigned integers. "u-less-than"
NOT	(flag → ¬flag)	Reverse truth value. (Equivalent to 0=)

ARITHMETIC AND LOGICAL

+	(n1 n2 → sum)	Add. "plus"
D+	(d1 d2 → sum)	Add double-precision numbers. "d-plus"
−	(n1 n2 → diff)	Subtract (n1−n2). "minus"
1+	(n → n+1)	Add 1 to top number. "one-plus"
1−	(n → n−)	Subtract 1 from top number. "one-minus"
2+	(n → n+2)	Add 2 to top number. "two-plus"
2−	(n → n−2)	Subtract 2 from top number. "two-minus"
*	(n1 n2 → prod)	Multiply. "times"
/	(n1 n2 → quot)	Divide (n1/n2). (Quotient rounded toward zero) "divide"
MOD	(n1 n2 → rem)	Modulo (*i.e.*, remainder from division n1/n2). Remainder has same sign as n1. "mod"
/MOD	(n1 n2 → rem quot)	Divide, giving remainder and quotient. "divide-mod"
*/MOD	(n1 n2 n3 → rem quot)	Multiply, then divide (n1*n2/n3), with double-precision intermediate. "times-divide-mod"
*/	(n1 n2 n3 → quot)	Like */MOD , but give quotient only, rounded toward zero. "times-divide"
U*	(un1 un2 → ud)	Multiply unsigned numbers, leaving unsigned double-precision result. "u-times"
U/MOD	(ud un → urem uquot)	Divide double number by single, giving remainder and quotient, all unsigned. "u-divide-mod"
MAX	(n1 n2 → max)	Leave greater of two numbers. "max"
MIN	(n1 n2 → min)	Leave lesser of two numbers. "min"
ABS	(n → |n|)	Absolute value. "absolute"
NEGATE	(n → −n)	Leave two's complement.
DNEGATE	(d → −d)	Leave two's complement of double-precision number. "d-negate"
AND	(n1 n2 → and)	Bitwise logical AND.
OR	(n1 n2 → or)	Bitwise logical OR.
XOR	(n1 n2 → xor)	Bitwise logical exclusive-OR. "x-or"

MEMORY

@	(addr → n)	Replace address by number at address. "fetch"
!	(n addr →)	Store n at addr. "store"
C@	(addr → byte)	Fetch least significant byte only. "c-fetch"
C!	(n addr →)	Store least signficant byte only. "c-store"
?	(addr →)	Display number at address. "question-mark"
+!	(n addr →)	Add n to number at addr. "plus-store"
MOVE	(addr1 addr2 n →)	Move n numbers starting at addr1 to memory starting at addr2, if n>0.
CMOVE	(addr1 addr2 n →)	Move n bytes starting at addr1 to memory starting at addr2, if n>0. "c-move"
FILL	(addr n byte →)	Fill n bytes in memory with byte beginning at addr, if n>0.

CONTROL STRUCTURES

DO . . . LOOP	do: (end+1 start →)	Set up loop, given index range.
I	(→ index)	Place current loop index on data stack.
J	(→ index)	Return index of next outer loop in same definition.
LEAVE	(→)	Terminate loop at next LOOP or +LOOP , by setting limit equal to index.
DO . . . +LOOP	do: (limit start →) +loop: (n →)	Like DO . . . LOOP , but adds stack value (instead of always 1) to index. Loop terminates when index is greater than or equal to limit (n>0), or when index is less than limit (n<0). "plus-loop"
IF . . . (true) . . . THEN	if: (flag →)	If top of stack true, execute.
IF . . . (true) . . . ELSE . . . (false) . . . THEN	if: (flag →)	Same, but if false, execute ELSE clause.
BEGIN . . . UNTIL	until: (flag →)	Loop back to BEGIN until true at UNTIL .
BEGIN . . . WHILE . . . REPEAT	while: (flag →)	Loop while true at WHILE; REPEAT loops unconditionally to BEGIN . When false, continue after REPEAT .
EXIT	(→)	Terminate execution of colon definition. (May not be used within DO . . . LOOP)
EXECUTE	(addr →)	Execute dictionary entry at compilation address on stack (*e.g.*, address returned by FIND).

Operand key:

n, n1, . . .	16-bit signed numbers	d, d1, . . .	32-bit signed numbers	addr, addr1, . . .	addresses	char	7-bit ascii character value
		u	unsigned	byte	8-bit byte	flag	boolean flag

TERMINAL INPUT-OUTPUT

CR	(→)	Do a carriage return and line feed. "c-r"
EMIT	(char →)	Type ascii value from stack.
SPACE	(→)	Type one space.
SPACES	(n →)	Type n spaces, if n>0.
TYPE	(addr n →)	Type string of n characters beginning at addr, if n>0.
COUNT	(addr → addr+1 n)	Change address of string (prefixed by length byte at addr) to TYPE form.
-TRAILING	(addr n1 → addr n2)	Reduce character count of string at addr to omit trailing blanks. "dash-trailing"
KEY	(→ char)	Read key and leave ascii value on stack.
EXPECT	(addr n →)	Read n characters (or until carriage return) from terminal to address, with null(s) at end.
QUERY	(→)	Read line of up to 80 characters from terminal to input buffer.
WORD	(char → addr)	Read next word from input stream using char as delimiter, or until null. Leave addr of length byte.

NUMERIC CONVERSION

BASE	(→ addr)	System variable containing radix for numeric conversion.
DECIMAL	(→)	Set decimal number base.
.	(n →)	Print number with one trailing blank and sign if negative. "dot"
U.	(un →)	Print top of stack as unsigned number with one trailing blank. "u-dot"
CONVERT	(d1 addr1 → d2 addr2)	Convert string at addr1+1 to double number. Add to d1 leaving sum d2 and addr2 of first non-digit.
<#	(→)	Start numeric output string conversion. "less-sharp"
#	(ud1 → ud2)	Convert next digit of unsigned double number and add character to output string. "sharp"
#S	(ud → 0 0)	Convert all significant digits of unsigned double number to output string. "sharp-s"
HOLD	(char →)	Add ascii char to output string.
SIGN	(n →)	Add minus sign to output string if n<0.
#>	(d → addr n)	Drop d and terminate numeric output string, leaving addr and count for TYPE. "sharp-greater"

MASS STORAGE INPUT/OUTPUT

LIST	(n →)	List screen n and set SCR to contain n.
LOAD	(n →)	Interpret screen n, then resume interpretation of the current input stream.
SCR	(→ addr)	System variable containing screen number most recently listed.
BLOCK	(n → addr)	Leave memory address of block, reading from mass storage if necessary.
UPDATE	(→)	Mark last block referenced as modified.
BUFFER	(n → addr)	Leave addr of a free buffer, assigned to block n; write previous contents to mass storage if UPDATEd.
SAVE-BUFFERS	(→)	Write all UPDATEd blocks to mass storage.
EMPTY-BUFFERS	(→)	Mark all block buffers as empty, without writing UPDATEd blocks to mass storage.

DEFINING WORDS

: xxx	(→)	Begin colon definition of xxx. "colon"
;	(→)	End colon definition. "semi-colon"
VARIABLE xxx	(→) xxx: (→ addr)	Create a two-byte variable named xxx ; returns address when executed.
CONSTANT xxx	(n →) xxx: (→ n)	Create a constant named xxx with value n; returns value when executed.
VOCABULARY xxx	(→)	Create a vocabulary named xxx ; becomes CONTEXT vocabulary when executed.
CREATE ... DOES>	does: (→ addr)	Used to create a new defining word, with execution-time routine in high-level FORTH. "does"

VOCABULARIES

CONTEXT	(→ addr)	System variable pointing to vocabulary where word names are searched for.
CURRENT	(→ addr)	System variable pointing to vocabulary where new definitions are put.
FORTH	(→)	Main vocabulary, contained in all other vocabularies. Execution of FORTH sets context vocabulary.
DEFINITIONS	(→)	Sets CURRENT vocabulary to CONTEXT.
' xxx	(→ addr)	Find address of xxx in dictionary; if used in definition, compile address. "tick"
FIND	(→ addr)	Leave compilation address of next word in input stream. If not found in CONTEXT or FORTH, leave 0.
FORGET xxx	(→)	Forget all definitions back to and including xxx , which must be in CURRENT or FORTH.

COMPILER

,	(n →)	Compile a number into the dictionary. "comma"
ALLOT	(n →)	Add two bytes to the parameter field of the most recently-defined word.
."	(→)	Print message (terminated by "). If used in definition, print when executed. "dot-quote"
IMMEDIATE	(→)	Mark last-defined word to be executed when encountered in a definition, rather than compiled.
LITERAL	(n →)	If compiling, save n in dictionary, to be returned to stack when definition is executed.
STATE	(→ addr)	System variable whose value is non-zero when compilation is occurring.
[	(→)	Stop compiling input text and begin executing. "left-bracket"
]	(→)	Stop executing input text and begin compiling. "right-bracket"
COMPILE	(→)	Compile the address of the next non-IMMEDIATE word into the dictionary.
[COMPILE]	(→)	Compile the following word, even if IMMEDIATE. "bracket-compile"

MISCELLANEOUS

(	(→)	Begin comment, terminated by) on same line or screen; space after (. "paren", "close-paren"
HERE	(→ addr)	Leave address of next available dictionary location.
PAD	(→ addr)	Leave address of a scratch area of at least 64 bytes.
>IN	(→ addr)	System variable containing character offset into input buffer; used, *e.g.*, by WORD. "to-in"
BLK	(→ addr)	System variable containing block number currently being interpreted, or 0 if from terminal. "b-l-k"
ABORT	(→)	Clear data and return stacks, set execution mode, return control to terminal.
QUIT	(→)	Like ABORT , except does not clear data stack or print any message.
79-STANDARD	(→)	Verify that system conforms to FORTH-79 Standard.

FORTH

Denna spalt är inledningen till tidningens i fortsättningen återkommande FORTH-spalt. Jag hoppas att vi i denna spalt (ev. avdelning) kan ventilera synpunkter och program i FORTH. Alla ni som är intresserade av språket är hjärtligt välkomna att skriva till klubben, både för att delge oss andra de program ni åstadkommer och för att överhuvudtaget "ventilera" eventuella synpunkter om detta fenomenala språk. Alla bidrag, grundläggande och avancerade, har chans att komma in.

Denna gång skall jag återge ett par olika synpunkter och intressanta saker från KOM på Q-Zentralen. De är små funderingar om FORTH, plus ett bidrag till programbytet inom FORTH. Håll till godo, FORTH-hackers!

(Text 26122) 82-10-17 20.51 Mats Knuts ABC-klubben
Ärende: FORTH screens via modem

Är det någon som har programvara för att sända respektive ta emot FORTH screens från ABC-klubbens Monitor?

(Text 26400) 82-10-18 11.45 Gunnar Tidner
Kommentar till: (Text 26122) av Mats Knuts ABC-klubben
Ärende: FORTH screens via modem

Screens till Forth på ABC80 distribueras inom ABC-klubben på kassett som textfiler SCREEN.TXT och SCREEN2.TXT som låg på ABC-kassett nr 4 reesp 5. Med programmet DOSCEEN på kassett nr 4 konverteras dessa textfiler till det diskformat som användes på ABC80. Jag har gjort ett motsvarande program EXSCREEN som konverterar diskscreens till textfiler. Jag föreställer mig att det är smidigast att sända screens på detta sätt som textfiler, då alla vanliga filöverföringsprogram användas. Sedan kan man i resp dator återskapa screens som gjorts på en annan ABC80 eller efter vissa modifieringar av DOSCREEN screens som gjorts på det konventionella sättet (4 block per screen med 16 * 64 rader ungefär som du gjort när gällde ABC800).

(Text 26749) 82-10-19 01.19 Överföring från FOA 3
Kommentar till: (Text 26400) av Gunnar Tidner
Ärende: FORTH screens via modem

Verkar intressant. Finns det någon möjlighet att få tillgång till programet?

(Text 27242) 82-10-19 23.18 Gunnar Tidner
Kommentar till: (Text 27182) av Mats Knuts ABC-klubben
Ärende: FORTH screens via modem

Javisst men det kommer annars på kassett (nr 7 tror jag).

Anm. Meningen var att programmet EXSCREEN skull komma på kassett nr 8, men vid slutredigeringen måste det utgå i brist på utrymme. Programet publiseras därför härintill i detta ABC-blad.

(Text 28464) 82-10-23 22.40 Michael Evans S-E-Banken
Kommentar till: (Text 30244) av Torgny Tholerus QZ
Ärende: Hur ser ett FORTH-program ut egentligen?

Forth arbetar med en stack som utnyttjas för parameteröverföring till och från Forthord. Skall man addera två tal a och b och skriva ut dreras summa skriver man bara:

a b + . (där a och b motsvaras av siffervärden, + är liksom . Forthord)

Om A B och C är variabler och man skriver B så lägges adressen till B på stacken. Skriver man E så tar Forth översta elementet på stacken och tolkar det som en adress och ersätter detta element med innehållet på den aktuella adressen. Skriver man ett utropstecken ! så lägger forth på den adress som ligger överst på stacken det talvärde som ligger näst överst (och stacken blir av med dessa två element). Forthordet SWAP låter de två översta elementen på stacken byta plats. Motsvarigheten till A:=B+C blir alltså i Forth: A B E C E + SWAP !

(Text 30335) 82-10-27 17.47 Torgny Tholerus QZ

Kommentar till: (Text 30860) av Gunnar Tidner
Ärende: Hur ser ett FORTH-program ut egentligen?

En av anledningarna till att FORTH program kan bli snabbare än program skrivna i andra mera traditionella högnivåspråk är ju att temporära resultat INTE behöver läggas i någon "variabel" dvs minnesposition utan kan användas direkt på stacken i vidare beräkningar. God FORTH-stil är alltså att undvika variabler där de inte absolut behövs. Detta underlättas om man skriver korta men kraftfulla operatorer och inte förväntar sig ha långa parameterlistor på stacken utan endast ett eller två värden. Adresser kan förstås manipuleras godtyckligt liksom i assembler. Detta gör att man kanske bör se FORTH som en interaktiv stackorienterad macroassembler med goda möjligheter att skriva strukturerad kod. Den viktigaste skillnaden mot till exempel ett LISP-system är att användaren får bygga alla datastrukturer själv (utom stacken) och alltså måste vara mera medveten om hur FORTH-systemet fungerar internt när han gör avancerade saker. Naturligtvis saknas också den teoretiskt tillfredsställande egenskapen hos LISP att data och program har en identisk och enkel representation. För en realtidstillämpning misstänker jag dock med risk för att bli offentligen halshuggen att man med FORTH kan få effektivare program än med motsvarande arbetsinsats i LISP. Framförallt kan användaren lättare styra utnyttjandet av minnesrymden (på gott och ont förstås !).

(Text 31200) 82-10-30 18.19 Michael Evans S-E-Banken
Kommentar till: (Text 31176) av Thomas Sjöland
Ärende: Hur ser ett FORTH-program ut egentligen?

Jag tror inte att avsaknaden av variabler i FORTH påverkar prestanda alltför mycket. Däremot tror jag att användning av en "kompilator" som gör om koden till "Indirect Threaded Code" (don't ask me what it means) ger bättre prestanda.

Trots att man på FORTHs lägsta nivå är medveten om adresser mm så finns det goda möjligheter att bygga en serie applikationsnivåer ovanför grundnivån. På så sätt behöver användaren/applikationsprogrammeraren inte var medvetna om dessa.

T ex så kan jag manipulera portarna från FORTH. Det första jag gör är att skriva funktioner som heter STARTA-KASSETT och STOPPA-KASSETT och sedan behöver jag aldrig bryr mig om portarna för det jobbet igen.

Liknande tekniker används i APL där man brukar inte vilja ge slutanvändarna rå APL som programmeringsspråk

(Text 31314) 82-11-01 12.54 Torgny Tholerus QZ
Kommentar till: (Text 31200) av Michael Evans S-E-Banken
Ärende: Hur ser ett FORTH-program ut egentligen?

Trådad kod innebär att ett FORTH-ord lagras i en symboltabell med ett headerfält och ett contentfält i princip. I headerfältet finns information bl.a. om vilken interpretator som skall exekvera just detta FORTHord. I contentfältet ligger data rakt upp och ner. Hur detta data tolkas beror på vilken interpretator som interpreterar FORTHordet ifråga. Om det är frågan om en : definition (kolondefinition) dvs. den normala interpretatorn så utgör contentfältet en lista av adresser till de FORTH-ord som sekvensiellt genomgås av detta ord. Är det frågan om maskinkod så utgör contentfältet själva programmet och pekaren till vilken interpretator som avses pekar rakt in i contentfältet. Detta avslutas med ett hopp tillbaka till FORTHinterpretatorn (:). Är det frågan om variabelfält med lite intelligens så kan contentfÄltet tolkas på annat sätt. I FORTH har man alla möjligheter i världen att göra egna interpretatorer för speciella syften och är inte beroende av syntaktiska och semantiska egenheter hos ett specifikt språk körande i ett specifikt operativsystem. Det ger naturligtvis också möjligheten att göra galna saker, något som dock fullständigt måste lastas på programmeraren i fråga. Trådad kod har använts även i andra sammanhang för t.ex. FORTRAN-kompilatorer och är en teknik som framförallt optierar minnesanvändningen (alltså inte tiden; direkt rak kod är alltid snabbare förutsatt att den inte håller på med en massa onödiga saker fÖr säkerhets skull). Sist några ord om säkerheten hos FORTH: Det finns inte mera säkerhet i FORTH än vad programmeraren själv bygger in, men liksom i assembler finns möjligheten att själv spärra ut farliga saker. Skillnaden mot att skriva i assembler direkt är framförallt den interaktiva miljön och mekanismerna fÖr stackhantering och symboltabellhantering samt sekundärminneshantering som i allmänhet finns i systemet. Lustigt nog tenderar FORTH-program att bli GOTO-fria och snyggt strukturerade. Goto existerar inte och jag har svårt att se behovet av det heller.

Postgirokonton:

Medlemsavgifter	**: 15 33 36 - 3**
Publikationer	**: 62 93 00 - 5**
Q-Zentralen	**: 43 51 74 - 8**

Preprocessor till ABC80

En intressant nyhet som kommit heter **StorBASIC**, och är en preprocessor till ABC80 och som ytterligare vidgar ABC80:s kapacitet. Preprocessorn fungerar så att man först gör iordning en textfil som sedan bearbetas av ett program som tillhandahåller en "översättning" till ett nytt program samt en hel del dokumentation i olika former. Detta torde vara värdefullt vid fortsatt bearbetning och utveckling av programmen.

Vad kan då denna nyhet?
Långa variabelnamn
Långa radetiketter
Långa radnummer
Automatisk packning av flera satser per rad
Automatisk omvandling till heltal
Obegränsat med kommentarer

Långa variabelnamn, upp till 30 tecken långa med få undantag av vilka tecken som helst kan användas och kombineras till urskiljbara "ord". Den använder radetiketter i stället för radnummer. Dessas kan också vara upp till 30 tecken långa och med samma begränsning av vilka tecken som får användas.
En finess är att man kan ha två typer av kommentarer, dels vanliga REM och dels efter !, och som ej är kvar efter RUN och under exekveringen. Bidrar till att man kan ha betydligt bättre dokumentation för att undvika att behöva gissa vad ens program gör efter någon tid på hyllan samtidigt som de inte tynger exekveringen.

Möjligheten finns att använda en annan packning av programmen än vad ABC80 gör normalt. Det gör att man kan få rum med 20-30 procent större program. Man måste å andra sidan skriva helt rätt. Med denna typ av packning medför alltså att signifikanta mellanslag måste sitta där de skall.

Vidare tolkar **StorBASIC** alla variabler som heltal utan att man behöver sätta dit procenttecknet efteråt. Vill man ändå ha variabeln som flyttal sätter man dit ett uttropstecken efter variabeln eller siffran. På samma sätt finns möjligheten att slippa hålla reda på sol-tecknet när man arbetar med strängar.

BASIC kan köras på en ABC80 i grundutförande med 16 k ords minne. Då får dock ingen printerrutin plats. För att få plats tas 3 st DOS-buffertar bort av programmet, något som inte sker om man har extra minne. Det får plats över 200 radetiketter om vardera 5 tecken. Detsamma gäller för antalet variabler. Och ju kortare namnen är desto fler går det att ha. I en ABC80 med 8 k extra minne får det plats över 600 etiketter om vardera 10 tecken. Och har man fullt utbyggt minne (32 k extra) ryms drygt 1200 etiketter om 10 tecken och teoretiskt sett lika många variabler.

ABC80 kan själv hantera maximalt 319 olika variabler av varje typ, d v s heltal, flyttal, sträng, heltalsmatris, flyttalsmatris och strängmatris. Antalet maximala variabler beror på om man använder alla olika typerna.

Vidare upplysningar om **StorBASIC ver 1.0** som kostar 595:- kan erhållas av **Peter Stahl**, Särlavägen 21, 175 40 Järfälla, 0758-119 90.

Radiostörningar från ABC80

Jag har sedan den 9 november 1980 varit ägare till en ABC80. Fram till härom veckan har allt varit frid och fröjd.

Men då hände det! På gatan utanför fanns en gul skåpvagn med en "tv-antenn" på taket. I färskt minne hade jag lokala avisans notis om någon som piratmässigt tagit över sändningarna via P3-sändaren i grannstaden Trelleborg.

Vad kan televerkets pejlbuss vilja här utanför, tänkte jag och vände mej åter mot min nystartade ABC80.

-Ring, ring, sa dörrklockan.

-Va, tror han att det är jag som är radiopiraten! Det var det värsta. Jag som inte i hela mitt liv sänt ut en enda illegal ton!

-Goddag, det är från televerket. Jag har spårat en störning till det här området.

-Jasså och jaha, det enda jag har ingång är kaffebryggaren och ABC80!

-Ja, kan det inte vara den då, ABC80 alltså?

Televerkaren och jag kom överens om att om han ute i sin buss kan se att störningen försvinner när jag slår av ABC80, så skall ha blinka med lyset. Han gick ut, jag slog av, han blinkade med lyset.

Saken var klar. Jag **var** en radiopirat! Det visade sig att jag i månader hade stört kommunens radionät och att man jagat mej i halva sydsverige och även i angränsande delar av Danmark! Men nu hade man mej fast!

Så här är min geografiska belägenhet: på andra sidan Ringvägen härstädes, avstånd ca 400 meter, står två av stadens trenne vattentorn. En av dessa två gör tjänst som support för antenner, bl a för den service som jag stört ut till oanvändbarhet.

Jag fick nu ålägga mig total och fullständig avhållsamhet vad beträffar datoranvändande, då även i radioamatörsammanhang.

Kontakt togs med Luxor i Motala, där Hans Karlsson tog del av mina problem. Efter många turer och mycken hjälp, kan jag nu åter använda min dator.

Vad jag gjort är ungefär följande:

1) Skärmning av tangentbordet med aluminiumfolie på insidan.

2) Bättre jordning av skärmstrumpa på kabel mellan tangentbord och dataskärm.

3) Bättre jordning av kylflänsar baktill på tangentbordet..

4) Utbyte av flatkabel från buss till printer och FD 2 mot en rund skärmad kabel.

5) Flyttning av printerinterfacet till ledig plats inuti FD 2.

6) Upplindning av nätsladd på ferritring.

7) Avkoppling med kondensatorer på flera känsliga ställen.

8) Bra jordning av hela systemet.

Dessa åtgärder bidrog var för sig till att radiostörningarna avtog markant. Min i mitt fall räckte det inte. Varje gång jag slog igång datorn så startade basstationens sändare i vattentornet och la ut en bärvåg, fylld med sus och brus.

Nu är det så att i datorn finns en styrkristall med frekvensen 11980,8 kHz. Den kristallens sjunde överton blir 83,86 MHz, just den frekvens som är basstatios infrekvens. Jag fick i stället löda in en 12 MHz kristall med en seriekondensator, och har således flyttat utstrålningen från 83,86 MHz till en "ofarligare" frekvens. Nu går ju internklockan i datorn ännu mer fel, men jag hade ju att välja mellan pest och kolera...

Nämnas bör att ABC80 är godkänd enligt tyska och amerikanska normer vad beträffar störande utstrålning. Dessa normer förefaller mej vara en aning liberala. Men ja kan nu använda min dator med gott samvete, väl vetande att det inte finns några som helst svenska (och strängare) normer som sätter stopp för sortsatt bruk. Drar jag dessutom ur busskontakten på baksidan så är jag så gott som störningsfri.

Hoppas ingen annan råkat ut för samma problem.

John G Birkland
<238>

UPPMANING !

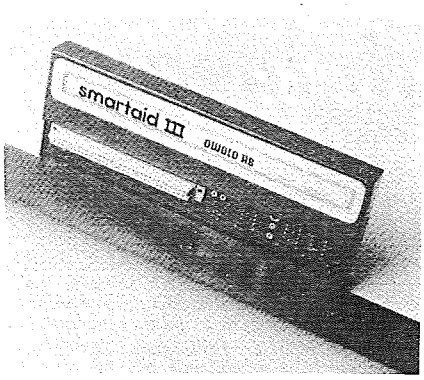
För att visa mångsidigheten hos ABC80 planerar vi en artikelserie om olika sätt att använda den.

Vi vill därför uppmana Dig att skicka in en kort rapport och berätta hur och till vad Du använder Din ABC80

Adressen är

ABC-klubben
Vidängsvägen 1
161 33 Bromma

	Postgirokonton:
Medlemsavgifter	**: 15 33 36 - 3**
Publikationer	**: 62 93 00 - 5**
Q-Zentralen	**: 43 51 74 - 8**



Användarrapport SMARTAID III
Vid min test av första SMARTAID från OWOCO AB i ABC-bladet 1981 nr 3 sidan 8 skrev jag att användningen av SMARTAID var vanebildande. Efter några veckors användning av SMARTAID III står det klart att Owoco har fått fram den verkligt tunga vanebildaren!

På ABC-kassett 8 har klubbmedlemmarna fått tillgång till "Hjälpare" som har en hel del av SMARTAIDs funktioner. Man har t o m samma CTRL-tecken och benämningar på flertalet av funktionerna. Detta kanske kan uppfattas som plagiat, men jag tror att det är en stor fördel för dem som blivit vanebildade och vill komma upp sig till en riktig SMARTAID.

Den direkta fördelen med SMARTAID III framför andra BASIC-editorer är att man får ett kretskort förpackat i en stadig plåtlåda monterat på en fästplatta som skruvas fast under ABC80s tangentbord. Anslutning sker till busskontakten, och motsvarande uttag sitter i samma höjd på baksidan av SMARTAID III.

Detta innebär att SMARTAIDs funktioner ligger lagrade i ett PROM på adresser som inte inkräktar på det ordinarie arbetsminnet för BASIC-program. SMARTAID III är även försedd med en rutin som ger automatisk uppstart vid tillslag av nätspänning, tryck på resetknappen eller återgång till BASIC från OS med o BAS. Utskrift sker då av aktuell systeminformation med aktuella värden på programlängd, minnesåtgång för variabler etc. Samma information får man med kommandot SYS, som framgår av fotografiet härintill.

Dessutom får man "på köpet" en drivrutin för skrivare, antingen seriellt över V24-kontakten, eller parallellt över busskontakten. Man har här samma uppställning av parametrar som blivit standard genom Luxors printer-prom.

SMARTAID III är anpassad för körning på alla tre versionerna av ABC80 med olika checksummor. Den klarar även 80 kolumners bildskärm.

Allt verkar genomtänkt med avseende på att få så enkelt handhavande som möjligt. Vill man göra LIST på ett laddat BASIC-program behöver man bara trycka "tyskt y" (ASCII 94 alternativt ASCII 126) som ligger omedelbart till vänster om RETURN-tangenten, åtföljt av RETURN. SMARTAID ser då till att inga rader rullas förbi och försvinner ovanför kanten på bildskärmen. Mellanslag eller högerpil fortsätter listningen framåt, medan tangenten för vänsterpil backar listningen och scrollar texten nedåt på skärmen. Vill man lista på fil skriver man t ex (ASCII 94/126)-PR: med bindestreck (minustecken) efter ASCII(94/126), för att förhindra oavsiktlig lagring på diskett. Man kan även göra
LIST 40-90 PR:
för att få ut del av program.

Det är svårt att säga vilken funktion jag värdesätter mest. Vid försök att göra ett överskådligt BASIC-program brukar jag låta olika programdelar börja med radnummer på jämna 1000-tal. Efter tillräckligt många ändringar måste man göra RENumber, och då försvinner alla goda förutsatser. Med SMARTAID III kan man t ex göra
REN 2000,3 FROM 2000-2999
och få enbart omnumrering av raderna mellan 2000 och 3000 i steg om 3.

Här följer ett försök till sammanfattning av funktioner som skiljer sig från eller saknas i andra BASIC-editorer.

CTRL L Rensar bildskärmen till höger och nedanför markören. Är det redan tomt sker radering av hela skärmen.

CTRL LL Rensar alltså hela skärmen.

CTRL P flyttar alla tecken till höger om markören ett steg åt höger. Börjar man sedan mata in text fortsätter gammal text till höger om markören att flyttas åt höger ända tills man trycker någon tangent för ren markörförflyttning.

CTRL D tar bort tecknet omedelbart till höger om markören och efterföljande text flyttas upp åt vänster.

CTRL C stannar programkörning temporärt.

CTRL CC eller CTRL C+RETURN avbryter programkörning helt.

Fortsättning kan göras med RESUME. Anges radnummer, RESUME n, fortsätter exekveringen från rad nummer n, i bägge fallen med bibehållna variabelvärden. Enbart RESUME kan ge ERR 29 om man råkat stoppa i en subrutin. Då får man göra RESUME med angvande av ett radnummer.

Efter CTRL C kan man trycka
CTRL S för att få stegvis körning eller
CTRL T för att få stegvis körning med TRACE-funktion.

Vill man fortsätta efter stegvis körning trycker man mellanslag.

CTRL + SHIFT + O (samtidig nedtryckning av CTRL, SHIFT och bokstaven O) ger utskrift av bildskärmens innehåll på PR:

OLD tar fram föregående program efter NEW, SCR eller RESET. Här sker en fullständig syntaxkontroll, inte enbart sökning efter vagnreturer som man gör i "HJÄLPARE". Har RESET-knappen varit intryckt för länge, kan inte heller SMARTAID III återhämta ett program.

FIND A skriver ut alla rader (hela textraden) som innehåller variabeln A

FIND 100 skriver ut alla rader med hoppinstruktion till rad 100.

FIND "text" skriver ut alla rader där godtycklig textsträng finns. Man får alltså utskrivet hela raden, inte enbart aktuellt radnummer.

PEEK är en funktion för listning av internminnet. PEEK 4096- ger minnesutskrift decimalt och hexadecimalt från adress 4096 och framåt. Piltangenterna har samma funktion här som vid LIST, så att man kan "scrolla" framåt och bakåt. Vill man veta hexadecimala värdet av t ex 29876 skriver man PEEK 29876 och tittar vad som skrivs i kolumnen för hexadressen.

CHANGE A TO B byter alla variabler A till variabeln B Man kan från början skriva alla variabler utan %-tecken i AUTOMODE. Sedan gör man CHANGE A TO A% o s v för varje variabel.

LIB eller LIB n visar biblioteket över innehållet på vanliga 5.25"-disketter

DIR eller DIR n visar biblioteket på 8"-disketter. n betecknar nummer på viss diskettdrive.

PRINT CALL(16393) visar tiden i tim. min. och sekunder.

FOR ... NEXT kan anges som kommando, hela satsen (utan radnummer) skrives i samma rad (max 120 tecken).

START n ger samma funktion som RUN från rad nummer n.

De andra funktionerna liknar motsvarande funktioner från andra BASIC-editorer och jag vet inte någon jag saknar i SMARTAID III.

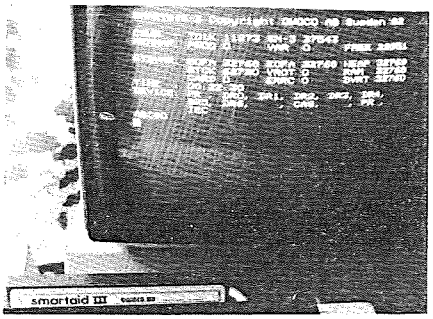
Om man redan har en drivrutin för skrivare i systemet väljs denna och SMARTAID IIIs skrivarbeordring ligger inaktiv. Vill man hellre ha skrivarrutinen i SMARTAID III, kan man göra ;CALL(16387).

Nu har jag kört SMARTAID III i över en månad och ännu har jag inte råkat ut för några otrevliga överraskningar. Allt har fungerat enligt den utförliga bruksanvisningen

I mitt arbete har jag nu börjat köra ABC800, men jag föredrar faktiskt att göra programutveckling på ABC80 med SMARTAID III.

Ytterligare informationer kan fås från OWOCO AB tel. 08/774 02 90.

Göran Sundqvist <1255>
tel. 0756/30310.



Brev

Hej !

Skickar över en beskrivning på ett program som får ABC80 att läsa kassetter som inspelats med ABC800. Det gör att ABC800-ägaren i övre Norrland kan skicka en kassett till ABC80-polaren i Skåne i stället för att skicka program över telefonnätet.

Eftersom programmet är utförligt dokumenterat kan det ha ett visst intresse även för de som inte omedelbart behöver använda det.
Programmet är skrivet för att vara lätt att läsa och förstå och inte för att vara så smart eller litet som möjligt.
Gack du och gör sammanlunda.
Om det finns intresse så är det fritt fram för publicering i ABC-bladet.

..ilsningar

Per Ahlin >1911<
Åsögatan 56 6tr
116 24 Stockholm

Hur man får ABC80 att läsa kassettband inspelade med ABC800.

ABC800 kan läsa kassettband med data eller program lagrade i BAS-format som inspelats med ABC80. Det omvända går emellertid inte eftersom ABC800 skickar signaler till kassettbandspelaren med en högre hastighet (2400 baud) än ABC80.

Det är emellertid möjligt att lära ABC80 att läsa kassettband inspelade med ABC800. Förutsättningen är förståss att det är fråga om data eller program (i BAS-format) som är kompatibla med ABC80's BASIC.

Snarare än att bara presentera en programlista, något som visserligen kan vara användbart, så skall jag också beskriva programmet någorlunda detaljerat, med förhoppningen att det även skall bli lärorikt. För den som vill följa resonemangen kan det vara bra att ha tillgång till disassembleringen av ABC80's programvara (ABC-klubbens rapport 1) samt någon bok som beskriver Z80's maskinkoder. Adresserna betecknas i första hand som i rapport 1 dvs adress 128:15 = 128 * 256 + 15 = 32783.

När ABC80 läser från kassettbandspelare, liksom från tangentbordet så sker det genom en s.k. avbrottsrutin. När ABC blir avbruten ..d det den råkar ha för sig, så tittar den efter vilka tal som finns i CPUn's I-register (här hittar ABC80 talet 0). Dessutom anländet ytterligare ett tal vid själva avbrottet. Dessa båda tal bildar tillsammans en adress, 0:52 för tangentbordet och 0:54 för kassettspelaren. På dessa adresser hittar ABC80 följande:

Adress	Innehåll	Betydelse
0 0:52	30	På adress 3:30 finns instruktioner för vad
0 0:53	3	som skall göras vid avbrott från tangentbordet.
0 0:54	148	På adress 5:148 finns instruktioner för vad
0 0:55	5	som skall göras vid avbrott från kassettbandspelaren.

Om vi vill att ABC80 skall göra något annat än den brukar när den blir avbruten så kan det ske genom att man ändrar I-registrets innehåll till exempelvis 128 (i stället för 0). När ett avbrott sker tittar ABC80 då i stället efter på adress 128:52 för tangentbordet och 128:54 för kassettbandspelaren.

Fler rutiner än så här behövs inte om det bara är fråga om att läsa data. Omställningen av läshastighet får då ske i BASIC-programmet med CALL-satser till de båda rutinerna ovan. Om man emellertid också vill använda kommandot LOAD, måste man på något sätt bygga in ändringen av läshastighet eftersom ju ABC800 skriver första blocket med låg hastighet och därefter resten med hög hastighet. Vi blir tvugna att definiera en ny enhet. Med enhet menas här yttre enhet som CAS PR V24 DR0 osv.

Det första som då skall göras är att sätta dit en ny första länk i den kedja av enheter som redan finns. När ABC80 vill snacka med en enhet tittar den först efter på följande adresser med exempelvis detta innehåll:

KAS:

Adress	Innehåll	Betydelse
254:10	147	På adress 1:147 finns information om den
254:11	1	första enheten i kedjan.

Om vi vill stoppa in en ny enhet ändrar vi till:

Adress	Innehåll	Betydelse
254:10	0	På adress 128:0 finns information om den
254:11	128	första enheten i kedjan.

Sedan måste vi förståss se till att det finns något begripligt på adress 128:0

Adress	Innehåll	Betydelse
128:0	147	Information om nästa enhet i kedjan finns på
0 :1	1	adress 1:147
0 :2	75	K
0 :3	65	A Den här enheten heter KAS (ASCII-
0 :4	83	S tecken)
0 :5	7	På adress 128:7
0 :6	128	får du veta mera om du vill.

Och det mera som ABC80 måste få veta om enheten har formen av en s.k. hopptabell. Hopptabellen för CAS finns på adress 3:164 - 3:190. Vi gör därför en hopptabell åt KAS.

Adress	Innehåll	Betydelse
128:7	195,40,128	Om du skall göra OPEN, hoppa till 128:40.
128:10	195,191,3	Samma som för CAS.
128:13	195,40,4	"
128:16	195,160,9	"
128:19	195,18,9	"
128:22	195,46,128	Om du skall läsa en record, hoppa till 128:46.
128:25	195,14,4	Samma som för CAS
128:28	195,14,7	"
128:31	195,39,7	"

På dessa adresser får vi alltså se till att det finns hänvisningar till vad vi vill att ABC80 skall göra vid avbrott. Låt oss exempelvis ha:

Adress	Innehåll	Betydelse
128:52	30	Här har vi samma hänvisning som förut.
128:53	3	Tangentbordets funktion vill vi inte ändra på.
128:54	56	Hänvisning till en ny adress 128:56, där vi
128:55	128	måste ha en ny rutin för kassettbandspelaren.

Om vi vill ändra på vad ABC80 skall göra vid avbrott från kassettbandspelaren, så tittar vi först efter vad som sker i vanliga fall med början på adress 5:148. Vi hittar snart att en konstant (nämligen 120) laddas in i ett register och senare används ett par gånger i en vänteloop. Vi vill alltså ändra på denna konstant för att läsa med en annan hastighet och kopierar rutinen på 5:148 fram till konstanten, lägger in den nya konstanten och hoppar till den vanliga rutinen efter konstanten:

Adress	Innehåll	Betydelse
128:56	251	Samma som 5:148 - 5:158
0 :57	245	
0 :58	197	
0 :59	219	
0 :60	58	
0 :61	230	
0 :62	191	
0 :63	211	
0 :64	58	
0 :65	79	
0 :66	62	
0 :67	25	Ny konstant
0 :68	195	Hopp till adress
0 :69	160	5:160
128:70	5	

Vi behöver också två subrutiner för att ställa om läshastigheten, dvs för att ändra innehållet i I-registret:

Adress	Innehåll	Betydelse
128:34	62	Lägg i A-registret
128:35	128	konstanten 128
128:36	237	Kopiera A-registret
128:37	71	i I-registret
128:38	201	Return
128:71	175	Lägg 0 i A-registret
128:72	237	Kopiera A-registret
128:73	71	i I-registret
127:74	201	Return

Utom för OPEN och "läsa en record" så skall ABC80 alltså göra samma sak med KAS som med CAS. Vid OPEN måste vi emellertid försäkra oss om att läsning sker med den låga hastigheten och vid "läs record" att läsning sker med den höga hastigheten. Vi pular därför in:

Adress	Innehåll	Betydelse
128:40	205	Subrutinhopp till
0 :41	71	adress 128:71 dvs
0 :42	128	nollställ I-registret (låg läshastighet)
0 :43	195	Hoppa till OPEN-rutinen för CAS
0 :44	145	på adress 4:145
0 :45	4	
128:46	205	Subrutinhopp till
0 :47	34	adress 128:34 dvs
0 :48	128	lägg 128 i I-registret (hög läshastighet)
0 :49	195	Hoppa till "läs record"-rutinen för CAS
0 :50	9	på adress 5:9
128:51	5	

Slutligen får vi inte glömma att ställa om BOFA (på adress 254:-28-29) som talar om för ABC80 var BASIC-program skall börja skrivas in:

Adress	Innehåll	Betydelse
254:28	75	Från och med adress 128:75
254:29	128	är det ledigt.

Här följer nu ett BASIC-program som lär ABC80 att läsa ABC800-kassetter enligt beskrivningen ovan. Programmet är gjort för ABC80 med checksumma 11273 men är inte som ovan bara gjort för maskiner med 32K minne. Av utrymmesskäl är det lämpligt att köra det här programmet först innan du kör eventuellt printerprogram, exempelvis ABCV24. Om du vill använda CAS efter att ha använt KAS så är det lämpligt att försäkra sig om att den låga läshastigheten är inställd. Det görs genom satsen eller kommandot:

Z%=CALL(Y%) där Y%=B%+71% i programmet.

Om du får problem, så pröva med att ändra läshastighetskonstanten på adress B%+67% i intervallet 22-30.

Lycka till !

Per Ahlin >1911<

```
270 A%=PEEK(65053%)+SGN(PEEK(65052))
280 B%=A%*256%
290 POKE 65052,75,A% : REM NY BOFA
330 C%=PEEK(65034) : D%=PEEK(65035)
370 POKE 65034,0,A%
450 POKE B%,C%,D%,75,65,83,7,A%
490 POKE B%+7%,195,40,A%,195,191,3,195,40,4,195,160,9,195
500 POKE B%+20%,18,9,195,46,A%,195,14,4,195,39,7,195,39,7
580 POKE B%+34%,62,A%,237,71,201,0
630 POKE B%+40%,205,71,A%,195,145,4
670 POKE B%+46%,205,34,A%,195,9,5
720 POKE B%+52%,30,3,56,A%
770 POKE B%+56%,251,245,197,219,58,230,191,211,58,79,62
820 POKE B%+67%,25
860 POKE B%+68%,195,160,5
910 POKE B%+71%,175,237,71,201
960 CHAIN ""
970 END
```

För checksumma 9913 och 10042 kolla sidan 28

```
100 REM ******************************************************
110 REM *                                                    *
120 REM * KAS800.BAS  VER 1983-01-10                         *
130 REM *                                                    *
140 REM * HOPTOTAT AV PER AHLIN  >1911<                      *
150 REM *             ÅSÖGATAN 56,  6TR                      *
160 REM *             116 24  STOCKHOLM                      *
170 REM *                                                    *
180 REM ******************************************************
190 REM
200 REM GÖR DET MÖJLIGT FÖR ABC80 ATT REM LÄSA KASSETTER
210 REM INSPELADE MED ABC800
230 REM ------------------------------------------------
240 REM
250 REM BOFA
260 REM
270 A%=PEEK(65053%)+SGN(PEEK(65052))
280 B%=A%*256%
290 POKE 65052,75,A% : REM NY BOFA
300 REM
310 REM ENHETSROT GAMMAL
320 REM
330 C%=PEEK(65034) : D%=PEEK(65035)
340 REM
350 REM ENHETSROT NY
360 REM
370 POKE 65034,0,A%
380 REM ------------------------------------------------
390 REM
400 REM RUTINERNA LAGRAS PÅ ADRESSERNA  B% TILL B%+74%
420 REM
430 REM ENHETSLÄNK FÖR KAS
440 REM
450 POKE B%,C%,D%,75,65,83,7,A%
460 REM
470 REM HOPPTABELL FÖR KAS
480 REM
490 POKE B%+7%,195,40,A%,195,191,3,195,40,4,195,160,9,195
500 POKE B%+20%,18,9,195,46,A%,195,14,4,195,39,7,195,39,7
540 REM ------------------------------------------------
550 REM
560 REM HÖG LÄSHASTIGHET
570 REM
580 POKE B%+34%,62,A%,237,71,201,0
590 REM ------------------------------------------------
600 REM
610 REM OPEN KAS
620 REM
630 POKE B%+40%,205,71,A%,195,145,4
640 REM
650 REM LÄS RECORD KAS
660 REM
670 POKE B%+46%,205,34,A%,195,9,5
680 REM ------------------------------------------------
690 REM
700 REM ADRESSER VID AVBROTT
710 REM
720 POKE B%+52%,30,3,56,A%
730 REM ------------------------------------------------
740 REM
750 REM AVBROTTSRUTIN KAS
760 REM
770 POKE B%+56%,251,245,197,219,58,230,191,211,58,79,62
790 REM
800 REM LÄSHASTIGHETSKONSTANT
810 REM
820 POKE B%+67%,25
830 REM
840 REM HOPP TILL VANLIG RUTIN
850 REM
860 POKE B%+68%,195,160,5
870 REM ------------------------------------------------
880 REM
890 REM LÅG LÄSHASTIGHET
900 REM
910 POKE B%+71%,175,237,71,201
920 REM ------------------------------------------------
930 REM
940 REM RENSNING
950 REM
960 CHAIN ""
970 END
```

FORTH PÅ ABC 80

Vad är Forth egentligen? Ett omständligt sätt att skriva program, som förmodligen skulle fungera lika bra i Basic?

-Inte alls.

Forth är lika lätt att använda och lära sig som Basic, och betydligt roligare. Förutsättningen för att lyckas är att inte försöka skriva Basicprogram i Forthkod.

Ladda in Forth, för nu ska vi ta reda på hur det fungerar. Forth är ett "indirekt trådat" språk. Det innebär att det kompilerade programmet består av pekare till pekare, som pekar på exekverbar maskinkod. Forth ... alltså alla förutsättningar att gå forth ... vitsen!). När du skriver något tas det om hand av den yttre interpretatorn. Den kompilerar orden (instruktionerna kallas "ord" eller "verb" i Forth) till pekare till maskinkod, och anropar sedan den inre interpretatorn, som exekverar koden. Denna exekvering går i princip ut på att ladda registerpar HL med pekarens värde, och sedan ...ppa dit.

Det här låter ju bra, men vad ska man egentligen skriva för att få någonting gjort? Viss hjälp kan man få av VLIST, som visar alla tillgängliga ord. Utskriften kan stoppas med Return, och sedan svarar Forth med "ok", precis som Basic svarar med "ABC80". Kom ihåg att Forth ser skillnad på stora och små bokstäver, och att mellanslag används som separator. Skriver du VLi sT, börjar Forth söka efter ordet VLi i ordlistan. Eftersom det inte finns, skrivs ett felmeddelande ut:

```
VLi ?
```

Har du tillgång till skrivare, kan du få listan utskriven även på denna:

```
PR-ON VLIST CR PR-OFF
```

PR-ON aktiverar skrivaren (gör ett call till printerinterfacets OPEN-rutin, och sätter en flagga som markerar att utskriften ska ... även till skrivaren. VLIST skriver ut ...an. CR skriver ut en radframmatning (det är nödvändigt om inte sista raden ska gå förlorad, eftersom PR-OFF inte skriver ut bufferten). PR-OFF, slutligen, släcker printerflaggan och anropar skrivarens CLOSE-rutin.

Bland de första ord som skrivs ut finns ...IT. Det är en s.k. "screen-editor". "screen" syftar snarare indirekt än direkt på datorns skärm.

En screen är det format i vilket Forth sparar data och program på skivan eller kassetten (screen syftar nog i sin tur på dataskärmen). Screenen består (på ABC80) av 768 bytes, dvs exakt 3 block. Standarden är egentligen 1024 bytes, dvs 4 block, men eftersom ABC80 i grundutförande endast har 40 kolumners skärm, har 3 block valts. Detta är snarare en fördel än en nackdel. Det kan vara nog så svårt att använda 64 positioner på en rad, om programmet ska bli lättläst. Dessa screens betecknas med löpande nummer från 1 och uppåt. För att editera en screen skriver man den aktuella screenens nummer, och sedan VEDIT:

```
1 VEDIT
```

När ett tal påträffas, kompileras automatiskt ordet LIT. Efter LIT läggs talet i den kompilerade koden. När koden sedan exekveras, anropas LIT, som lägger talet på stacken.

Innan vi börjar programmera, ska vi gå igenom lite grundläggande programmeringsmetodik.

Ett Forthprogram bygger på att man definierar ett ord bestående av en rad andra ord. Man skapar alltså macroord av de ord som redan finns i ordlistan.

En detalj som är ganska unik, är att man använder en stack, och att beräkningar utförs enligt en metod som kallas "omvänd polsk notation". Detta är ett sätt att göra Forth-interpretatorn enklare (och snabbare). ABC80s Basictolk använder också omvänd polsk notation internt.

När man skriver program i Forth, bör man försöka att inte göra som i Basic, du vet, med en massa instruktioner på samma rad, hopp hit och dit, och en allmänn gröt.

Det mest iögonfallande för en inbiten Basicfantast är att man använder programtextens vänstermarginal för att avgränsa loopar och villkorssatser.

Med det vi nyss lärt oss ska vi skriva vårt första ord, TJUT.

TJUT ska sätta fart på ljudgeneratorn. Vi vet att ljudgeneratorns portaddress är 6, och att "tjut" har värdet 7.

```
: TJUT ( ---)
   7 ( Ljudet)
   6 ( Porten)
   P! ( Data Port ---)
      ( Skriv på porten) ;
```

Kolonet inleder ett nytt ord. Efter kolonet, skilt med ett mellanslag, ska ordets namn stå. 7 och 6 är de värden ordet P! behöver. P! skriver på en port, och motsvarar alltså OUT i Basic. Kommentarer skrivs som parenteser. Observera, kommentarerna efter TJUT och P!. De anger vilka värden som ska passas mellan de olika orden. — markerar ordets plats.

Skriv TJUT, så börjar ljudgeneratorn tjuta. Eftersom Forth använder Basic-tolkens in- och utmatningsrutiner, stängs ljudet av när du trycker ner någon tangent. Det kan vara bra att veta.

På samma sätt kan vi definiera BELL.

```
: BELL ( — )
   131 6 P! ;
```

Man kan också skriva

```
: BELL ( — )
   7 EMIT ;
```

EMIT skriver ut ett tecken, och motsvarar alltså PRINT CHR$( ) i Basic.

Anta nu att vi vill ändra TJUT till att avge ljud nr. 3. Vi skriver

```
: TJUT ( — )
  3 6 P! ;
```

Då skriver Forth hastigt och lustigt ut ett felmeddelande. Vi måste tydligen ta bort det gamla TJUT innan det det nya kan skrivas in. Forth har trots det lagt in det nya TJUT sist i ordlistan. Men det gamla ligger kvar och tar upp minne.

Skriv FORGET TJUT, så tas det först påträffade TJUT bort, dvs det sist definierade.

Skriv FORGET TJUT igen, så tas även det gamla bort, men också BELL, eftersom det ligger efter TJUT i ordlistan. Kompilerade ord kan inte återlistas eller sparas, så det är ganska omständigt att skriva program direktmot kompilatorn. Det är därför man använder en editor som VEDIT. Då kan man skriva och editeraprogrammen i lugn och ro, medan kompileringen sker i en särskild fas.

En screen kan liknas vid en sida i ett anteckningsblock. När man skrivit full en sida, sätter man en markering i det nedre högra hörnet, som markerar att "fortsättning följer på nästa sida". Denna markering skrivs i Forth som —>. Sätt i en tom skiva i drive 0, och välj den för editering.

```
1 CLEAR 1 VEDIT
```

CLEAR suddar sida 1, och VEDIT skriver ut den på skärmen, med markören på position 0 på rad 0. Rad 0 bör bestå av en kommentar, en innehållsdeklaration för screenen.

Så här skulle en screen med orden TJUT och BELL se ut:

```
 0 ( TJUT   BELL   BELL2                 )
 1
 2 FORTH DEFINITIONS DECIMAL
 3
 4
 5 : TJUT   ( --- )
 6          ( OUTPUT SOUND NO. 3 )
 7    3 6 P! ;
 8
 9
10 : BELL   ( --- )
11          ( OUTPUT 'BELL' )
12    131 6 P! ;
13
14
15
16 : BELL2  ( --- )
17         ( OUTPUT BELLCHAR )
18    7 EMIT ;
19
20 ;S
```

Du flyttar markören med Ü, RETURN, -> och <-, precis som i TV-editorn, och går ur editorn med CTRL-E. Ordet ;S på rad 20 gör att kompileringen avstannar där. Du kan alltså skriva vad som helst efter det ordet. Skriv nu 1 LOAD, så laddas de nya orden. Du kan få automatisk start av ett ord egnom att skriva ordets namn efter dess definition.

Eftersom det bra ryms 98 sidor på en SD-skiva, och 205 på en DD-skiva, kan det vara lockande att snåla på utrymmet genom att skriva kompakta program. Gör inte det! Resultatet blir bara svårlästa program som är omöjliga att ändra i. Lämna mycket "luft" på sidorna. Speciellt när det gäller loopar, IF-satser och andra programstrukturer är det viktigt att man lämnar mycket "tomt" utrymme.

Appropå IF kan vi titta på hur det hanteras i Forth.

Ett exempel:

```
: IF-TEST  ( N --- )
   5 =
   IF
     ." FEM"
   ELSE
     ." EJ FEM"
   ENDIF
   CR  ( SKRIV UT EN RADFRAMMATNING ) ;
```

IF-TEST kräver alltså ett värde på stacken. Ordet = och 5 kollar om talet är 5 (och förstör samtidigt talet som testas), och lägger en 1:a på stacken om svaret är "ja", annars en nolla. IF utför i sin tur det som står efter IF om "ja" ligger på stacken, annars det som står efter ELSE. Om inte ELSE finns, sker hoppet direkt till ENDIF (istället för ENDIF kan ordet THEN användas). ." skriver ut den text som avslutas med citationstecken, utan att göra radframmatning. Observera att mellanslaget mellan ." och texten inte skrivs ut. Det måste dock finnas där, eftersom kompilatorn annars börjar leta efter ordet ."FEM", som ju inte finns.

På liknande sätt som IF fungerar BEGIN ... UNTIL, BEGIN ... AGAIN och BEGIN ... WHILE ... REPEAT. BEGIN markerar strukturens startpositon. UNTIL hoppar till BEGIN om "nej" ligger på stacken, annars fortsätter exekveringen förbi UNTIL. AGAIN gör ett ovillkorligt hopp till ordet efter BEGIN. WHILE ... REPEAT är lite mer komplicerad. Om WHILE får ett "nej" som argument, fortsätter exekveringen efter REPEAT, annars "rakt ner" till REPEAT, som gör ett hopp till ordet efter BEGIN.

Eftersom ett Forthprogram inte kan stoppas med CTRL/C, kan en BEGIN ... AGAIN-loop bara stoppas utanför programmet med reset. I programmet finns det två sätt att avbryta loopen. Det ena är att använda ordet QUIT, som motsvarar END eller STOP i Basic.

Ett sådant stopp kan ordnas på följande sätt:

```
: SHOW-QUIT ( --- )
   BEGIN
     ?TERMINAL
     IF
       QUIT
     ENDIF
   AGAIN ;
```

Det enda nya ordet är ?TERMINAL, som lämnar "ja" om en tangent tryckts ner, annars "nej".

En vanlig FOR-loop heter i Forth DO. Loopen avslutas med ordet LOOP eller +LOOP, och utförs så länge som loopindexet är mindre än gränsvärdet. Loopen kommer alltså att utföras en gång mindre än motsvarande Basicloop med samma värden.

Ett exempel:

```
: LOOPA ( --- )
   10 0 DO
      I .
   LOOP ;

1 FOR I%=0% TO 9%
2   ; I%;
3 NEXT I%
```

Dessa två program är likvärdiga. Eftersom Forth arbetar med heltal, används heltalsvariabler i Basicprogrammet. Observera att Forth-loopen har gränsvärdet 10, medan Basic-loopen har värdet 9. I båda fallen kommer talen 0-9 att skrivas ut på skärmen.

Nya ord:

I lägger loopens index på stacken. . (punkt) skriver ut värdet som finns på stacken.

Om man använder flera loopar i varandra, kan de nås med J. I' och J' tar fram respektive loops gränsvärde. Om man vill avsluta en loop fast den inte har snurrat färdigt, använder man ordet LEAVE. Det sätter loopens gränsvärde lika med dess index. Detta görs bara på den innersta loopen.

För att göra oss mer hemmastadda med loopar och villkorssatser, tar vi ett exempel till.

```
: A ( --- )
   0 BEGIN
     ." A"
     1+ DUP 10 =
   UNTIL
   DROP CR ;


: B ( --- )
   0 BEGIN
     ." B"
     1+ DUP 10 =
   UNTIL
   DROP CR ;


: ANNAT ( --- )
   BELL
   ." FEL!" CR
   1000 0 DO ( TOM LOOP )
     TASK
   LOOP ;

: LOOPA ( --- )
   100 0 DO
     I 1+ 3 .R ." : "
     KEY
     DUP 65 =
     IF
       A
     ELSE
       66 =
       IF
         B
       ELSE
         ANNAT LEAVE
       ENDIF
     ENDIF
   LOOP
   CR CR CR ;
```

1+ ökar värdet på stacken med ett. .R skriver ut det näst-översta talet i ett fält som är lika brett som det översta talet på stacken är stort, i det här fallet 3. KEY motsvarar GET i Basic, och lägger ascii-värdet på stacken. DUP duplicerar talet på stacken. DROP gör det omvända, dvs tar bort ett tal.

CR gör en radframmatning.

TASK är ett no-operation ord.

Observera att huvudordet måste stå sist.

För att åstadkomma ett mer lättläst program, kan man definiera ofta förekommande tal som konstanter (den möjligheten finns inte i Basic). Konstanter och variabler definieras lämpligen först i ett program. Talet 10, som används på flera ställen i exemplet ovan, kan läggas i en konstant:

```
10 CONSTANT TEN
```

Talet 10 läggs nu i 2 bytes som får namnet TEN. Eftersom 10 ryms i endast 1 byte, kan man också skriva:

```
10 CCONSTANT TEN
```

När en konstant exekveras läggs dess värde på stacken. Variabler definieras på liknande sätt:

```
0 VARIABLE TEMPORÄR
```

TEMPORÄR blir en 2-bytes variabel. Om TEMPORÄR istället skapas med ordet CVARIABLE reserveras endast 1 byte för värden.

När en variabel exekveras läggs addressen till dess värde på stacken. För att lagra värden i den använder man orden ! och C! (VÄRDE ADDRESS ---). Motsatsen utförs med @ och C@ (ADDRESS --- VÄRDE).

Det kan vara bra att kunna komma åt värden som ligger lite längre ner i stacken.

SWAP kastar om de 2 översta talen.

ROT roterar de 3 översta talen, så att det 3:dje hamnar överst, och de 2 översta flyttas ner ett snäpp.

OVER kopierar det näst-översta talet, och lägger det överst.

2SWAP, 2DUP, 2DROP, 2! och 2@ betraktar ett tal som 4 bytes långt, och används främst för 32-bitars heltal.

Fig-Forth är inte den enda Forthen, och inte heller den bästa. Två andra vanliga standarder är Poly-Forth och MMS-Forth, som båda bygger på Forth-79. En sådan Forth kommer antagligen att implementeras på ABC80.

I nästa avsnitt ska vi bla. titta på en del Forth-79-ord, och skapa en sk. CASE-sats. En sådan liknar en ON... GOTO-sats i Basic.

Robert Claeson, Boden

Radlängdskontroll i TV-editorn

För lång rad

Om man i Basic skriver textrader på fil och någon av raderna innehåller mer än 117 tecken (exkl Carridge Return och Line Feed) så får man problem när man senare försöker läsa filen. Läser man den på vanligt sätt med INPUTLINE så får man ERR 20, för lång rad, när man träffar på den för långa raden. Läsningen avbrytes och programmet stoppar och det går inte att läsa den text som ligger efter den för långa raden. Detta kan lätt hända när man arbetar med TV-editorn om man inte är uppmärksam. Har man råkat ut för detta så är det visserligen möjligt att "rädda" texten men man får då tillgripa metoden att läsa filen sektor för sektor med random access. Det blir ganska tidsödande och krångligt, de flesta ger upp och föredrar att skriva in texten på nytt. Men det är klart irriterande när man kanske har arbetat i timmar. Jag har sedan TV-editorn släpptes ut, från flera medlemmar mottagit önskemål om att få någon slags automatisk radlängdskontroll i TV-editorn. Nedan följer några programrader som man kan peta in i sin version av TVMAIN för att lägga in radlängdskontroll innan man skriver på fil.

Genom tilläggen nedan har kommandot ;S blivit ändrat och två nya kommandon ;P och ;T har tillkommit. ;P skriver filen på printer och ;T motsvarar det gamla ;S-kommandot.

När du i fortsättningen skriver på fil så kommer programmet att innan den frågar efter "Output file" att checka att ingen rad är längre än 117 tecken (119 inkl <CRLF>). Påträffas en sådan rad, kommer texten "Raden är för lång!" fram och markören stannar framför den för långa raden och du får möjlighet att dela upp den i mindre delar innan du skriver på fil.

När du skriver på printer genom kommandot ;P sker ingen radlängdkontroll och inte heller om du använder ;T. Ändra i rad 3127 om du vill ha en annan printerkod.

Andra buggar

Samtidigt kan det vara bra att rätta till några ytterliggare buggar som insmugit sig.

När man laddar in TV-editorn från diskett så finns det i programmet TV en maskinkodsrutin som laddar in TVSUBR.ABS. Rutinen gör ett inhopp på ett bestämt ställe i DOS-et. Denna rutin läggs in med hjälp av POKE-satsen i rad 300 (rad 1000 i den ursprungliga versionen).

```
POKE 65408%,33%,146%,255%,205%,94%,
108%,17%,240%,255%, osv
```

Nu har det visat sig att det finns olika versioner av DOS-et (med olika checksummor på samma sätt som tangentborden har olika checksummor för Basictolken).

I artikeln om TV-editorn i ABC-blad 1982:2 gavs ett tips att man för DOS märkta "681" borde ändra 5:te byten från 94 till 89 för att få inladdningen av TVSUBR att fungera riktigt. Nu har Sven-Åke Abrahamsson <medlem 105> omtalat att han fick det att fungera på sin FD2D sedan han ändrat byte 5 och 6 enligt följande:

```
POKE 65408%,33%,146%,255%,205%,194%,
108%, osv
```

För att sökning med kommandos skall fungera helt riktigt bör rad 2450 och 2480 i TVMAIN ändras enligt nedan. Vidare har kommandot ;X (skriv på fil och exit) blivit bortmaskat när TVLIB.MRG lades in. Vill man återställa detta kommando och få radlängdskontrollen att fungera även här bör rad 3275 läggas in. Eventuellt kan det räcka att göra motsvarande ändring i rad 3260 ((d.v.s för er som inte lagt in TVLIB.MRG) red).

Så här går det till att lägga in dessa ändringar

Gör LOAD av den version av TVMAIN du normalt använder. Gör LIST 20- och leta upp rad 3110 (eller 3119 i TVMAIN3) som innehåller

```
IF C$<>'S' THEN rrrr
```

där rrrr står för ett radnummer längre ned i programmet.

Ändra denna rad (3110 resp 3119) till:

```
IF C$<>'S' THEN 3125
```

Efter rad 3120 REM — save file lägger du till raderna 3121 - 3128 nedan.

```
3121 P%=1% : P1%=P%
3122 IF P1%<LEN(M$) THEN GOSUB 3690 ELSE
     3130
3123 IF P1%-P%<120% THEN P%=P1% : GOTO
     3122
3124 ; CUR(23%,0%)"Raden är för lång!";
     : RETURN
3125 IF C$<>'P' THEN 3128
3126 ; CUR(23%,0%)"Skriver på printer";
3127 X$='PR:' : GOTO 3240
3128 IF C$<>'T' THEN rrrr
3129 ;
3130 ; CUR(23%,0%);TAB(39%);
3135 ;
3240 PREPARE X$ ASFILE 1% : ; §1%;M$; :
     CLOSE 1%
```

I stället för rrrr skriver du in det radnummer som fanns i den ursprungliga raden "IF C$<>'S' THEN rrrr". Ändra i rad 3127 om du vill ha en annan printerkod.

Lägg gärna in även följande ändringar:

```
2450 P1%=INSTR(P1%+1%,M$,S$)
2480 P%=P1%-1%+LEN(S$) : GOTO 1700
```

```
3275 IF C$='X' THEN GOSUB 3121 : IF NOT
     (P1%-P%<120%) RETURN ELSE STOP
```

Därefter gör du SAVE med det programnamn du vill ge denna -din nya- version av TV-MAIN.

Lycka till!

Gunnar Tidner

INSÄNDARE TILL ABC-BLADET

"NJET!"

Varför lär jag mig aldrig av erfarenheten? Också i år stack jag fram hakan och fick mig den smäll som den avvikande så väl förtjänar.
Kanske har jag omedvetna masochistiska böjelser?

Konstigt att jag inte kan låta bli att argumentera för det jag tycker är rätt, när det ju är helt uppenbart att det inte finns någon marknad varken för mina ideer om föreningsdemokrati eller om klubbens anpassning till modernare datorteknik.
Skönt i alla fall att ha en fast punkt i tillvaron som man kan lita på i alla väder. Kärnan av styrelsen följde nämligen som vanligt traditonen att låta sig återväljas, efter att framgångsrikt ha avslagit alla motioner "i sin helhet" från personer utanför styrelsen. Man får väl gratulera?

Trots att jag står i ideologisk opposition till etablissemanget inom ABC-klubben, så har jag ändå förståelse för styrelsens nästan klassiska agerande.
Då är jag mer desillusionerad på de vanliga medlemmarnas ointresse för förbättrad demokrati, samt likgiltighet inför den snabba tekniska utvecklingen.

Är datoramatörer en särskild sorts människor som bara tänker binärt, och som är fullt tillfreds med tillvaron så länge man får sitta och knappa in spännande ASCII-koder på sitt snart antika tangentbord? Det kanske är för mycket begärt att datoramatörer skall vara akademiska humanister till sin intellektuella läggning, men man borde väl ändå ägna mer än en flyktig tanke åt en utveckling som i högsta grad kan påverka värdet av att ha en ABC-dator?

John Kvarnström

RAPPORT 1 ?

RAPPORT 2 ?

SAMLINGSNUMMER ?

På grund av det enorma beställningsgensvaret angående:
Rapport 1 (Disassemblern)
Rapport 2 (Fig FORTH)
och Samlingsnumret (80+81).

Är vi tacksamma om de vilka har betalt i förskott men ännu ej erhållit leverans vill kontakta:

Joe Johnsson per brev:

Joe Johnsson
ABC-klubben
Vidängsvägen 1
161 33 BROMMA

eller telefon 0756/44255

Ange medlemsnummer och inbetalningsdag (och Ert NAMN förstäss).

joe

RAPPORT 1 !

RAPPORT 2 !

SAMLINGSNUMMER !

Kasettproblem ABC80/800

ERR 35 och 37

Vid användning av ABC-80 kassettbandspelare, har vi fått en del uppgifter från medlemmarna om problem vid inladdning av program. Dessa problem yttrar sig som ERR 35 och ERR 37 vid läsning. Eftersom misstanke funnits att orsakerna varit fel vid kassettframställningen, har vi ansett det nödvändigt att göra en grundlig undersökning av orsakerna till dessa problem.
Det har inte visat sig vara det lättaste att snabbt och enkelt få fram ett svar på ...sa frågor eftersom problemen visat sig ...a flera än jag från början trodde. Jag kan redan nu ge en sammanfattning och en del tips som jag hoppas att ni som användare ska ha en del nytta av.

Standardbandspelare.

Till att börja med så är den bandspelare ... hittills använts som databandspelare ... modifierad standardbandspelare.
Anledningen till att man använt en standardbandspelare är enkel, man har velat få fram en användbar databandspelare till ett överkomligt pris. En riktig databandspelare kostar ju en hel del.

Nödvändigt med felfri dataöverföring.

Det som är viktigt när man spelar in data på band, jämfört med inspelning av tal och musik, är att man inte kan tillåta något som helst avbrott på överföringen.
Varje puls måste komma in till datorn och det får heller inte komma in en extra puls, t ex orsakad av en störning. Detta ger då omedelbart ett fel. En skada på bandet eller en fläck med dålig magnetbeläggning orsakar en s k dropout, med avbrott i dataöverföringen som följd.

Hastighetsvariationer.

Eftersom det i en hel del fall visat sig att band inspelade på ett kassettminne inte gått att köra på ett annat har jag trott det haft med hastigheten att göra, dvs ... bandspelarna har gått med olika hastighet. Det har dock visat sig att hastigheten inte har spelat så stor roll som svaj, dvs korta och snabba hastighetsändringar i samband med in eller avspelning av data. Orsaken till dessa hastighetsändringar redogör ... för senare.

Läsning av data från band.

Jag ska här enkelt försöka förklara hur läsning av data går till. På en tidsenhet sänder man en eller två pulser beroende på om det är en etta eller en nolla. Vid avläsningen låter kassettrutinen datorn hålla ett tidsfönster öppet som är drygt hälften av en tidsenhet. Om det i detta fönster kommer en puls tolkas den som en etta eftersom pulsen kom på halva tiden. Kommer det ingen puls tolkas det som en nolla. Varje block har en checksumma, som kontrolleras. Skulle checksumman vara fel uppstår ett ERR 35. Den inbyggda kassettrutinen medger inte att man fortsätter läsa övriga block även om de första och efterföljande blocken skulle vara riktiga. ERR 37 uppstår om läsfelet kommer innan datorn börjat läsa data inne i blocket.

Oförklarliga fel.

Jag har under undersökningens gång kört många kassetter och med hjälp av ett program kört ut ett protokoll på skrivare. Vissa kassetter har jag kört om och om igen för att utröna varför man ibland kan få ERR 35 och även ERR 37 på olika ställen på bandet och i olika block utan något påvisbart samband, och i många fall kunde inladdningen gå helt perfekt. Detta visade sig sedan bero på två olika orsaker:

Känslighet för nätstörning.

Det skulle visa sig att anledningen till dessa oförklarliga fel var nätstörningar, s k störpikar som alstrades av magnetventiler, transformatorer och dylikt. I mitt fall kom störningarna från oljebrännaren och tvättmaskinen. Dessa störningar gick följdaktligen via nätledningen rakt in i bandspelaren och störde läsningen av data. Kommer en sådan pik i tidsfönstret vid avläsningen kan alltså en nolla tolkas som en etta och man har ett error. Jag hoppas kunna återkomma i senare ABC-blad med tips hur man kan avhjälpa eller minska dessa störningar. Just nu kan jag bara ge rådet att stänga av alla maskiner med elektriska motorer, magnetventiler och annat som kan orsaka störningar på elnätet. Men eftersom elmotorer finns överallt i huset kan jag bara rekommendera att skruva ut alla säkringar utom den som går till datorn.

Vibrationskänslig.

Ett annat fenomen som kom fram vid proven var att bandspelaren vid in och avspelning är väldigt vibrationskänslig, dvs om bandspelaren står på ett bord eller liknande och man knackar ur pipan i askfatet eller råkar vidröra bandspelaren så kan man få ett läsfel. Här får man också vara påpasslig vid inspelning eftersom man då inte direkt upptäcker att det blivit fel i överföringen.

Mekaniska orsaker.

Bandet drivs i bandspelaren med kapstanaxel och tryckrulle, detta för att få en jämn och säker bandtransport. Den högra bandspolen som ska rulla upp bandet är kopplad med en slirkoppling till en drivanordning som roterar snabbare än själva bandrullen. Denna slirkoppling ska se till att bandet rullas upp mjukt och försiktigt på spolen utan allt för kraftigt drag mot tryckrullen. Den vänstra bandrullen är också försedd med en slirkoppling som tillsammans med tryckkudden i kassetten bromsar bandet något för att få ett jämnt och säkert tryck mot läshuvudet. Dessa mekaniska justeringar är utförda på det sätt som är vanligt för standardbandspelare avsedda för tal och musik. Dvs att draget på den upprullande spolen är tillräckligt kraftigt för att kassetter med ganska skiftande kvalitet skall kunna användas.
När bandspelaren används för dataändamål kan dessa slirkopplingar orsaka problem om de är för hårt justerade, så att draget mot tryckrullen blir för kraftigt. Kapstandrivningen klara då inte riktigt av att föra bandet i jämn hastighet. Då uppstår lätt det svaj jag tidigare talat om.
Jag hoppas kunna återkomma i senare nummer av ABC-bladet med noggrannare anvisningar om hur du justerar din bandspelare, men jag kan redan nu säga att det är ganska lätt att minska draget på högra bandspolen genom att vrida en justeringsanordning som sitter inne i bandspelaren i axelns förlängning. Det är en plastmutter som är lettrad för fingergrepp. Muttern är låst med en droppe låslack. Man kan pröva att skruva upp den ett halvt varv.

ABC-kassetter.

Jag trodde, när jag började med dessa undersökningar att orsaken främst var ABC-kassetterna. På dessa har, som säkert alla har märkt, funnits en hel del sprak och störningar som uppkommit vid kopieringarna och som ingen riktigt vetat orsaken till. Det visade sig dock att dessa störningar sällan har haft någon praktisk betydelse, utan felorsakerna har i de flesta fallen orsakats av bandspelarna eller nämnda störningar.

Billiga kassetter.

Eftersom det för varje utgivningstillfälle är ca. 3500 ABC-kassetter som skall kopieras så är det naturligtvis av avgörande betydelse att hålla ner kostnaderna. Därför har billiga kassetter använts. Eftersom klubben inte använt högsta kvalitet på kassetterna så kan det vara en del dropouts på bandet eller mekaniska fel på kassetten som har gjort att inspelningarna skadats. Det finns på andra sidan av bandet en reservinspelning med precis samma program. Hittills har ändå felprocenten varit låg. En del har säkert undrat varför inte baksidan börjar från slutet av kassetten (eller från början när man har vänt den). Detta beror på att inspelningen av baksidan sker samtidigt som framsidan, dvs två parallella kanaler spelas in framåt, och två baklänges.

Tröga kassetter.

En del kassetter går lite trögt, vilket ibland orsakar att kassetten kan stå helt eller delvis stilla vid kopieringen. Detta är inte alltid lätt att upptäcka vid kopieringen, speciellt om kassetten bara har hackat till på något ställe. Eftersom programmen ligger spegelvänt inspelade är oftast ett program som är skadat på ena sidan oskadat på den andra, förutsatt att inte programmet råkar ligga på samma ställe på andra sidan. Om kassetten går trögt i din bandspelare så försök att lätta något halvt varv på skruvarna i kassetten, och då främst den mittersta.

Reklamation av band.

Om ni skulle råka ut för något missöde liknande ovan kontrollera om programmet på andra sidan är oskadat och går att använda. I annat fall skickar ni in kassetten så får ni en ny.

Masterbandspelaren ombyggd.

Som tidigare nämnts hade ABC-kassetterna en del oförklarliga sprak och störningar. Jag har tittat på masterbandspelaren och hittat felet. Bandspelaren, som är en ombyggd Tandberg-bandspelare, var slarvigt ombyggd och spraket berodde helt enkelt på att en del trådar som använts vid ombyggnaden var oskärmade. På grund av felaktig impedans blev trådarna även mikrofoniska och uppfångade vibrationer, som sedan omformades till knaster och sprak. Troligen blev dessa "falska datapulser" inte tillräckligt kraftiga för att i någon högre grad påverka läsningen.
Dessutom saknades en omkopplare vilket inte gjorde det möjligt att göra masterinspelningen direkt från datorn, utan man fick alltid tidigare gå mellanvägen över en masterkassett och sedan kopiera till masterband. Nu kan emellertid masterbandet inspelas direkt från datorn och därför med högre kvalitet än tidigare.
Jag hoppas att ovanstående ska vara till lite hjälp för er som har haft problem med kassettbandspelarna. Om jag får tillfälle så återkommer jag i ABC-bladet framöver med mera detaljerade tips.
Lycka till!

Stig Löfgren <872>

JULNÖTEN

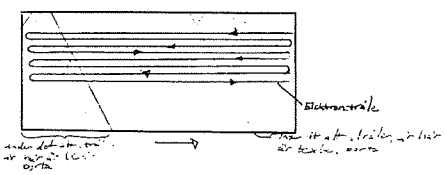
Hej!

Det här är förklaringen till fenomenet i GODJUL82 på ABC-kassett nr. 7:

Det tar nästan lika lång tid för ABC80 att utföra rad 20 som för elektronstrålen i bildskärmen att göra ett svep (inklusive horisontalåtergång och släkt tid i ramen), nämligen 64 mysek.
Om vi då tänker oss att datorn just har släkt texten och att elektronstrålen just är klar med linjen över första linjen på raden där texten skall finnas, så kommer den linjen att vara helt släkt. I och med att datorn inte riktigt hinner med att släcka texten när strålen börjar på andra linjen, kommer en liten, liten del av punkterna att slinka med i början på linjen. När så strålen kommer till nästa linje (3:e) kommer lite mer att plottas ut. På detta vis fortsätter det. Vid bildväxlingen kommer dock ytterligare 4608 mysek att gå bort. Denna förskjutning av tiden passar in med tiden för rad 20 på så sätt, att liten, liten bit mer av första linjen på raden där texten skrivs ut.

<2321> Mikael Djurfeldt

Angående "GODJUL82" på kassett 7, som kom förra veckan, så anser jag det vara ett klart exempel på "stroboskopeffekten". När man t.ex. mäter en rotation med ett stroboskop, och stroboskopets frekvens nästan sammanfaller med föremålet, som roterar, så verkar detta föremål rotera sakta. Texten försvinner, och dyker upp igen, under det att elektronstrålen sveper av skärmen, lätt i otakt med varandra. Den kan dyka upp när elektronstrålen ritar en annan del av skärmen, och försvinna innan strålen nått den, och kommer därigenom att vara osynlig, eller också dyker texten upp när elektronstrålen är mitt i den, så att bara halva syns. Då elektronstrålen och texten ligger nästan lika i frekvens, men bara nästan, verkar stället, där texten dyker upp, eller försvinner, att sakta röra sig. Eftersom elektronstrålen kommer senare, ju längre ner på texten den kommer, verkar texten att vara snett avskuren. (se bild)



Jag skulle också vilja passa på och tacka för alla de fina program som kommit på kassetterna under året.

<1426> Mats Öhrman

"GODJUL82.BAS"

Det är förhållandet till bildskärmens avsökningstid som återspeglas i detta fenomen: Dvs. datorn ger informationen: skriv rad, släck rad, skriv rad ...osv med en cykeltid som ligger mycket nära bildavsökningstiden. Orsaken till att det blir en sned interferenslinje är den att för varje avsökningslinje så hinner datorn mata ut ytterligare en bit av strängen, se fig.

Motsvarande gäller då också när strängen släks.
Genom att göra små förändringar i utskriftstiden (cykeltiden) så kan man få interferenslinjen att gå med olika hastighet och i olika riktningar.
För att få motsatt lutning på interferenslinjen skulle krävas att man matar ut strängen från höger till vänster.

<2206> Dan Johansson

Hejsan! Jag vet the svaro på GODJUL82. Först skriver den Godjul på rad 20, sedan börjar den om på rad 20 och raderar texten, sen skriver den ut samma sak igen osv.
Att det skrivs ut så festligt beror på att datorn är snabbare på att skriva resp. radera än vad TV:n är. TV:n hinner alltså inte med att släcka resp tända alla punkter förrän det är dags att släcka resp. tända dem igen.

Kalle Lindström

Ässå har vi fått svar från TMP!

"Fenomenet" i programmet GODJUL82.BAS på ABC-kassett nr 7 beror på att man får interferens med videosvepet.
Ett annat kul exempel på detta fenomen är:

```
 1 REM list  tmpskärm
10 T%=26%
20 A$="THE MAD PROGRAMMER STRIKES AGAIN!
   !! <<< " : REM LEN (A$) = EXAKT 40
30 ; A$; : A$=RIGHT$(A$,2%)+LEFT$(A$,1%)
   : FOR I%=0% TO T% : NEXT I% : GOTO 30
40 REM det gör sig bäst i 40 teckens mod
```

TMP

Övriga som svarat är:

<2815> Rolf Ernerfeldt

<2697> Rene Dröschler

<2728> Bengt Larsson

RÄTTELSE I FORTH MANUALEN

På sidan 60 saknas tecknet för ett kommando (P), det skall in före det andra n --- , och betyder: Put the following text on line n.

Kasettproblem

Magnus Matts
Engelbrektsg 8
783 00 Säter
<899>

83-02-07

ABC-Klubben

Hej!

För några dagar sedan fick jag ABC-kassetterna nr: 6 och 7 från er, det var alltså ersättningskassetter för de felaktiga som jag sände ert för en tid sedan.

När jag satte en av dessa i min kassettbandspelare fungerade den som vanligt inte som den skulle, det var det samma med den andra också. Det gick liksom inte att få kassetten att röra sig ur fläcken, bandspelaren orkade inte alls driva den runt. Att spola den gick med nöd och näppe. När jag såg detta som så många gånger förut blev jag naturligtvis en smula besviken, då ju alla era kassetter brukar innehålla så mycket roligt.
Nu gjorde jag inte som jag brukar, dvs skickade tillbaka kassetterna till er och skrev att det var fel på dem. Nej, den här gången tog jag i stället fram en av de få ABC-kassetter som jag under året fått som fungerade perfekt och jämförde den med de felaktiga kassetterna. Utåt sett var de helt lika. När jag satte den riktiga kassetten i bandspelaren så fungerade den som väntat perfekt. När jag bytte ut den kassetten mot den felaktiga så märkte jag hur den trycktes uppåt i riktning nakåt i bandspelaren. Detta gjorde att de två skruvarna som skall driva bandet runt inte orkade med att driva det. Efter att ha sett detta och tänkt en stund kom jag fram till att det måste vara kassetten som på något sätt var felaktig. För att lösa detta tog jag helt enkelt och skruvade sönder kassetten och kollade den innuti, detta gav emellertid inte mycket hjälp, men när jag tog och skruvade isär även den hela kassetten och bytte bandrullarna i denna mot rullarna i den felaktiga kassetten fann jag att den fungerade då jag skruvade ihop och körde den. När jag sedan satte bandrullarna från den hela kassetten och skruvade ihop denna fann jag att den inte fungerade alls. På detta sätt kunde jag kopiera över båda de felaktiga banden till andra kassetter, så nu slipper jag besvära er med två felaktiga kassetter igen.
Detta måste väl bevisa att det är något fel med den tekniska kvaliten på vissa kassetter och att det inte är fel på min bandspelare.

Jag hoppas att min berättelse skall kunna vara till någon hjälp för er, då ju ni håller på att utreda vad det är för fel på vissa kassetter. Om ni vill får ni gärna skriva om detta problem i kommande nummer av ABC-bladet, dvs om ni finner något värde i detta brev och dess berättelse om hur jag lyckades få två felaktiga kassetter att fungera med hjälp av en hel.

Läs kasetter i 2400 BAUD

Kassettrutiner som gör ABC80 kompatibel med ABC800-kassetter.

Klubben har tittat på ett par kassettrutiner som gjorts för att läsa kassetter skrivna med ABC800.

KAS800, som vi fått från Per Ahlin <911>, är en enkel rutin för enbart läsning av band i 2400 baud.
Se texten KAS: på annan plats i bladet.
Vid tester har programmet visat sig ha svårt att klara olika bandspelare, troligen behöver tidskonstanterna justeras. Programmet är gjort för checksumma 11273.
Programmet är inte särskilt långt och tar därför inte så lång tid att knappa in.
Prova!

Per Magnusson har gjort en rutin på drygt 1 Kb som kan ställa om hastigheten från 700 till 2400 baud i 6 steg. Programmet kallas FASTCAS och är enligt Magnusson inte helt färdigutvecklat. Vi kommer att prova programmet så snart vi erhållit en testversion. Dock är Magnussons program inte kompatibelt med ABC800 eftersom headerblocket också skrivs i högre hastighet.
CasTurbo skriv- och läsrutin.
C-system har givit ut ett programpaket som kallas CasTurbo. Programmet som både läser och skriver i 2400 och 700 baud är skrivet helt i assembler. Det är 1.25 Kb stort, med många intressanta finesser som man kan ha nytta av.
Övriga program i programpaketet:
RUN2400 är hela läsrutinen från CasTurbo som läser alla filer i 700 eller 2400 baud skrivna med ABC80 eller ABC800.
CASLIB.BAS är ett inte särskilt användbart libprogram för kassett där klubben tidigare har givit ut andra liknande program.
CASCOPY.BAS kopierar alla filtyper från kassett till skiva.
CASCAS kopierar alla filtyper från en kassett till en annan.
FILECOPY.BAC kopierar alla filtyper från floppy till kassett eller från floppy till floppy.
LÄSPRINT.BAS skriver ut filer från kassett eller skiva på bildskärm eller skrivare.
Programmen kräver att CasTurbo eller RUN-2400 är inlänkade och arbetar i 700 eller 2400 baud.

RUN2400.BAC på ABC-kassett.
Klubben har erbjudits RUN2400, CASCOPY och CASCAS för distribution på nästa ABC-kassett.
En del av dessa program har motsvarande funktioner som program utgivna på ABC-kassett, men observera att tidigare program inte klarar dataöverföring i detta snabba tempo.

ABC-kassett i 2400 baud.
Möjligheten att använda RUN2400 är intressant och klubben har en längre tid gjort omfattande försök med snabbkopiering i 2400 baud. Avsikten är att i framtiden ge ut kassetter i 2400 baud och därmed rymma fler program på varje ABC-kassett.
Dock återstår en del olösta problem och en närmare redogörelse kommer senare.
Av de kassettrutiner för läsning i 2400 baud som provats så är utan tvekan RUN-2400 det mest professionella och det som också fungerat bäst på olika bandspelare.
Programmet kommer på nästa ABC-kassett.

Stig Löfgren

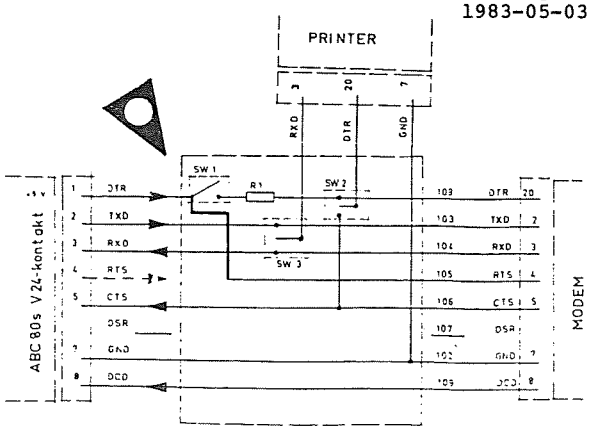
TIO SMÅ HASSELNÖTTER

1. Kan du förklara vad som egentligen sker vid OUT 57,79 ?
2. Hur sparar man en fil som börjar med en siffra på skiva ?
3. Vilket är det snabbaste sättet att POKA i bildminnet på R%,K% ?
4. Vad har man för nytta av instruktionen HALT i assemblerprogram ?
5. Hur nollställer man en minnesarea med LDIR ?
6. Vilket är det kortaste programmet som ger ERR 3 ?
7. Hur får man den längsta BASIC-raden?
8. Hur listar man högst tio rader av ett program genom enbart LIST ?
9. Hur kan man få listningen att börja så att inga rader försvinner ?
10. Vilken ASCII-kod 0<=x<=127 kan man inte generera från tangentbordet ?

Insänt av <592> Nils Häggblom,
Östersundom, FINLAND

Modemproblem

Många har den senaste tiden haft problem att få modemet att fungera korrekt vid filöverföring, kommandona **GETFIL SENDFIL FILELIB.**
Jag tror att vi kommit på en möjlig felkälla.
När programmet ABCTRANS eller motsvarande går i lokal mode så kommer bit 4 i V24-porten att sättas av drivrutinen ABCV24 och signalen **RTS** på stift nr 4 i den 9-poliga kontakten blir låg.
Denna signal går ofta in på modemets ledning CCITT 105 (stift 4 i den 25-poliga modemkontakten). Vissa modem är internt byglade så att signalen 105 alltid är ON, andra är byglade så att 105 styrs av signalen **RTS** från datorn.
De som har haft sina modem byglade så att 105 alltid är ON har inte drabbats av detta fel. När medlem 2859 byglade om sitt modem så fungerade allt som det skulle.
Om man inte kan eller vill bygla om inne i modemet kan man klara sig genom att ändra modemkabelns anslutning till ABC-80's V24-kontakt. Gör så här:
Lösgör anslutningen av den ledning som går till stift nr 4 och **anslut den i stället till stift nr 1 DTR** i den 9-poliga kontakten.



>>njaeee--bäst??<kolla KAS>>!! (1182)

FILELIB

```
* ABC-klubben Databank Library *
        1983-05-07
Drive 0
CENTKORT.SÄL 4     FRÅGA    .SVA 4
FRÅGA    .FRÅ 4    EJSVAR   .QZ  8
AMAZING  .GAM 28   JUPITER  .SÄL 4
ROBERT   .DÖD 4    SORT     .FRÅ 4
STLMMÖT  .INF 8    MONITOR  .FRÅ 8
QZGRATIS.INF 16    TJUT     .800 8
NOT      .ZZZ 4    MYSTISKT.FRÅ 4
FEMKAMP  .GAM 52   ABCV24   .SVR 4
PASCAL   .ABC 4    DISKREG  .UTL 28
FLUGAN   .INF 4    FLUGAN   .GAM 32
FILTITT  .INF 12   FILTITT  .BAS 8
ABC800   .SVR 4    MRX      .III 4
MRX      .FRÅ 4    BASICII  .SVR 8
BANDSPEL.800 8     BASICII  .NÄR 4
FILNAMN  .SVR 4    FELEN    .INF 8
NYPROM   .800 8    BASICII  .KOM 8
BASICII  .TXT 20   NYLINJE  .TXT 8
ABCKLB83.INF 12    FILTRANS.80K 32
BOFA     .SVR 4    DATABANK.INF 12
STHRAP1  .INF 12   DJUR     .BÅS 24
DJUR     .REM 8    SPIDER   .BRA 4
FILELIB  .HLP 4    SPIDER   .GAM 32
SYSTIME  .800 8    SNÖ      .800 8
BOFA     .FRÅ 4    LOOK     .UTL 16
DELREN   .UTL 12   NYCHANGE.UTL 12
KOLUMNNR.80T 4     ARCADE   .INF 8
YATZY    .800 36   BLADLIB  .REM 8
BLADLIB  .TXT 48   SPEL     .BRA 4
LUXOR    .TXT 8    VÄRD800  .SVR 4
PROGBANK.INF 4     VÄRD800  .FRÅ 4
MÖNSTER  .800 16   ABCV24   .INF 4
KLANTIGT.PRO 4     FILEPRO  .REM 4
FILEPRO  .BAS 8    ERAFILE  .BAS 12
ERAFILE  .REM 4    MASKEN   .800 32
CURSORST.FRÅ 4     RIVAN    .LÅT 32
ABCV24   .FEL 4    KABEL    .HLP 4
CURSORST.SVR 4     LÄSRAM   .800 20
CURSORST.YRN 56    FILSTAT  .800 28
DISKSTAT.800 28    NEWFILES.REM 12
FILELIB  .800 24   NEWFILES.800 28
VEDIT    .FRT 12   EXSCREEN.REM 8
EXSCREEN.BAS 8     FILTRANS.800 48
ABCMINI  .800 8    PROTALL  .800 12
CASDISK  .800 40   RITA     .SUB 12
ABCTRANS.800 24    MODEMBUG.INF 8
KOLLDIM  .800 12   VARNING  .800 4
MENY     .800 16   COPYFAST.800 16
GÖRCAS80.REM 8     GÖRCAS80.BAS 24
ABC800   .FRÅ 4    LOGOUT   .INF 8
GÖRITH   .BAS 8    GÖR      .ABS 8
QZANVISN.TXT 16    QZANSVAR.TXT 12
TKN80LIB.BAS 12    QZENTRAL.INF 12
FILELIB  .INF 8    CURSOR   .INF 12
SAFT     .BAS 24   CASTRANS.800 12
WRITE    .YES 4    SYSDIR   .SYS 4
     2524 av 3944 sektorer kvar
```

Anm En större utgallring av filer som legat länge på Monitorn gjordes 1983-05-03

ABC-KASSETTER I 2400 BAUD??

Åtta ABC-kassetter utgivna på två år.
Klubbens satsning på utgivning av program på kassett har varit lyckad och mycket populär bland medlemmarna.
Under den senaste tvåårsperioden har åtta kassetter utgivits, vilket är i genomsnitt fyra per år och mer än de ca. tre per år som planerades från början.
De flesta programmen har skickats in till klubben av intresserade medlemmar som gjort programmen dels för egen del men ibland speciellt med tanke på klubben.
Många program har lagts in på klubbens monitor och har därefter genomgåtts av screeninggrupp för att sedan slussas ut på kassetterna. En del av programmen är sådana som har friköpts av klubben för just detta ändamål.

ABC800 program på ABC-kassett!
Klubben kommer att fortsätta sin satsning på kassetter där nu även en hel del ABC800-program börjar komma in. Satsningen på program för ABC800 kommer att fortsätta och vi är här tacksamma för bidrag från medlemmarna. Redaktionen är också tacksamma för artiklar om ABC800.

Kompatibilitet ABC80/ABC800 ?
Program som är skrivna i ren BASIC på ABC80 går oftast köra direkt på ABC800 eller med mycket små ändringar. Förutsättningen är givetvis att programmen har sparats med LIST, eftersom kompileringen för ABC80 och ABC800 är olika. Vid inladdning av ett program i ABC800 markeras varje felaktig rad med ett ? först på raden om någon instruktion inte skulle stämma med ABC800:s dialekt. Ett felmeddelande skrivs även ut på skärmen.
I möjligaste mån görs sådana ändringar av screeninggruppen vid genomgången av programmen så att de blir direkt körbara på både ABC80 och ABC800, men i vissa fall krävs alltså en ändring för att kunna köra programmen.
Program med PEEK och POKE eller om man använt sig av diverse smartprogrammering kan däremot vara jobbiga att ändra. Programmen kommer i möjligaste mån att vara märkta om de speciellt är avsedda för den ena eller den andra datorn.

Hastighetskompatibilitet 2400/700 baud.
Den inbyggda kassettrutinen för ABC80 är som bekant 700 baud, medan ABC800:s kassettrutin går med 2400 baud. Men den kan dock läsa kassetter i 700 baud. Detta har man i ABC800 löst genom att headern alltid ligger i 700 baud. I den finns information om vilken hastighet det efterföljande programmet är sparat med. Detta gör ABC800 hastighetskompatibel med ABC80, men inte vice versa, eftersom ABC800 alltid skriver i 2400 baud.

Ny kassettrutin för 2400 baud på ABC80
C-system i Täby har tagit fram en ny kassetthanteringsrutin som kallas CASturbo. Med denna rutin kan man både läsa och skriva i 2400 baud, och därmed blir kassetthanteringen på ABC80 helt hastighetskompatibel med ABC800. Programmet kan skriva i såväl 700 som 2400 baud, men när det gäller skrivning med den högre hastigheten krävs en mindre modifiering av bandspelaren för att förbättra frekvensgången vid skrivning. Vid läsning görs ingen förändring. Med programmet som säljs medföljer de motstånd som behövs för modifieringen.

Läsrutin i 2400 baud på nästa ABC-kassett!
C-system har sänt klubben programmet RUN2400 som är en komplett läsrutin för att läsa kassetter inspelade med ABC800 eller det nya CASturbo. Programmet har automatisk avkänning för 2400/700 baud.

Försök med snabbkopiering i 2400 baud
Med tillgång till läsrutinen RUN2400 har klubben en unik möjlighet att ge ut ABC-kassetterna i 2400 baud.
Fördelarna skulle vara flera. Vi skulle få plats med minst 2.5-3 ggr mer per kassett, vilket skulle bli en besparing, både på porto och kassetter. Det skulle kanske kunna innebära att vi fick råd att använda kassetter av lite högre kvalitet.

Tekniska förbättringar måste göras.
För att kunna genomföra detta projekt har det varit nödvändigt att i detalj gå igenom och förbättra den tekniska kvalitén vid framställningen av ABC-kassetterna.
Till att börja med så har redan masterbandspelaren byggts om. Detta dels för att ta bort de störningar som funnits på tidigare utgivna ABC-kassetter (se annan artikel i tidningen), dels för att förbättra frekvensgången på inspelningskanalerna. Det senare för att kunna skriva med högre hastighet.

Eftersom en del svaj uppstår senare, vid snabbkopieringsfasen, är det int lämpligt att använda en ABC80 vid framställande av master för 2400 baud. Detta på grund av klockinterruptet, som normalt orsakar en del svaj på utgående datapulser. Därför avser vi att använda en ABC802, som använder en inbyggd SIO-krets för samma ändamål, vilket ger ett betydligt stabilare pulståg. Denna del av kedjan fram till ABC-kassetten, nämligen överflyttning av programmen från flexskiva till masterbandet via ABC802 är ännu ej klar.
Försök har däremot gjorts att med hjälp av CASturbo och FILECOPY kopiera ner filer från skiva till masterband, för kopiering på den vanliga snabbkopieringsanläggningen. Där kopieras tjugoen kassetter samtidigt och med 16 ggr högre hastighet än normalt.

Vissa problem återstår att lösa!
Försöken har varit långt ifrån problemfria. Tidigare har programmet CASDISK använts för att flytta data från band till skiva, då har flexskiveenheten hunnit skriva blocken i den takt de läses från bandet. Tyvärr har inte CASDISK klarat det nya tempot, som är mer än 3 ggr högre (ett block per sekund!). Därför läses nu först 16 block in i minnet, sedan stannas bandspelaren och alla 16 blocken skrivs ner på skivan. Sedan upprepas proceduren tills filen är slut. Detta är nu utprovat och fungerar bra.
Vid de prov som utförts har det dock visat sig att kopieringskvalitén varit ojämn. Detta beror på att mer än hälften av kopieringsstationerna inte klarat den höga hastigheten, vilket tyder på att kopieringsutrustningen måste gås igenom ordentligt innan kopiering sker. Men de kassetter som fungerat har haft en mycket bra läsbarhet och fungerat väl.
Här vill jag dock säga att om vi skall gå över till 2400 baud, måste läsbarheten på kassetterna vara minst lika bra eller bättre än de tidigare utgivna kassetterna.

Mindre läsfel i 2400 baud ?
Man kan här fråga sig om inte läsbarheten skulle försämras om man ökar hastigheten, med den kassettkvalitet som för närvarande används. Svaret är kanske lite svårt att ge, men man kan faktiskt säga att om kassetterna har en viss procent dropout kommer den procenten att minska med en faktor två till tre, eftersom vi rymmer så mycket mer data på de felfria banden.

Är bandspelarna ett större problem?
Ett annat mera svårlöst problem som kommit fram under proven har varit att mekaniken i bandspelarna varit ojämnt justerad (se artikel om kassettproblem i detta ABC-blad). Det tycks vara så att om slirkopplingen för den högra bandspolen är justerad med för hård upprullning orsakas någon typ av svaj eller ojämn bandhastighet, som i vissa fall gör att bandspelaren kan läsa band i 700 baud men inte i 2400. Hur stort problemet är vet vi ännu inte, därför att vi bara har upptäckt det i något fåtal fall. Vi kommer dock att prova ett större antal kassetter för att få en bredare erfarenhet innan kassetterna slutligen ges ut i 2400 baud.

2400 baud på ena sidan, 700 på baksidan
Om vi kan lösa de sista tekniska problemen kommer den första kassetten som ges ut att ha programmen inspelade med 2400 baud på ena sidan, och reservprogrammet på baksidan i 700 baud. Då är det viktigt att medlemmarna meddelar kassettredaktionen om det har gått bra att läsa in programmen i 2400 baud, eller om man måste använda baksidan. Detta kommer att bli av avgörande betydelse om vi fortsättningsvis kan distribuera programmen i 2400 baud. Vi tackar på förhand för den hjälp som ni ger oss!

Ny bandspelare från Luxor.
Som en del redan vet finns inte längre den nuvarande bandspelaren i produktion hos Luxor. Många går även och väntar på bandspelare. Jag har haft kontakter med Luxors tekniska avdelning, och man säger att man nu efter omfattande utvärderingar har hittat en bandspelare till ett överkomligt pris, som skall vara bra för dataändamål. Priset på den bandspelaren är ännu ej fastställt, men lär ska ligga vid ca. 800 kronor.

De flesta ABC800 utan bandspelare.
Eftersom de flesta ABC800-ägarna inte har någon bandspelare, får klubben ofta frågor om vi kan distribuera programmen på diskett. Svaret måste tyvärr i dagens läge bli nej. En försiktig uppskattning av hur många som skulle vilja ha programmen distribuerade på diskett, då naturligtvis även ABC80-ägare, kan bli uppemot ettusen medlemmar. Det är därför inte möjligt, med nuvarande resurser, att kopiera så många skivor, speciellt med tanke på att en del medlemmar har enkeldensityskivor. Då ryms inte alla program från en kassett på en skiva, utan man måste använda två.

Kassettdistribution i framtiden.
Vi får nog inse att kassettdistributionen ändå är den enda lösning som finns inom överskådlig framtid, och här kan 2400 baud vara ett utmärkt hjälpmedel för att kunna distribuera flera program till en överkomlig kostnad.
Skriv gärna till mig om ni har några synpunkter eller problem. Det är värdefullt för mig i samband med detta arbete.

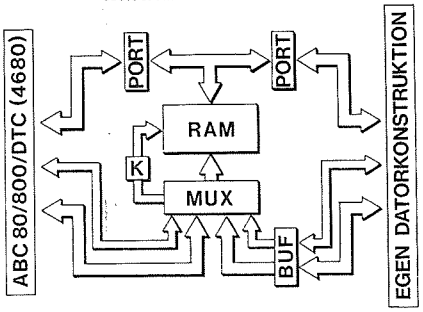
Stig Löfgren <872>
Adress: Måsvägen 16, 183 51 Täby

MinnesEmulator

En MinnesEmulator som gör ABC 80/800/DTC användbar vid utveckling av egna mikrodatorkonstruktioner har tagits fram av Reologi och Mätdatorteknik HB. Emulatorkortet passar även till Databoards 4680-buss.

Emulatorkortet innehåller upp till 4kbyte statiskt RAM-minne med dubbel data och adressbuss. Det kan därmed adresseras med läs- och skriv-signaler från både utvecklingsdatorn och det prototypkort man vill testa. Emulatorkortet, som är klart för insättning i ABC80/800/DTC Expansionsenhet, är bestyckat med en 24-polig DIP-kontakt på den bandkabel för anslutning till minnessockel i prototypdatorn.

MINNES EMULATOR



MinnesEmulatorn lämpar sig för emulering av EPROM,

- utveckling av maskinvara där korta program med snabba loopar enkelt kan läggas ut för testning av CPU och I/O -signaler. Dessa signaler kan då enkelt följas på ett vanligt oscilloskop. Program kan modifieras under körning.

  utveckling av programvara med hela prototypsystemet (utom EPROM:et) i funktion. Således är även CPU:n med vid programkörningen. Här skiljer sig MinnesEmulatorns funktion från en I:C.E.

och även för andra uppgifter,

- överföring av data och program mellan 2 olika datorsystem.
- emulering av RAM-minne.
- programmerbar teckengenerator för videokort.

Följande minnestyper är pinkompatibla med MinnesEmulatorns sockelanslutning:

EPROM: 2716, 2732, 2764, 27128, 2508, 2516, 2532.

EEPROM: 2816, 2817.

SRAM: 2128, 4016.

Emulatorn kan enkelt anpassas till andra datorer som Apple, M85/850 och PET.

Bohlin Reologi
Box 212
240 15 Sandby
046-62 444 eller 62 113

SUPER-SMARTAID gör ABC80 till stor dator

Med det nya programmeringshjälpmedlet **SUPER-SMARTAID** får ABC80 egenskaper, som betydligt större datorer saknar. Avavc erad bildskärmseditor, multitasking, funktionstangenter och automatisk initiering är några egenskaper som **SUPER-SMARTAID** ger.

SUPER-SMARTAID är ett nytt programmeringshjälpmedel för ABC80, utvecklat av OWOCO AB.

SUPER-SMARTAID bygger på den Basic som finns i ABC80. De program som redan är utvecklade, kan användas även i framtiden. Samtidigt får ABC80 samma egenskaper som betydligt större datorer.

SUPER-SMARTAID bygger vidare på OWOCO's mångåriga erfarenhet och tidigare hjälpmedel för ABC80.

Datorns ordinarie arbetsminne belastas inte. **SUPER-SMARTAID** lagras i ett separat minne. Den avancerade bildskärmseditorn gör att hela arbetsminnet kan utnyttjas för program eller filer, eller för att skriva källkod i, när man programmerar i assembler.
Möjlighet att programmera egna funktioner på tangenterna. Utskrift av en del av en fil på skrivare. Felsökning med kommandot Step/Trace som går att följa på skärmen utan att menyn förstörs är några av egenskaperna.

Vid felsökning med Trace kan man stega fram programkörningen en rad i taget, utan att förstöra mätvärden eller menyer. Hela Basic-raden kan visas före exekvering, något som även kan utnyttjas vid vanlig programkörning.

En fil som lagrats på flexskivor kan visas på skärmen, utan att minnesinnehållet i datorn påverkas. Bildskärmens innehåll kan enkelt dumpas på en printer även under programkörning.

SUPER-SMARTAID har automatisk initiering vid påslag och automatisk start av jobfil. I CMOS-minnet lagras vissa pekare, och de tangentfunktioner man själv programmerat in. Det gör att dessa är direkt tillgängliga nästa gång datorn slås på. 1kByte av CMOS-minnet kan också användas för datalagring på samma sätt, data är tillgängliga direkt vid start.

Vissa uppgifter kan utföras samtidigt, en sorts multi-tasking. Utskrift av textfiler kan ske samtidigt som programmering pågår. Bakgrundsjob i maskinkod kan också exekveras samtidigt som programmering pågår.

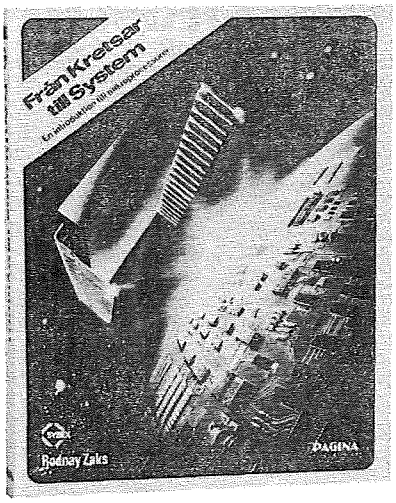
Utskrifter på skrivare sker med hjälp av en speciell utskriftsrutin, där en rad parametrar kan väljas: hastighet i baud, sidlängd, radlängd osv.

Editorn är anpassad för 80 teckens bildskärm. Redigeringsfunktioner och sökbegrepp är förbättrade.

Assembleringar, tester av program och mycket annat utförs lätt på ABC80, med hjälp av **SUPER-SMARTAID.** Jobbströmmar kan användas med vanliga kommandon och med villkorssatser. Även inmatningar i program kan ske från jobströmmen. Kort sagt, med **SUPER-SMARTAID** får man en nu dator utan att skaffa någon ny dator.

SUPER-SMARTAID marknadsföres av:

OWOCO AB
Kvarnbergsvägen 25
141 45 Huddinge
Tel. 08-7740290



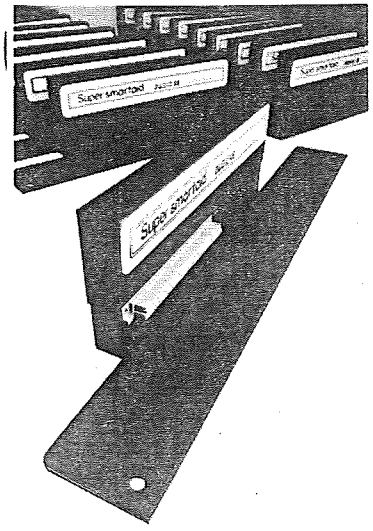
press-info

Nu har den kommit ut på svenska, boken **"Från kretsar till system".** Den är avsedd för nybörjare och alla som vill lära sig hur datortekniken fungerar. På ett enkelt och överskådligt sätt presenterar Rodnay Zaks mikroprocessorer ocn kringkomponenter. Boken börjar med grundläggande begrepp och förklarar mikroprocessorns inre funktion. De olika systemkomponenterna gås igenom, en utvärdering av olika mikroprocessorer görs, användningsområden för mikroprocessorer och metoder för anslutning av yttre enheter beskrivs, liksom programmering och systemutveckling i olika steg.

Varje kapitel är progressivt: från grundläggande definitioner till avancerade applikationer - från principer och metoder till faktiska exempel.

Denna bok ger alla chansen att följa med den svindlande snabba utvecklingen på mikrodatorområdet.

Från kretsar till system av Rodnay Zaks
Översättning: Anders Thulin
Distribution: Trim Marketing AB
Box 49035, 100 28 Stockholm
Tel: 08-54 00 10



0-262 K-BYTES EPROM UTVIDELSE FOR ABC80

Efter at have arbejdet med E-PROM ekspansionskort til ABC80 igennem laengere tid, måtte jeg efterhånden erkende overfor mig selv at denne traditionelle lösning havde store begraensninger.

Normalt er hele adresseområdet fra 16384 til 31743 anvendeligt, ~åfremt man ikke benytter printer, floppy-disk eller IEC-udstyr. ellem adresse 24576 og 31743 er der reserveret plads til disk-operative systemer, IEC-udrustning, printer, smartaid, superbasic-rutiner og m.m.

OK! - Så har vi trods alt 8 K-bytes fri mellem 16384 og 24575, men önsker vi at anvende 80 tegn til vor ABC80 forsvinder yderligere 2 K til ekstra billedlager. Derfor har jeg valgt kun at anvende adresseområde 16384-20479, som kan ligge i en 2732 eller 2732A E-PROM.

For stadig at kunne laegge store programmer i E-PROM, må andre lösninger altså tages i brug.

PIO'ens I/O kortvalgs procedure er en af de muligheder,som kan anvendes sammen med adressekodningen. Dette giver muligheder for at lagre intil 16 K eller 32 K-programstörrelser i E-PROM, afhaengig af egen ram udvidelsede. Forskellige E-PROM udvaelges derefter ved hjaelp af I/O chipselect (CS), som åbner adgang til den valgte E-PROM.

16 K eller respektive 32 K-programstörrelser kan nu direkte ilaegges E-PROM, uden begrensninger af adresserings område.

Princippet virker på den måde, at når 1. E-PROM's 4 K-bytes er henter op i arbejdslageret, sker chipvalg til naeste E-PROM, og disse naeste 4 K-bytes stakkes ovenpå de förste 4 K i arbejdslageret og så fremdeles.

Lyder det besvaerligt?

Ja - men for Z80 CPU'en er det ingen sag. Den tid det tager, at hente et stort program er ofte kortere end den tid, t tager at skrive run og taste return i basic.

Diagrambeskrivelse.

I opstillingen ska 2 betingelser opfyldes, nemligen chipvalg (kortvalg) og den korrekte adresse forespörgsel. Valget af chip kan ske direkte med kommando eller sats i program. Når chipselect (CS) laegges ud på bussen tolker alle kort, som er tilsluttet, den erdi der udsendes på databussen, og det kort, som er koddet med denne vaerdi, udvaelges. CS er altså en engangs forespörgsel, hvis signal fra PIO'ens outport 1 hentes fra stift A23 på bussen. Instruktionen out 1,X giver adgang til den önskede E-PROM og X kan have en vaede mellem 0 og 63, altså 64 muliga valg. Reset eller nulstilling sker med "; INP (7)".

Inden vaerdier vaelges for önskede kombinationer, skal man vaere opmaerksom på, at printerinterface og andra kort kan vaere koddet med en sådan kortvalg funktion, og sådanne vaerdier må der naturligvis tages hensyn til, for at der ikke skal ske kollision af flere kort.

I övrigt kan jeg anbefale at studere Åke Westh's bog "Styr og mål med ABC80", som kan give laeserne mange gode ideer.

Til lagring af programmer i E-PROM har jeg valgt 2732 og 2732A (4096 bytes), og på hvert europakort (10*16 cm) sidder 4 stk. parallel. Chipselect (pin 18) styres af I/O chipvalg. I diagrammet er A og B forbindelserne direkte henvisning til pin-tilslutningerne på bussen. Adresse tilslutning til IC 1 og IC 2 sikrer bevaegelse inden for det tilladte adr.område 16384-20479.

I/O signalet på A23 giver clockpuls til flip-flop i IC5 og IC6 og den datavaerdi som ligger på databussen A8-A13 dekoddes af IC4 og er bestemmende for hvilken af de 4 flip-flop, som önskes udvalgt. IC8's logik sikrer, at der kun er 1 E-PROM, som kan aflaeses. IC7 kan undvaeres, idet den sammen med de 4 lysdioder virker som indikator for den E-PROM, som er valgt.

IC4 kan koddes og dekodes efter följende sanhedstabel:

74LS138

Input						Output							
	-	-				-	-	-	-	-	-	-	-
Symbol:	E1	E2	E3	A0	A1	A2	0	1	2	3	4	5	6	7
IC pin:	4	5	6	1	2	3	15	14	13	12	11	10	9	7
	0	0	1	0	0	0	0							
	0	0	1	1	0	0		0						
	0	0	1	0	1	0			0					
	0	0	1	1	1	0				0				
	0	0	1	0	0	1					0			
	0	0	1	1	0	1						0		
	0	0	1	0	1	1							0	
	0	0	1	1	1	1								0

I diagrammet er följende data forbindelser anvendt:

D5	D4	D3	D2	D1	D0	Pin out	
0	0	1	0	0	0	15	
0	0	1	0	0	1	14	
0	0	1	0	1	0	13	
0	0	1	0	1	1	12	
0	0	1	1	0	0	11	(Ej anvendt)
0	0	1	1	0	1	10	(Ej anvendt)
0	0	1	1	1	0	9	(Ej anvendt)
0	0	1	1	1	1	7	(Ej anvendt)

Reset impuls af sam'tlige Flip-Flop kommer på A15.

I diagrammet er de 4 E-PROM valgt som chip nr. 8-11.

Hvis vi önsker at hente et program fra chip nr. 8, kan dette ske programmaessigt eller som kommando: out 1,8 (return), og lysdioden for nr. 8 taendes og derefter: ; CALL(16384) (return).

Et maskinprogram som kaldes i startrutinen henter det önskede program ind i arbejdslagret.

Större programmer som fylder mere end der kan vaere i en E-PROM, deles blot op i flere E-PROM. Et program som f.eks. TV-editoren kan med alle dens faciliteter laegras i 3 stk. E-PROM, og vaere placerad som chip nr. 9-11.

Opstart af dette program vil ske på följende måde:

OUT 1,9 (Return)
; CALL(16384) (Return)

Og mindre end 1 sek. efter at der tastet return er programmet köreklar. Smarte funktioner som: ;K9(Return) kan kombineres med ABC80 interruptvektorer og bruges som kommando til valg af chip nr. og CALL til adressen, men derom i en senere artikel.

Maskinprogrammet, som henter de data der ligger i E-PROM, kan udföres på mange måder og Z80 CPU'en giver muligheder for at hente store datamaengder med blot nogle få instruktioner.

Det er naturligvis ikke nödvendigt at hente alle programmer op i arbejdslageret. Mindre programmer som f.eks. hjaelpare og andre rutiner kan foreblive i E-PROMMEN, når disse er assembleret om, men större programdele hentes til bearbejdning i arbejdslageret.

Opstartsrutinen som skal ligge i den förste E-PROM indeholder initering og en eller flere block-move-instruktioner, samt chip valg. Programmet kan f.eks. se sådan ud:

```
EI
LD HL;(65063)
LD SP,HL
LD HL,65088
PUSH HL
LD HL, XXXXX      (E-PROM startadresse)
LD DE, 32768      (Startadresse 49152 eller 32768)
LD BC, XXXX       (antal bytes som skal hentes)
LDIR
LD A,10           (skift til chip nr. 10)
OUT 1
```

Herefter följer naeste block-move, som stakker naeste E-PROM's data ovenpå den forrige i arbejdslageret.

Pegere som eofa og heap reetableres fra adresse 65054-65057, og sidste instruktion er RUN.

Da der efterhånden forefindes mange differentieret adresser i de 4 ROM som ABC80 er bestykket med, vil jeg her anvise en metode som kan bruges wanset forskellige checksumme, ved at

FrågeBITen.

Ja, då är det dags igen efter sommarens heta dagar när man har legat i hängmattan och haft det skönt. Första frågan efter sommaruppehållet kommer från Knut Olsen-Solberg <3265> i Trondheim.

Fråga:

En av våre 10 ABC 80 skal nå kjöres med lokalprinter i serie over V24-kontakten. Alle våre maskiner har TKN 80, og jeg kan tydeligvis ikke benytte klubbens program 'ABC V24' (heller ikke T80PRT.) Men jeg har hört at Dere har et lignende program som kan brukes på maskiner med TKN 80, og som man kan få tilsendt på forespörsel. Hvis dette er korrekt, er jeg svärt takknemlig for å få dette på en kassett.

Svar:

Det går alldeles utmärkt att använda klubbens program ABCV24 som printerrutin både om man har 40- eller 80-teckensskärm. Det enda kravet är att printern är inkopplad via V24-kontakten (den lilla nio-poliga kontakten längst till höger bakifrån på datorn sett) och att man väljer rätt parametrar vid initieringen.

När man använder ABCV24 som printer rutin skall man i stället för menyval 3 ange 0 eller 1 beroende på om man skrivaren kräver handskakning eller inte. (Handskakning=när datorn och skrivaren "kommunicerar" med varandra och talar om att de är klara att sända, ta emot tecken osv). För närmare beskrivning hänvisas till ABC-bladet nr 2,1982 sid 22. De som inte har det kan få det genom att beställa Samlingspaket 2.

Någon frågade efter telefonnumret till Nyfors Data Science som säljer SuperBasic och adressen dit är:

Nyfors Data Science
Nyfors
190 40 ROSERSBERG
Tfn 0760-380 25

Någon annan anonym frågade något om HJÄLPARE och eftersom han inte preciserade sin fråga så kan jag bara hänvisa till bruksanvisningen i nummer 3,1982, sista uppslaget.

„VARNING: Ny Hjälpare KAN vara på gång... Vi får se!
Med vänliga halsringar, hmm, hälsningar i sommarvärmen (det är det när detta skrivs) från

THE COMPUTER PHANTOME

<837>
Kalle Lindström
Vasseur väg 35
182 35 DANDERYD

loade teksten i liniebufferen dette giver mange muligheder uden egent lig kendskab til basic-rom rutinerne.

```
LD HL, 65088
LD (HL), 82   ;R
INC HL
LD (HL), 85   ;U
INC HL
LD (HL), 78   ;N
INC HL
LD (HL), 13   ;RETURN
POP HL
JP 244
```

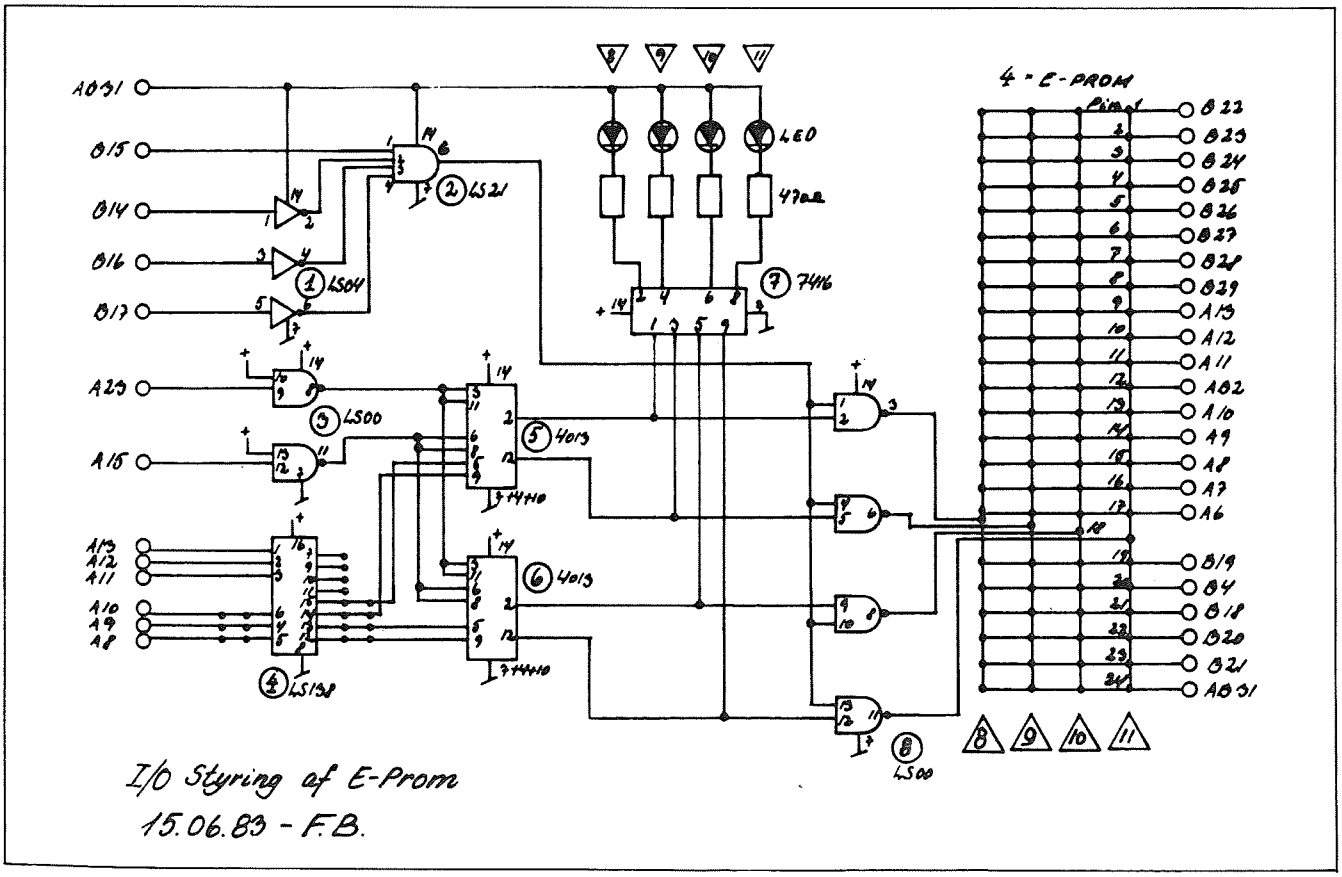
For dem der kunne taenke sig at benytte mange kort, kan jeg kun anbefale at gå igang med at fremstille en ekspansionsboks, som er ganske enkel at bygge.

Såfremt print önskes af opstillingen, kandette fremskaffes hos Spt print V/Jörn Mikkelsen, Söndergarde 76,DK4130 Viby S.-DK. og hos Jörn kan man ligeledes få fremstillet E-PROM af sine egne programmer. Framgangsmåde er blot at fremsende et kassettebånd med det önskede program, samt oplysning om du har ekstra 8K eller 16K-RAM udvidelse i din ABC80.

Skulle der vaere nogle som kunne taenke sig at gå igang, er jeg gerne behjaelplig med spörgsmål af enhver art, skriv gerne, der er ingen grund til at du tumler med nogle problemer, som måske allerede er löst.

Med venlig hilsen

Flemming Baagöe <198>
Söndergade 16, DK4130 Viby S., Danmark.



MERA SUPER(B)BASIC.

I ABC-bladet 1982:2 lämnades en användarrapport för programmeringshjälpmedlet SUPERBASIC från Nyfors Data Science, 190 40 Rosersberg. Här följer en kompletterande rapport.

SKRIV DIREKT I RAM!

Magnus Lundberg och hans kompanjon har inte legat på latsidan, utan kompletterat SUPERBASIC-paketet med flera nya delar. Den mest formidabla heter MEDIT med vilken man läser och skriver direkt i arbetsminnet. Innehållet i adresserna kan presenteras i hex- eller decimalform eller i textform. Man flyttar sig enkelt framåt och bakåt och skriver bara rakt över vad det är man vill ha.

Detta måste vara ett oumbärligt hjälpmedel för en som skall bränna PROM. Troligtvis är det litet för exklusivt för den vanlige hackern. Programmet är enormt, tar mera än 1 Kbyte, och det har börjat bli trångt bland de lediga adresserna. En del mindre viktiga äldre programdelar får stryka på foten om man vill ha plats med MEDIT.

PRAKTISKT ORDPROGRAM

Däremot har nu EDIT funnit sin form. Det är ett i princip mycket enkelt och okonstlat textediteringsprogram. Man skriver EDIT<program> så göra programmet plats för <program> på skivan. Finns det redan ett <program> skriver man bara EDIT<program>.

Sedan skriver man bara rakt fram. Programmet gör inga större konster, flyttar inte text utan bara skriver rakt på. Man kan "suga upp" gammal text eller göra plats för fler tecken i en gammal textrad. Man kan också ta bort hela rader eller skapa nya tomrader mellan de gamla. När man är färdig skriver man CTRL-Q och kan välja alternativen EN(d), BA(ckup), B(ort) som spar texten med eller utan varande av den gamla, eller går ur programmet utan att göra någonting annat.

Om skrivbufferten blir full flyttas automatiskt en del text över till skivan. Är det en gammal text som inte får plats i minnet, tas den fram bit för bit allteftersom man "tittar igenom den". Detta sista har inte haft tillfälle att uppleva, eftersom jag har 32 Kbytes och ganska små texter.

En annan värdefull finess är att om man redan har ett program i minnet, blir detta kvar när EDIT ropas in. När EDIT går ut, finns det gamla programmet där och kan köras! Läs in utskriftsprogrammet och editera ovanpå det. Skriv ut med utskriftprogrammet allteftersom dina texter blir klara. Enklare kan det nog inte bli.

Trots bristen på formateringsfinesser som är vanliga i andra skrivprogram är det lätt att få bra fason på sina texter. På bildskärmen kan man kontinuerligt se hur många rader resp kolumner texten upptar. På så sätt kan man styra storleken direkt.

Som sagt: ett ganska enkelt skrivprogram, men mycket behändigt och med fördelen att alltid finnas i "burken".

DEN SVARTA LÅDAN

"Burken" ja. I förra recensionen var en burk på väg, dvs en liten "svart låda" innehållande extra minne, som utnyttjar lediga adresser i ABC80, och alla fasta programmen i den. Man bara sätter lådan på busskontakten och skriver CMD, så kallas SUPERBASIC in. Lådan har en utgående busskontakt också, där en diskettstation kan anslutas.

LIST UTAN HALHUGGNING

När man på en vanlig ABC80 skriver LIST, visas de 23 första programraderna av inneliggande program. Men eftersom raderna ofta är längre än en rad på skärmen, kommer de översta raderna att glida upp och ut ur rutan.

Gör man LIST med SUPERBASIC inne visas endast 10 programrader. Knappast någon risk att komma utanför skärmen. LIST kan också kallas in av tecknet ASCII 126, som sitter bekvämt till nära RETURN-tangenten.

DATERA PROGRAMMEN

Men i "burken" finns fler finesser som inte syns utan ytterligare bruksanvisning. Magnus kan leverera en skiva med en lång rad mycket användbara programsnuttar som visar ut finesserna kan utnyttjas.

LIB-programmet som visar biblioteket utan att radera minnet, visar också datum då filen senast ändrades. Detta förutsätter att man har en special-DOS med viss finesser som också hör till systemet. Har man inte det kan man själv datera sina filer med FILDATE.

ORDNING PÅ TORPET

De flesta av SUPERBASIC-rutinerna kan kallas in och exekveras av vanliga BASIC-program, vilket demonstreras på skivan.

För att hålla reda på sina program, eller grupper av program, kan man (med t ex EDIT) göra en textfil där programnamnet anges tillsammans med en liten beskrivning över vad programmet gör. Sedan kan man med programmet ARKIV automatiskt söka genom denna fil och kopiera över dessa program till en annan skiva.

På liknande sätt kan man med PROG-LIST.XQT göra printerutskrifter av programmen i listan. (Det fordrar tillgång till EXEC, den i förra rapporten omnämnda autopiloten).

VIRTUELLT MINNE

SUPERBASIC-burken innehåller också en rutin som åstadkommer virtuellt minne. Man kan skriva sina subrutiner som separata program, som kan kallas in i huvudprogrammet utan att skrivas in där på nytt. Man uppnår en effekt som är välkänd från programspråket FORTH.

Men därmed har vi kommit långt in på det speciella. För att dra nytta av alla dessa finesser måsta man vara säker på att alltid ha sin SUPERBASIC inne.

DELKÖP OCH ÅNGERVECKA

Liksom förut kan man dock köpa bara de rutiner man vill ha, och man kan lagra dem i sitt RAM var man vill, lämpligen "under taket". Då behöver det varken bli för kostsamt eller för vidlyftigt.

Har man en gång prövat på att ha SUPERBASIC till hands, känner man sig handikappad utan. Sant är att ABC80 känns betydligt "större" med detta hjälpmedel.

Magnus Lundberg ger utomordentlig service och står gärna till tjänst på 0760-38025.

Sven Wickberg
<1384>

PLOTTER ANVÄNDARE

Vi har på min arbetsplats en Huston HI-plot kopplad till en ABC-80. För att kunna kommunicera med plottern med Basic kommandon finns en drivrutin skriven i maskinspråk. Problemet är att jag skulle vilja gå in och ändra i drivrutinen för att få en storlek på utskrift av tecken som är mittemellan minsta och näst minsta storleken. Scandia Metric som säljer systemet kan tyvärr inte hjälpa mig. Finns det någon som kan hjälpa mig med detta problem?

<1533>
Thomas Stridh
Uppsala

018-401628 (bost),
018-166306 (arb)

DATAKOMMUNIKATION

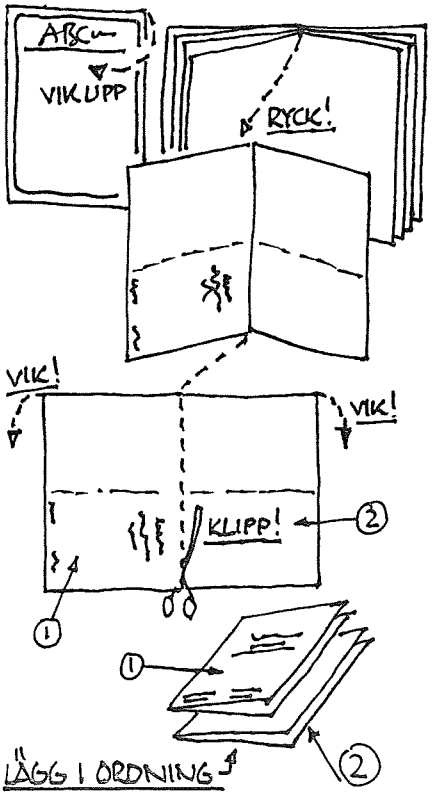
Kort och Brett om

DEC-10.

Stockholms datamaskincentral för högre utbildning och forskning, har givit ABC-bladet tillstånd att publicera skriften Kort och Brett om DEC-10 i bladet.

Skriften kommer att publiceras efter hand i mån av plats.

Den som vill ha tillgång till skriften i original kan beställa den direkt från QZ, 08-67 92 80. Tyvärr kan man inte beställa den genom ABC-klubben eller Q-Zentralen.



ABC80 SOM RAM-DATOR.

Jag har nu ett par månader provat ABC80 som RAM-maskin.

Det innbär att jag har köpt ett inbyggnadskort till tangentbordet, vilket gör att jag har tillgång till 128K extra minne. Kortet tar inte upp något minnesutrymme utan ligger gömt "bakom" basictolken.

När man har byggt in kortet (mycket lätt) så fungerar datorn som vanligt.

Vill man utnyttja RAM-maskinen kör man ett speciellt program som kopplar om datorn till denna.

Man kan också använda minnet som en RAM-disk på 488 sektorer, eller både RAM-maskin och RAM-disk på samma gång, ·ck blir RAM-disken då på 360 sektorer.

Har man en RAM-maskin är att man inte är bunden till den basictolk som finns i maskinen när man köper den. Jag har tillgång till ABC80-checksummorna 11273, 10042 och 9913 som jag använder när jag testar program för screeninggruppen.

Det går också att ändra i basictolken ·h är man riktigt duktig skriver man ihop en helt ny tolk eller varför inte ett nytt operativsystem.

Jag har också tillgång till 800 Basic och kan nu köra 800 program.

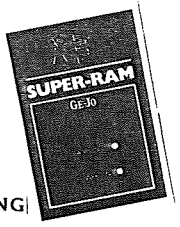
Det som jag tycker är mest utmärkande med RAM-floppyn är att allt går så fort. Har man läst in ett antal program som CHAINar varandra så märker man skillnaden direkt. En annan sak är att den är helt ljudlös.

Har man tillgång till virtuella minnesrutiner så märker man också snabbheten i förhållande till vanlig flexskiva.

När jag köpte kortet från Ge-Jo Elektronik (tel. 051-506 73) fick jag med en utförlig beskrivning som ABC-klubben har fått tillstånd att publicera.

Håll till godo.

Rune Mattsson
<904>



ANVÄNDARHANDLEDNING TILL 128K RAM-FLOPPY

När minnet har installerats börjar man först med att kontrollera att ABC80:n fortfarande fungerar normalt. Allt skall vara precis som vanligt när den kallstartas.

Därefter fortsätter man med att överföra ·· program som finns på den bifogade kassetten till en diskett genom att lägga i kassetten i bandspelaren och skriva RUN CAS: (om det ej finns floppy i systemet se beskrivning under rubrik "SYSTEM MED KASSETT").

Ett filöverföringsprogram ligger nämligen först på kassetten. Det frågar efter vilken diskett man vill lägga programmen på. När det körts finns följande program på disketten:

INIT.BAC
CASINIT.BAC
COPYLIB.BAC
CASDISK.BAC
LIB.BAC

De användes på följande sätt:

Programmet INIT används för att initiera RAM-Floppyn efter det att ABC80 kallstartats. Det börjar med att fråga om man vill kopiera BASIC-tolk m m till RAM. Om man svara ja på frågan kommer kopiering att ske och adress 0 till 32767 kommer att ligga i RAM i fortsättningen. Om man från början hade byggt ut ABC80:n till 32k RAM har man alltså nu en ren RAM-maskin!

Därefter frågar programmet efter om man vill reservera minne för maskinspråksprogram. Detta innebär att hela den kvarvarande minnesarean ej utnyttjas av RAM-Floppyn. Den reserverade arean kan alltså utnyttjas för andra ändamål. Se under rubrik "DIREKTMINNE". När programmet körts anges hur många tillgängliga sektorer det finns på RAM-Floppyn samt på vilken sida, resp adress det eventuellt reserverade minnet börjar.

Programmet COPYLIB kan användas för att kopiera filer mellan diskett till/från RAM-Floppyn och mellan RAM-Floppy/diskett till kassett. CASDISC kopierar .BAC, .BAS. ,TXT och .ABS filer från kassett till diskett/RAM-Floppy.

CASINIT används om man endast har kassett i systemet, se nedan under "SYSTEM MED KASSETT".

LIB programmet används för att visa innehållet på disketter och RAM-Floppy.

När man initierat RAM-Floppyn har den enhetsnamnet DR6:. Man kan lätt ändra numret med en POKE-sats, nämligen:

```
POKE 65033,N
```

där N står för det önskade numret. Innan man byter nummer skall alla filer vara stängda. Om man har BASICERR.SYS på RAM-Floppyn måste man också initiera detta igen med:

```
Z=CALL(0)
```

I övrigt fungerar RAM-Floppyn exakt som övriga drivrar i systemet. Man kan i vanlig ordning avända NAME, UNSAVE, KILL etc. Det går givetvis även att använda random-access. Se bruksanvisningen till ABC80.

Om man tvingas avbryta en programkörning med RESET kan i vissa fall RAM-Floppyn återiniteras utan att innehållet på denna förstörs. Om man inte hade koierat BASIC-tolk etc till ram skriver man först: OUT 135,0 (RETURN) därefter Z=CALL(0).

Om man hade kopierat över BASIC-tolken till RAM skriver man istället: OUT 207,128 (RETURN) samt därefter Z=CALL(0) (RETURN). OBSERVERA! I vissa fall kan delar av innehållet på RAM-Floppyn ha förstörts när man tryckt på RESET.

SYSTEM MED KASSETT

Har man en ABC80 med bara kassettbandspelare göra man istället på följande sätt vid initieringen:

Spola fram kassetten till programmet "CASINIT.BAC".

Utför kommandona:

```
OUT 135,0 (RETURN)
och
Z=CALL(0)
```

Sedan skriver man RUN CAS: varvid CASINIT-programmet startas. Det fungerar precis osm programmet INIT. Se ovan.

Återinitiering sker på samma sätt som om man har floppy i systemet.

DIREKTMINNE

128k-RAM:et är ett primärminne men kan ej utnytjas direkt för PEEK/POKE från BASIC-program eftersom det ligger på samma adresser som den och den kopplas ur när RAM:et kopplas in. Däremot går det alldeles utmärkt att använda RAM:et från maskinspråksprogram. Dessa måste inledas resp avslutas på ett speciellt sätt. För att koppla in önskad sida börjar man med följande instruktioner:

```
LD A,N+128 ; N=önskad sida
OUT (39),A ; Sidan inkopplas
JP STARTADRESS
```

OBSERVERA att internklockan stoppas. För att undvika problem med avbrott bör även sådana inaktiveras med instruktionen DI.

Sida resp startadress för första lediga byte i reserverat minne anges av initieringsprogrammet. Minnet ligger uppdelat i sidor om 16k byte. Programmet måste alltså hålla reda på sidorna. Varje sida omfattar adress 0 till 16383. Sammanlagt omfattar minnet 8 sidor, 0-7.

När man sedan vill återgå till BASIC måste man veta i vilket läge man befann sig från början. Tre olika möjligheter finns:

1. RAM-Floppyn ej initierad
2. RAM-Flopyn initierad, ej RAM på adress 0-32767
3. RAM-Floppyn initierad och RAM-inkopplat på adress 0-32767

I första fallet när RAM-Floppyn ej är initierad skall en nolla skrivas i port 7:

```
LD A,0
OUT (7),A.
```

I andra fallet när den är initierad men ej RAM inkopplat på adress 0-32767 skrivs en nolla i port 135.

```
LD A,0
OUT (135),A
```

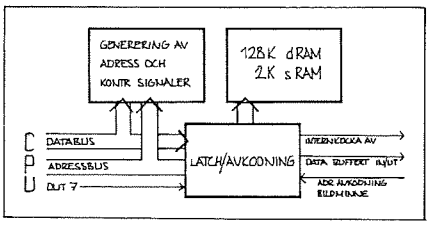
I tredje fallet där även RAM är inkopplat skrivs 128 i port 207:

```
LD A,128
OUT (207),A
```

Glöm inte att möjliggöra avbrott igen med EI. Se vidare under "TEKNISK INFORMATION".

TEKNISK INFORMATION

Här bredvid visas ett förenklat blockschema av 128k RAM:et. För att koppla



in olika sidor och övriga optioner används en latch som adresseras av utport 7 som normalt inte används i ABC80. Förutom databitarna 0-7 latchas några adressbitar. De olika bitarna används på följande sätt:

D7 -in/urkoppling av RAM:et
D6-D0 - adressbitar till RAM:et

Adressbitar som används:

A7 - kopplar in ett statiskt RAM på adress 24576-25599 (används till RAM-Floppyn)
A6 - kopplar in RAM från adress 0 till 32767, förutsatt att D7=1
A5 - stänger av internklockan
A3 - kopplar in bildminne då man har RAM inkopplat på adress 0-32767

Det statiska RAM:et har prioritet framför det dynamiska om de är inkopplade samtidigt.

FORTH.

FORTH PÅ ABC 80, DEL 2

Robert Claeson <3492>
Domarvägen 47 961 40 Boden Tel. 0921-501 87

Så var det dags igen.
Den här gången ska vi titta lite närmare på omvänd polsk notation, hur FORTH fungerar internt, hur man utökar kompilatorn, en del FORTH-79 ord, och till sist ett hjälpprogram. Vi ska i kommande avsnitt skriva fler liknande hjälpprogram, tills vi kan yta samman alla till samma meny. I samband med dessa program ska vi ävenstudera strukturerad programmering (Top/Down, Bottom /Up, JSP m.fl.).
Innan vi går vidare ska jag rätta ett fel som tyvärr insmög sig i ett av exemplen i förra avsnittet, och som jag hoppas inte har ställt till med allt för stora besvär.
I ordet LOOPA (den senare och större versionen) står det:

```
DUP 65 =
IF
   A
ELSE
```

Där det egentligen ska stå:

```
DUP 65 =
IF
   A DROP
ELSE
```

Anledningen är följande:
Vi duplicerar tecknet, som ges av KEY, innan vi testar det. Om det var ett "A", anropas FORTH-ordet A, som skriver ut en rad med 10 st. A:n. Ordet A använder INTE den kopia av tecknet som nu finns på stacken. Om vi enbart matar in A:n, kommer det att finnas 100 st. A:n på stacken sedan loopen har avslutats. Stacken har alltså "svämmat över". Eftersom stacken ligger högst upp i minnet, just under den buffert som kallas TIB (det är inmatningsbufferten, ordet TIB är en variabel som anger var i minnet bufferten TIB ska börja), kommer stacken att arbeta sig ner mot ordlistan, som i sin tur växer uppåt (i datorsammanhang växer stackar neråt, från högre addresser). Har vi otur (maximal otur i detta fall), kommer ett eller flera ord att helt eller delvis skrivas över av stacken. Det är dock inte särskilt troligt, eftersom det är åtskilliga Kbytes mellan stacken och ordlistan vid kallstart (kallstart är när man laddar in FORTH från skiva eller kassett och minnet nollställs). I stället kommer FORTH att skriva ut ett felmeddelande ("FULL STACK", eller något liknande, eller som numeriskt felmeddelande, "MSG § n", där "n" är en felkod, mer om dem senare), men det är ju inte heller särskilt trevligt.
Vi ska börja med den omvända polska notationen, eftersom det annars på vissa ställen kan vara svårt att hänga med. Det är faktiskt inte alls så svårt som det kan verka till en början. Skillnaden mellan vanlig (infix) och omvänd polsk (postfix) notation är helt enkelt en fråga om var man ska placera plustecknet (det samma gäller naturligtvis minus-, multiplikations- och divisionstecknet). Vid t.ex. en addition placerar man vid postfix notation plustecknet EFTER de tal som ska adderas, i stället för mellan dem. Det finns faktiskt också en "polsk notation" (skilj den från den omvända polska), där man placerar tecknet FÖRE de tal som ska t.ex. adderas. Denna notation används i programmeringsspråket LISP (LISt Processing). Ett exempel på skillnaden mellan infix resp. postfix notation:

A+B skrivs som A B +

Detta gör att man helt slipper parenteser för att prioritera t.ex. en addition före en division i infix notation, något som datorn uppskattar.

A*(B+C) skrivs som B C + A *
eller A B C + *

I FORTH är ett tal vanligen 2 bytes långt (d.v.s samma längd som ett heltal i BASIC). Men det finns också s.k. dubbeltal, som är 4 bytes långa. 2 bytes kallas i FORTH för en cell, och ett dubbeltal är följdaktligen 2 celler långt. Dubbeltalen har, efter en liten räkneövning (2 upphöjt till 31, minus ett), ett talområde på ca +/-2 000 000 000 (2 miljarder). De används ofta för s.k. fixed-point (kan någon hitta på en bättre svensk beteckning än "fastpunktstal" ?), vilket är ett snabbt, om än inte fullt så enkelt, sätt att använda decimaler i beräkningar (det kräver mer av programmeraren än motsvarande beräkning i flyttal). Nackdelen är framför allt ett begränsat talområde, samt, som beteckningen antyder, ett fast antal decimaler (bestäms när man skriver programmet). Fördelen är att det går flera gånger snabbare än motsvarande flyttalsberäkning. Mer om fixed point i nästa avsnitt. Tills vidare ska vi hålla oss till heltal.
Förutom de fyra räknesätten (+ - * /), har FORTH ytterligare ett antal ord med aritmetisk (beräknings-) funktion.

MOD beräknar resten av en division, d.v.s. det som blir över när en division inte går jämnt upp (om den gör det blir resten naturligtvis noll). MOD kan uttryckas med formeln R=A-INT(A/B)*B, där R är resten.

/MOD kombinerar funktionen av / och MOD. Stacken ser ut så här: N1 N2 — REM N3 där REM är resten av divisionen (REMainder). Kom ihåg att det värde som ligger överst på stacken alltid skrivs till höger i stackkommentarer.

*/ utför först en multiplikation,sedan en division. Det man vinner jämfört med att använda de vanliga * och / är en högre precision, då mellanresultatet emllan * och / lagras i 32 bitar. Detta ord är faktiskt mer användbart än det kan verka, och används bl.a. till procenträkning och avrundning.

*/MOD kombinerar funktionen av */ och MOD. Skillnaden mot */ är att även resten av den avslutande divisionen lämnas. Så här ser stacken ut: N1 N2 N3 -- REM N4

U* multiplicerar 2 otecknade tal, och lämnar ett otecknat dubbeltal.

U/ dividerar ett otecknat dubbeltal med ett enkelt otecknat tal, och lämnar resten (otecknad), och kvoten (även den otecknad), båda i form av enkeltal.

M* fungerar som U*, med den skillnaden att alla tal ska vara tecknade (med plus- eller minustecken angivet).

M/ fungerar som U/, men behandlar tecknade tal.

M/MOD dividerar ett otecknat dubbeltal med ett otecknat enkeltal, och lämnar resten (otecknat enkeltal) och kvoten (otecknat dubbeltal).

D+ adderar 2 tecknade dubbeltal till ett tecknat dubbeltal.

Det finns även ett antal ord som inte direkt utför någon beräkning, men är väldigt användbara.

+- tar 2 tal från stacken. Om det översta talet är negativt, lämnas det undre i negativ form, annars i positivt. Enklare uttryckt: det undre talet tar sitt tecken från det övre.

0< lämnar en sann flagga (skilt från noll) om talet på stacken är negativt, annars en falsk flagga (en nolla).

0= lämnar en sann flagga om talet på stacken är noll, annars en falsk.

NOT fungerar faktiskt precis som 0=, men används för att invertera funktionen av en jämförelse. Sekvensen = NOT motsvarar <>.

0> lämnar en sann flagga om talet på stacken är större än noll, annars en falsk flagga.

< N1 N2 -- FLAG Lämnar en sann flagga om N1 är mindre än N2, annars en falsk.

= Lämnar en sann flagga om N1 är lika med N2, annars en falsk.

> Lämnar en sann flagga om N1 är större än N2, annars en falsk.

1+ ökar talet på stacken med ett.

2+ ökar talet på stacken med två.

D+- fungerar som +-, men det undre talet är dubbelt.

ABS lämnar absolutvärdet (talet utan tecken) av talet på stacken.

DABS som ABS, men opererar på dubbeltal.

MINUS byter tecken på ett tal, d.v.s att ett negativt tal blir positivt, och ett positivt blir negativt.

DMINUS byter tecken på ett dubbeltal.

MAX N1 N2 -- N1 eller N2 Lämnar det största av N1 och N2 på stacken.

MIN Lämnar det minsta av N1 och N2 på stacken.

S->D Omvandlar ett enkeltal till ett dubbeltal, och behåller talets tecken. Den omvända operationen, d.v.s. från dubbeltal till enkeltal, sker med DROP.

De vanliga bitlogiska AND, OR och XOR finns också. Om du redan vet vad dessa ord gör, kan du hoppa över följande stycke.

För att förklara AND, OR och XOR underlättar det mycket om man arbetar med binära tal. Antag att vi har följande binära tal:

```
1 0 0 1 0 1 1 1
```

Om vi gör AND 10001011 på detta tal, ser det ut så här:

```
   1 0 0 1 0 1 1 1
AND 1 0 0 0 1 0 1 1
===================
   1 0 0 0 0 0 1 1
```

AND ger en etta som resultat ENDAST om båda bitarna är ettor. OR ger en etta som resultat om en eller båda bitarna är ettor. XOR ger en etta som resultat endast om en av bitarna är en etta. Är båda bitarna ettor, blir resultatet en nolla. De bitlogiska funktionerna gör det alltså möjligt att släcka (s.k. maskning), tända och invertera enskilda bitar.

Antag nu att vi vill göra en rutin som skriver ut textrader om 64 tecken på skrivaren. Raderna ska tas från tangentbordet. Rutinen ska skriva ut 65 rader per sida, och sedan göra ett sidbyte, d v s. hoppa över skarvarna i papperet. För att hålla reda på när det är dags att byta sida, kan man använda en variabel, som man ökar för varje rad som skrivs ut.

Efter varje utskriven rad testar man variabeln, och byter sida och nollställer variabeln om 65 rader är utskrivna. Detta test kan man göra dels med =, dels med MOD.
Ett exempel, med användande av MOD:

```
: PRINT-65-LINES   ( -- )
    PR-ON   0 §CR !        (NOLLSTÄLL RÄKNAREN )
    BEGIN
      TIB É  64 EXPECT     ( TA IN EN RAD TILL TIB )
      0 IN !  0 WORD       ( LÄGG DEN I HERE )
      HERE CÉ  0=          ( KOLLA OM ANTALET TECKEN ÄR NOLL )
      IF     1             ( I SÅ FALL, MARKERA "KLART" )
      ELSE   HERE COUNT TYPE CR ( SKRIV UT RADEN )
             §CR É  65 MOD 0= ( KOLLA OM DET ÄR DAGS FÖR SIDBYTE)
             IF      72 §CR É DO
                        CR
                     LOOP
                     0 §CR ! ( NOLLSTÄLL RÄKNAREN )
             THEN    0 ( MARKERA "EN RAD TILL" )
      THEN
    UNTIL    PR-OFF  ;
```

§CR är en variabel som automatiskt ökas med 1 för varje CR. THEN är ett annat (modernare) ord för ENDIF.

TIB är en variabel som anger platsen för inmatningsbufferten. EXPECT tar in en rad från tangentbordet, med max antal tecken angivet. WORD söker igenom raden efter ett visst tecken. När tecknet noll (NULL) påträffas, stoppas sökningen automatiskt. Teckensträngen läggs fr.o.m. address HERE, som anger första lediga byten i ordlistan, med antalet tecken i första byten. WORD använder variabeln IN, som anger från vilket tecken från TIB som sökningen ska börja. Därför måste den nollställas först. WORD ändrar också värdet i IN. COUNT tar adressen till strängens första byte, och lägger adressen + 1, och värdet i den minnesaddressen (vanligen antalet tecken) på stacken. Dessa värden används av ordet TYPE, som skriver ut raden.

Ett annat exempel för MOD är när man vill omvandla en löpande teckenposition till rad/kolumn-koordinater.

```
R§ É ANTAL-TECKEN-PER-RAD /MOD
```

R§ är en variabel som är avsedd för just en teckenposition, t.ex. i en editor. Kvoten anger raden, och resten vilken position på raden som teckenpositionen svarar mot. ANTAL-TECKEN-PER-RAD är en konstant.

Om du vill multiplicera ett tal med 1.5 i heltalsarimetik, måste du gå omvägar. 1.5 är ju 3/2, så man skule kunna multiplicera talet med 3, och sedan dividera med 2. Om man gör så på för höga tal (över ca 10000), får man fel resultat p.g.a. heltalens talområde (+-32767). Då är ordet */ väldigt användbart, eftersom det använder 32 bitar i mellanresultatet.

Ett annat exempel:
Antag att du vill dela ett tal med 1.2. Resultatet ska vara avrundat. Skriv så här:

```
TAL  100  12  */  5 + 10 /
```

Metoden går ut på att multiplicera upp talet tills det finns plats för decimalerna, addera fem innan sista decimalen tas bort, för att få en korrekt avrundning vid nästa division, där den sista decimalen tas bort. Härmed kom vi in lite på fixed-point, vilket inte var meningen.

De övriga orden torde det inte vara något problem att förstå.

Innan vi tar oss en titt "under huven", ska vi gå igenom en del förekommande ord.

NUMBER konverterar en talsträng till ett dubbeltal. Som argument (indata) krävs addressen till strängen, med längden i första byten.

INTERPRET är den yttre interpretatorn, som tolkar det som matas in, antingen via tangentbordet, eller via skärmar (de s.k. screensen).

EXECUTE är den inre interpretatorn, som kör programmen. Den är, till skillnad från INTERPRET, skriven i maskinspråk, och därför kommer den inte att behandlas närmare. Det enda argument som krävs är addressen till kod-fältets address i ett ord.

BLK är en variabel som anger vilket block inmatningen ska tas från. Om BLK är noll, tas texten från tangentbordet. I princip ska BLK vara noll utom vid LOAD.

IMMEDIATEs är ord som utförs under kompileringen, för att vidta särskilda åtgärder. Alla "programflödesord" är sådana. De lägger in orden BRANCH och 0BRANCH i koden, vilka är FORTHs grundläggande hoppinstruktioner. Det är med hjälp av dessa vi ska konstruera CASE-satsen. ." är ett annat exempel på en IMMEDIATE.

IMMEDIATE gör det senast definierade ordet till en IMMEDIATE.

/COMPILE/ gör det möjligt att använda IMMEDIATEs som vanliga ord, vilket vanligtvis utnyttjas i andra IMMEDIATEs.

LB sätter FORTH i interpreteringsmode, vilket innebär att allt som matas in utförs direkt.

RB sätter FORTH i kompileringsmode, vilket innebär att allt som amtas in läggs i ordlistan. RB ger oss en möjlighet att mata in rader längre än 80 rader från tangentbordet. Sedan 80 (eller däromkring, raden får inte ta slut mitt i ett ord) tecken matats in, trycker man på RETURN, och inleder nästa rad med RB.

STATE är en variabel som anger om FORTH befinner sig i interpreterings- eller kompilerings-mode. Ett värde skilt från noll anger kompilering.

QUERY är definierad som TIB É 80 EXPECT 0 IN !. Vi har just laddat in FORTH. Ordet COLD är det första som utförs. Det tömmer bl.a. skärmbuffertarna, initierar vissa variabler (bl.a. STATE och BASE), skriver ut välkomstmeddelandet (via ordet ABORT), och anropar ordet QUIT, som är FORTHs huvudloop. QUIT nollställer variabeln BLK och returstacken, och väntar sedan på något att skicka vidare till INTERPRET. Sedan skrivs "ok" ut. Om man avslutar ett program med ordet QUIT, kommer "ok" inte att skrivas ut på skärmen.

Så här ser QUIT ut:

```
.: QUIT   ( -- )
.   0 BLK !  /COMPILE/ LB
.   BEGIN
.      CR  RP!
.      QUERY  INTERPRET
.      STATE É  NOT
.      IF     ." ok"  THEN
.   AGAIN  ;
```

Forts i nästa nr.

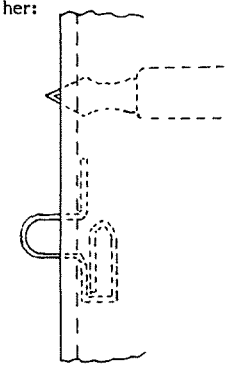
DIV:

Tips om oppbevaring av ABC-bladet.

Med alt det verdifulle innhold som ABC-bladet bringer, er det god grunn til å ta vare på det. Det er iallfall min oppfatning. Men for å få samlet alle heftene på ett sted, trenger man system. Jag anvender brevpermer (også kallt ringorder o.l.), men det fordres at heftene har hull for ringmekanismen.

Jeg fant snart ut at det var ikke aktuelt å stanse hull i ABC-heftene.
1: Det var for mange sider på en gang.
2: Marginen mot midten var for smal så jeg risikerte å stanse bort verdifulltekst.

Problemet blev löst på det sett som illustreres her:



I en bok/papir-forretning kjöper man sig en "radir-penn" (stålpenn med V-vormet skarp spiss). Utrustet med en slik innsatt i en solid penneholder, stikker man fra midtsidene av heftet to (eller fire) hull i ryggen på heftet, med avstand lik avstanden mellom ringen.

Hvert av hullen settes en "binders" (papirclips) som er böyd slikt som skissen viser. Man får da passende lökker til å tre inn på ringene i permen. Man kan selvsagt kutte av den ekstra "kröllen" med en avbiter, men jeg fant den god å ha til å holde fast sedler (anmerkninger til teksten i vedkommende hefte) slik at de ikke faller ut av permen settes inn i eller tas ut av hyllen.

Med hilsen
Halvard Torgersen <3342>

Lokalavdelning i Skåne

Mats Knuts <746> och Jan Friberg <429> arbetar med planer för att kalla till ett möte i Malmö för att bilda en lokalavdelning i Skåne. I skrivande stund är inga detaljer klara, men intresserade kan vända sig till:

<746>
Mats Knuts
Serenadgatan 24
214 72 Malmö
040-874 98

<429>
Jan Friberg
Bofinksvägen 14
040-41 31 13

Hälsar
<1789>
Bo Kullmar
Skåning i exil

Nu har **CP/M-handboken** kommit.
Det är en översättning av den engelska **"The CP/M Handbook"** - en storsäljare i USA. Den svenska översättningen används redan nu av flera utbildningsinstitut. Boken innehåller en steg för steg beskrivning av alla CP/M kommandon, så att även nybörjaren får det lätt för sig.

CP/M har blivit en industristandard för operativsystem som kan användas tillsammans med mikrodatorer. Många av de stora tillverkarnas datorer har eller kan förses med CP/M. T.ex PET, APPLE, TRS-80, ABC80, IBM PC, Dynabyte, Altos o s v.

Alla CP/M-användare har nytta av de tabeller och sammanställningar över kommandon, parametrar, felmeddelanden och omvandlingstabeller som finns i boken.

CP/M-handboken med MP/M av Rodnay Zaks
Översättning: Siv Grinde
ISBN: 91-86200-02-X
PAGINA FÖRLAGS AB
Distribution: TRIM MARKETING AB
Box 49035, 100 28 Stockholm
Tel. 08-54 00 10

==========

BASIC i praktiken

Nu har boken **BASIC i praktiken för ABC-datorer** och andra kommit.Det är en översättning (engelsk titel: Fifty BASIC Exercises, orginalets titel: Le BASIC par le Pratique) och har bearbetats för att passa svenska förhållanden.

Genom en mängd praktiska övningar lär sig läsaren hantera problem inom matematik, affärsberäkningar, operationsanalys, spel och många andra ämnen.

Alla övningar är utförligt förklarade med problemformuleringar, lösningar med flödesscheman, program med detaljerade kommentarer och körexempel.

Författaren J P Lamotier har lång erfarenhet som lärare i BASIC och Fortran. Han betonar den praktiska delen av datorprogrammering.

Översättningen har gjorts av Jerker Thorell. Cirkapriset anges till 148 Skr inkl moms.

BASIC i praktiken av J P Lamotier
Översättning: Jerker Thorell
ISBN:91-86200-03-8
PAGINA FÖRLAGS AB
Distribution: TRIM MARKETING AB
Box 49035, 100 28 STOCKHOLM
Tfn 08-54 00 10

SÄLJES.

Fabrikstillverkad JOY-STICK, modifierad till ABC80. 8 riktningar och avtryckare. Drivrutin, en rad Basic, medföljer.
Pris 190:- inkl.porto.

Erik Hänel,<1452>
Älvdansvägen 16
436 00 ASKIM

031-28 56 76.

SÄLJES

Akustiskt modem till ABC80 komplett med T80PRT på kassett och kabel. pris: 1950:-

Magnus Matts <899>
Engelbrektsgatan 8
783 00 SÄTER

0225-516 76

KÖPES.

Floppydisk till ABC80

Åke Brisvall <2800>
Kalmar

0480-19267

EFTERLYSNING AV STULET MATERIAL

2 st ABC80 med monitor och kassettspelare märkta på höljet med "KATEDRALSKOLAN" och med spak på baksidan för omvänd video (2-polig 2-vägs omkopplare).

Upplysningar kan lämnas till:
Martin Berg <215>
Döbelsgatan 32 C
752 37 Uppsala

Reset

Har du kanske satt i modem-kabeln eller printer-kabeln eller på något annat sätt kommit åt RESET knappen när du absolut inte vill ha den intryckt.

Jag har i alla fall lyckats med detta konststycke. Men nöden är uppfinningarnas moder.

Jag har ingen större lust att medelst PEEK leta mig fram till radslutet och räkna bytes däremellan. Jag satte mig därför ner och tänkte (det var ett ganska stort program jag hade förlorat).

BASIC-tolkens kompileringsrutin borde kunna göra jobbet åt mig.

Jag skrev POKE <BOFA>,0

Detta lurar datorn att tro att det finns ett program i minnet. Efter RESET står det I där.

Sedan LIST <FILNAMN>

BASIC-tolken behöver ju bara kompilera .BAS-filer. Och jag hoppades också att den på samma gång skulle räkna ut den riktiga checksumman. Detta gjorde den också.

Sedan skrev jag LOAD <FILNAMN> och därefter SAVE <FILNAMN>. Detta för att spara den kompilerade versionen. Och det var med glädje jag kunde skriva RUN - och programmet fungerade.

Lars-Börje Cid
Bondegatan 24 B
641 45 Katrineholm

Forts från föreg nr.

FORTH.

FORTH PÅ ABC 80, DEL 2

INTERPRET är betydligt mer komplicerad. Först söker det efter ordet TIB i ordlistan. Finns det inte där, är det antagligen ett tal. Är det inte ett tal heller, är det fel, och följdaktligen ges ett felmeddelande.

INTERPRET ska dessutom kunna skilja mellan interpretering (tolkning) och kompilering.

```
: INTERPRET   ( -- )
   BEGIN
      -FIND
      IF    STATE É <
            IF     CFA ,
            ELSE   EXECUTE
            THEN
      ELSE  HERE NUMBER
            DPL É 1+
            IF     /COMPILE/ DLITERAL
            ELSE   DROP  /COMPILE/ LITERAL
            THEN
      THEN  ?STACK
   AGAIN    ;
```

-FIND letar i ordlistan efter det ord som finns i TIB, med startposition tagen från IN. Om ordet finns där, returneras parameter-fält-addressen (parameterfältet är den area i ett ord där de ord det består av finns), längden på namnet, och en sann flagga, annars en falsk. Ordet CFA konverterar parameter-fält-addressen till kod-fält-addressen, vilket är den address där en pekare finns, vilken pekar på den maskinkodsrutln som ska utföras när ett ord utförs. Denna rutin är olika för kolon-definitioner, variabler och konstanter.

Ordet , (komma), lagrar ett 16-bitars värde i ordlistan, i det här fallet en address, i ett ords parameterfält. DPL är en variabel som sätts av NUMBER. Den anger det eventuella antalet siffror till höger om decimalpunkten. En decimalpunkt anger att talet ska tolkas som ett dubbeltal. Om DPL är -1, finns det ingen decimalpunkt, varför talet konverteras till ett enkeltal. ?STACK kontrollerar att stacken inte har svämmat över. Om den är överfull eller "undertom", ges ett felmeddelande. Både NUMBER och ?STACK ger själva felmeddelanden.

Observera att DLITERAL och LITERAL föregås av ordet /COMPILE/, eftersom de är IMMEDIATEs. /COMPILE/ letar igenom ordlistan efter det ord som följer /COMPILE/ (återigen används ordet -FIND). Om det inte finns där, ges ett felmeddelande. I nat fall kompileras ordet (addressen till kodfältet läggs efter HERE, med hjälp av ordet "komma", samt att ordlistepekaren (den heter DP för Dictionary Pointer, och är en variabel) ökas med 2 bytes.

```
: /COMPILE/   ( -- )
   -FIND  0=  0 ?ERROR
   DROP  CFA ,  ;
```

?ERROR skriver ut ett felmeddelande (i det här fallet nummer 0) om det näst-översta värdet på stacken är sant. Felmeddelandena tas med offset (startposition) från skärm 4 (om variabeln WARNING är noll, ges ett numeriskt felmeddelande). Om felkoden överstiger 21 (största radnumret på en skärm), sätts rad 0 på nästa skärm (nr. 5) till rad 22 o.s.v.

LITERAL kompilerar ett 16-bitars värde (DLITERAL kompilerar ett 32- bitars värde). En numerisk konstant kompileras genom att ordet LIT kompileras, sedan läggs ett 16-bitars värde ner efter LIT. När koden sedan utförs, ökar LIT programräknaren (det register som håller reda på var i minnet den kompilerade programkod som körs finns, programräknaren kallas IP för Interpretative Pointer),samtidigt som de 2 bytes som LIT stegar förbi läggs på stacken. DLITERAL använder LITERAL 2 gånger genom satsen /COMPILE/ LITERAL.

```
: LITERAL   ( N -- )
   STATE É
   IF    COMPILE LIT  ,
   THEN  ;

   IMMEDIATE

: DLITERAL   ( D -- ) STATE É
   IF    SWAP  /COMPILE/ LITERAL  /COMPILE/ LITERAL
   THEN  ;

   IMMEDIATE
```

COMPILE kompilerar det ord som ligger efter COMPILE i den kompilerade programkoden. Det här var ju krångligt. Men det fungerar i alla fall så här. Antag att du definierar ett IMMEDIATE ord, som ska lägga ner värdet 7 i ordlistan (från HERE). Ordet heter SEVEN, och definieras på följande sätt:

```
: SEVEN   ( -- )
   COMPILE LIT
   7 ,  ;

   IMMEDIATE
```

SEVEN får ingå i ett ord som heter .SEVEN, d.v.s. ett ord som ska skriva ut talet 7 när det exekveras (utförs). Punkten i .SEVEN talar om att något skrivs ut. .SEVEN är inte en IMMEDIATE.

```
: .SEVEN   ( -- )
   SEVEN .  ;
```

Naturligtvis skulle det gå lika bra att använda en konstant i stället för SEVEN, men för exemplets skull så...

Seven utförs när .SEVEN kompileras, eftersom det är ett IMMEDIATE ord. SEVEN lägger först ner ordet LIT i definitionen av .SEVEN, sedan talet 7. Addressen till ordet SEVEN kompileras inte. Om man vill använda ett IMMEDIATE ord i en egen definition, måste man skriva /COMPILE/ ccc, där ccc är ett IMMEDIATE ord.

Beräkningsstacken är den stack som vanligtvis kallas för "stacken". Beräkningsstackens startposition bestäms av variabeln S0. Beräkningsstackens nuvarande addressfås med SPÉ, som returnerar addressen till stackens översta värde (egentligen det understa, eftersom alla stackar växer neråt) som den var innan SPÉ utfördes.

Om man har fyllt stacken, och vill nollställa den, kan man, i stället för att skriva DROP DROP DROP ..., skriva SP!, vilket sätter stackpekaren (naturligtvis finns det en sådan, för att hålla reda på var i minnet man kan lägga upp på resp. hämta värden från stacken) till S0. Som stackpekare används mikroprocessorn Z80As stackregister SP. Med hjälp av S0 och SPÉ kan man definiera sina egna stack-mainpulerings-ord.

Returstacken används för att lagra hoppaddresser. När ett ord anropas (och det sker ju nästan jämt), måste man naturligtvis hålla reda på varifrån anropet skedde. När ett FORTH-ord anropas, läggs därför IP på returstacken (IP är i Z80 FORTH mikroprocessor -register BC), därefter laddas IP med addressen till det första ordet i det anropade ordet. Detta nya ord anropar i sin tur andra ord, varför IP återigen läggs på returstacken o.s.v. Returstackens startposition anges av variabeln R0. RP! använder det värde som finns lagrat i R0 för att nollställa returstacken. Returstacken har också en andra uppgift. Man kan nämligen spara undan värden tillfälligt på returstacken. Man får bara inte glömma att hämta tillbaka värdena innan man gör retur (när definitionen är slut), eftersom IP då kommer att laddas med fel värde från returstacken. >R lägger det tal som finns överst på beräkningsstacken på returstacken. R kopierar det värde som finns på returstacken till beräkningsstacken. Värdet på returstacken finns alltså fortfarande kvar. R> hämtar ett värde från returstacken, och lägger det på beräkningsstacken.

Var försiktig med dessa ord. Det kan krascha FORTH-systemet om de används fel.

Det finns ytterligare ett sätt att avbryta en BEGIN -- AGAIN -loop än det som visades i förra avsnittet, med hjälp av de ord som manipulerar returstacken. Du kan stoppa den med R> DROP.

Definierande ord är sådana som skapar ett nytt ord. Exempel på sådana ord är : (kolon), VARIABLE och CONSTANT. När ett sådant ord utföres, skapas ett s.k. headerfält. Det innehåller, i nämd ordning, namnets längd i en byte, namnet i varierande antal bytes, en pekare till nästa ord i ordlistan, en kodpekare till den assemblerkod som ska utföras när ett ord anropas (denna kod är olika för kolon, VARIABLE och CONSTANT), samt ett fält med pekare till de ord detta ord består av. Dessa fält kallas namnfält, länkfält, kodfält och parameterfält. Sedan headerfältet har skapats (det består av namnfält och länkfält), läggs kodpekaren ner, den s.k. run-time-koden, för aktuellt ord ner vid HERE (HERE är definierad som DP É). VARIABLE och CONSTANT sparar sedan ett 16-bitars tal, medan kolon sätter FORTH i kompileringsläge, vilket gör att INTERPRET kommer att spara addressen (till kodfältet) för de ord som räknas upp i ordlistan, fram till semikolon, vilket sätter FORTH i interpreteringsläge.

Man kan göra egna definierande ord, t.ex. för olika datatyper. Ett definierande ord har alltså dels en egen kod, dels en kod som utförs när ordet som skapades med detta ord utförs. Det är denna senare kod som kallas "run-time-kod".

Den egna koden inleds med ordet <BUILDS, och run-time-koden inleds sedan med DOES>. Run-time-koden skapas här i FORTH-kod, men kan också skapas i assemblerkod, men det tar vi i kommande avsnitt. Ett lämpligt exempel är ordet 2VARIABLE, som används för att lagra ett 32-bitars tal. Det kan definieras som 0 VARIABLE 2 ALLOT (ALLOT flyttar fram ordlistepeka-

ren det antal bytes som anges före), men för exemplets skull ska vi definiera allt själva.

```
: 2VARIABLE   ( D -- )
   <BUILDS    , ,
    DOES>     ;
```

Eftersom addressen till första byten efter run-time-kod-pekaren i det definierade ordet läggs på stacken innan 2VARIABLEs run-time-kod utförs (den som står efter DOES>), anger vi ingen run-time kod. Efter run-time pekaren ligger ju det värde vi ville nå, och addressen dit får vi s.a.s. på köpet. Lite värre blir det med 2CONSTANT.

```
: 2CONSTANT   ( D — )
   <BUILDS    , ,
    DOES>     2@   ;
```

Här ska vi ju också hämta det värde som konstanten motsvarar. På liknande sätt kan man göra arrays ("indexerade variabler"), flerdimensionella matriser, strängar m.m.

Kompilerande ord är ord som gör något under kompileringen. Dessa är IMMEDIATEs. Exempel på kompilerande ord är SEVEN (!), IF, ELSE, WHILE, REPEAT, UNTIL och AGAIN.

```
: IF    ( -- )
   COMPILE 0BRANCH
   HERE  0 ,  2 ;   IMMEDIATE
```

CASE-satsen är skapad av Charles E. Eaker (nej, han är INTE medlem i ABC-klubben). Det CASE-satsen ska göra är att jämföra ett argument, som ges CASE-ordet, med olika jämförelsevärden, ett åt gången, som ges ett annat villkorsord. Om argumentet överensstäm- mer med testvärdet, ska det som står efter villkors-ordet utföras, annars ska jämförelsen fortsätta till CASE-satsens slut (här kallat ENDCASE). Alla ord i CASE-satsen måste vara IMMEDIATEs, av kompilerande typ.

```
: CASE   ( N -- )
   ?COMP  CSP @ !CSP  4 ;   IMMEDIATE
```

CASE-satsen inleds med ordet CASE. Det testar om FORTH befinner sig i kompileringsmode, och avbryter om så inte är fallet. CSP är en variabel som används främst av kompilerande ord. Den är avsedd att spara stackens position i. Sedan kan man testa om stackens aktuella position motsvarar den som finns i CSP, och ge ett felmeddelande om så inte är fallet (med ordet ?CSP). Här hämtar vi CSPs värde (det kan ju ha satts av t.ex. en BEGIN -- WHILE -- REPEAT-sats), och lagrar det på stacken (stacken används inte till något speciellt just nu - vi kompilerar ju). Sedan sparas stackens position i CSP för senare felkontroll av ordet ENDCASE. Slutligen lämnas en fyra på stacken för typkontroll (se även IF-satsen, den lämnar en 2:a, vilken används av ELSE och THEN för att kontrollera att man inte skriver IF -- WHILE -- UNTIL, eller något annat galet. Den villkorliga satsen heter här OF.

```
: OF   ( N1 N2  --  N1 at no match, otherwise nothing )
   4 ?PAIRS                ( KOLLA OM DET ÄR RÄTT TYP )
   COMPILE OVER  COMPILE = ( LÄGG NER TEST-SEKVENSEN  )
   COMPILE 0BRANCH         ( VILLKORLIGT HOPP         )
   HERE  0 ,               ( SPARA ADRESSEN           )
                           ( TILL HOPPADRESSEN        )
                           ( OCH RESERVERA PLATS      )
                           ( FÖR ADRESS               )
   COMPILE DROP  5 ;   IMMEDIATE
```

Först kontrolleras att OF är rätt ord i detta fall (med ordet ?PAIRS), sedan kompileras ordet OVER, = resp. 0BRANCH in. HERE lämnas på stacken för senare bruk av de avslutande orden, varefter address 0 anges som hoppaddress tills vidare. Det är för att kunna lägga in rätt address här senare, som HERE läggs på stacken. 0BRANCH gör ett hopp om en falsk flagga ligger på stacken när koden utförs. Sedan läggs ett DROP in för att ta bort CASEs argument om jämförelsen stämde.

OF ska naturligtvis ha ett avslutande ord. Det heter, mycket logiskt, ENDOF.

```
: ENDOF   ( -- )
   5 ?PAIRS
   COMPILE BRANCH  HERE  0 ,
   SWAP  2 /COMPILE/ THEN  4 ;  IMMEDIATE
```

Som vanligt sker först en typtest. Därefter läggs ett ovillkorligt hopp in. Ett SWAP utförs för att hämta addressen där OFs villkorliga hoppaddress ska läggas. En tvåa läggs på stacken (det är den som THEN testar mot), varefter THEN utföres, med allt vad det innebär av ytterligare kompilering. Till sist läggs en fyra på stacken att användas av OF eller ENDCASE som testvärde vid kompileringen. Nu är det bara ENDCASE kvar.

```
: ENDCASE   ( -- )
   4 ?PAIRS
   COMPILE DROP
   BEGIN
      SPE  CSP @  = NOT
      WHILE
         2 /COMPILE/ THEN REPEAT
   CSP !  ;   IMMEDIATE
```

Först testas, som vanligt, att ENDCASE är rätt ord i sammanhanget. Därefter läggs ett DROP ner, för att ta bort eventuellt argument som kan ligga kvar, om inget av fallen stämde. En WHILE-loop utförs tills alla jämförelser har försetts med hoppaddresser. SPE hämtar stackpekarens aktuella värde, därefter hämtas stackens position som den var innan CASE-satsens början. Om dessa två värden inte överensstämmer, utförs THEN, som lägger ner hoppadresser alldeles gratis. Slutligen återställs CSP med det värde som lagrades på stacken. CASE-satsen används på följande sätt:

Antag att du vill att den som använder programmet bara ska få mata in A, B och C. Om någon annan tangent trycks ner, ska en 'bell'-signal ges. De (giltliga) tecken som matas in ska inte sparas, bara skrivas ut.

```
: GET-ONLY-ABC   ( -- )
   5 0 DO   ( 5 TECKEN )
      KEY  CASE
         65 OF  65 EMIT  ENDOF
         66 OF  66 EMIT  ENDOF
         67 OF  67 EMIT  ENDOF

         ( OTHERWISE )

         7 EMIT
      ENDCASE
   LOOP  ;
```

Observera att om en definition inte får plats på en skärm, kan du göra tre saker.

1) "Packa" programmet, d.v.s. skriv bara ett mellanslag mellan varje ord, utan att byta rad vid DO-LOOP, IF-ELSE-THEN, BEGINUNTIL o.s.v.
2) Dela upp definitionen på flera skärmar med ordet -->. Eftersom --> är en IMMEDIATE fungerar det utmärkt. Att låta en definition sträcka över flera skärmar brukar betecknas som "dålig FORTH-stil" (en FORTH-definition ska vara rätt kort), men jag anser att det är betydligt bättre att göra så än att packa programmet.
3) Dela upp definitionen i flera, små definitioner. Det går dock inte alltid att göra så (exempelvis vid en CASE-- sats med många jämförelser). I sådana fall är alternativ 2 det bästa.

En IMMEDIATE-definition som är praktisk att ha när man skriver program är ASCII. ASCII kompilerar ascii-värdet av det tecken som står efter ASCII. Ex:

ASCII A ger samma resultat som att skriva 65, vilket är ascii-värdet för A. ASCII ligger på systemskärmarna, som levererades i filen SCREEN.TXT. Skriv 6 LOAD för att ladda in ASCII.

På samma skärm ligger WHERE, som används för felsökning vid laddning från skärmar. När ett fel har uppstått, kan man skriva WHERE. Då skrivs skärmens nummer ut, samtidigt som den rad där felet uppstod skrivs ut. Det felaktiga ordet markeras med ett delete-tecken, "fylld ruta", på raden under.

```
: ASCII   ( -- )
   BL WORD  HERE 1+ C@
   /COMPILE/ LITERAL  ;   IMMEDIATE
```

Stränghanteringen i FORTH sker på samma sätt som i assembler. För som inte har någon erfarenhet av assemblerprogrammering, kan jag nämna, att det innebär full frihet att skjuta in tecken, dra ihop, flytta, skarva o.s.v.

En strängvariabel (man lägger inte gärna strängar på stacken, bara dess address) kan man definiera med ordet CVARIABLE (fåö Charachter VARIABLE), med strängens maximala längd och ordet ALLOT efter.

```
0 CVARIABLE STRÄNG  28 ALLOT
```

reserverar plats för en strängvariabel med namnet STRÄNG, som inte får innehålla fler tecken än 28 (i första byten ska strängens längd ligga), med antalet tecken satt till noll från början. Med hjälp av <BUILDS och DOES> kan man skapa egna datatyper, t.ex. den datatyp som kallas CHAR i Pascal. Orden CMOVE och <CMOVE används för att flytta minnesareor. Med

dessa kan ordet ASSIGN skapas, vilken tilldelar en sträng en annan strängs innehåll.

```
: ASSIGN   ( FROM TO  --  )
   OVER CE  1+  CMOVE  ;
```

ASSIGN flyttar hela strängen vid address FROM till address TO. Hela strängen flyttas, inklusive längdbyten.

Skillnaden mellan CMOVE och <CMOVE är att CMOVE flyttar lägsta byten först och går mot högre addresser, medan <CMOVE gör tvärt om. Om du ska flytta isär tecken i en sträng ska du använda <CMOVE, mdan CMOVE ska användas om du ska dra ihop.

Strukturerad programmering är ett område som det har yttrats många dumheter om. Här kommer några till.

I många böcker och manualer om FORTH står det att FORTH är lämpligt att användas för att konstruera program enligt Top/Down metoden. Top/Down är inte en metod. Det är ett sätt att skriva och konstruera program. JSP (Jackson Structured Programming) är en metod. Top/Down kan användas för att konstruera program i JSP. Bottom/Up går lika bra att använda. Top/Down till på följande sätt.Man börjar med att definiera huvudordet. Det ska beskriva vad programmet gör. För ett applikationsprogram, t.ex. ett databasprogram, kan detta se ut så här:

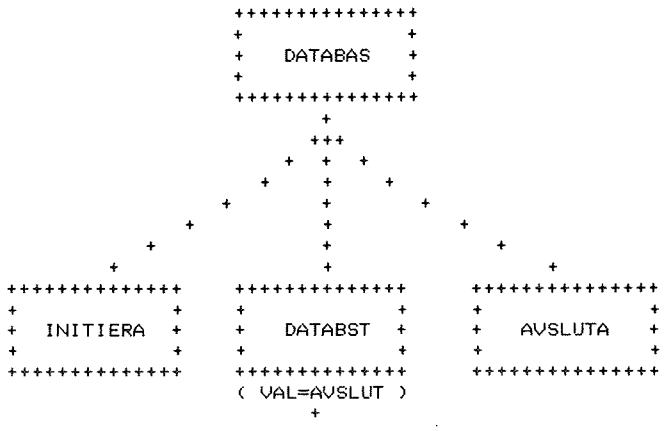
```
: DATABAS     ( MAIN )
   INITIERA
   BEGIN
      MENY
      UTFÖR
   UNTIL
   AVSLUTA   ;
```

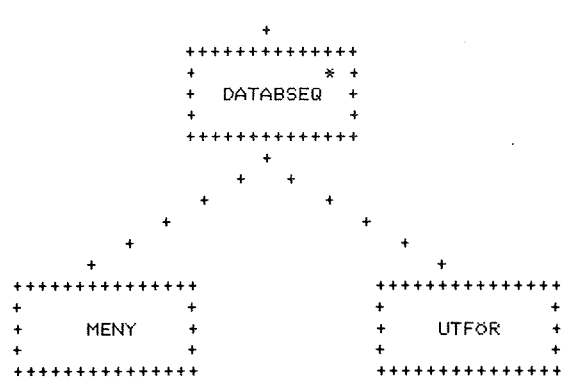
INITIERA ska initiera databasen. Det består i sin tur av andra ord, som på en lägre nivå beskriver hur denna initiering ska gå till. Var och ett av dessa ord består av ytterligare ord på en ännu lägre nivå, som beskriver en avgränsad process, t.ex. att fråga efter behörighet till databasen, o.s.v. På samma sätt är MENY, UTFÖR och AVSLUTA nerbrutna i ord på lägre nivå.

Bottom/Up går till på det omvända sättet, d.v.s. att man börjar med att definiera orden på den lägsta nivån.

I ett strukturerat program får ord på olika nivåer inte anropa varandra. Det kan vara svårt att hålla reda på om man enbart använder Top/Down eller Bottom/Up. Anledningen till att man håller så strikta regler är att det ska vara enkelt att följa programmet, ändra (utan att det blir grötigt), felsöka och göra tillägg. JSP är en metod. Det finns bestämda regler för hur man får göra i JSP. Det finns det inte i Top/Down eller Bottom /Up, eftersom de inte är metoder att utveckla program, utan sätt att mata in programmet i datorn. Men, som sagt var, Top/Down och JSP går utmärkt att kombinera.

I FORTH använder man en trädstruktur (det kallas så eftersom strukturdiagrammen, med lite fantasi, ser ut som ett träd med den högst upp). Högst upp, det som kallas "roten", ska programmets eller delrutinens namn stå. För att se vad denna är av för typ, måste man titta på närmast lägre nivå. Det finns bara 3 olika typer av strukturer, nämligen sekvensen (utförs efter varandra), iterationen (upprepning, t.ex. en loop), och selektionen (ett val, t.ex. IF -- ELSE -- THEN). Med dessa tre typer kan man konstruera ALLA program. Det bevisade två herrar vid namn Bohm och Böhm det minnesvärda året 1966. En sekvens markeras inte alls, en iteration markeras med en stjärna eller en asterisk, och en selektion markeras med en ring. Ovanstående exempel, DATABAS, kan uttryckas på följande sätt i ett JSP-strukturdiagram.





Man ska inte skriva större diagram än vad som ryms på en A4-sida (jag brukar använda liggande A4or). De ord som är utan vidare förgreningar (de s.k. löven, i detta fall MENY och UTFÖR), kan man beskriva närmare på särskilda sidor (naturligtvis i JSP-strukturdiagram). DATABST ovan utför ingenting. Det är bara en stomme som måste stå där. Eftersom DATABAS är en sekvens, kan det inte finnas en iteration mitt i. Man måste använda "rena" strukturer. Sedan alla diagram är färdiga, ska man inte sätta igång och skriva FORTH-kod på en gång. Vi har nämligen ett viktigt steg kvar, den s.k. schematiska logiken, vilket är ett slags mellan-programmeringsspråk som till sin struktur liknar Pascal. Observera att löven har semikolon efter, som procedur-anrop i Pascal.

Exemplet ovan skulle uttryckas så här:

```
DATABAS seq
   INITIERA;
   DATABST iter until VAL=AVSLUT
      MENY;
      UTFÖR;
   DATABST end
   AVSLUTA;
DATABAS end
```

Nu kan man skriva den FORTH-snutt som visades tidigare. JSP visar sina starka sidor framför allt vid mer komplicerade program som det var ett tag sen man skrev, eller som man kanske inte alls skrivit själv. Har man dessutom tillgång till strukturdiagrammen och den schematiska logiken, kan man mycket snabbt sätta sig in i programmet. De höga underhållskostnaderna för program inom framför allt mini- och stor-dator området gör att hundratals programmerare undervisas i JSP varje år i Norden.

Hur långa ska definitionerna vara i FORTH ? Det beror på vad de utför, men helst bör de inte vara längre än några få rader. I BASIC använder man subrutiner för att ersätta en sekvens som förekommer på flera ställen. I FORTH sätter man en avgränsad operation i en definition. Har man skrivit JSP-diagrammen bra, är det lagom att låta varje JSP-ruta vara en definition.

Innan vi tar i tu med utility-modulen, ska vi göra ett mycket, mycket enkelt spel. Det brukar kallas "rysk roulette", och går ut på att tävla mot ett slumptal. Det gäller att klara sig så länge som möjligt. Har man otur, kommer slumptalet upp i ett för högt tal, och sedan är man förlorad.

```
0 VARIABLE RÄKNARE

: SLUTA   ( AVBRYT SPELET )
   1  ;

: TRYCK   ( SE OM HJÄLTEN BLEV TRÄFFAD ELLER INTE )
   CLOCK CE  215 )
   IF    CR  ." Det är över nu..."  SLUTA
   ELSE  CR  ." -KLICK-"  0   1 RÄKNARE +!
   THEN  CR  ;

: AVBRYT   ( HJÄLTEN VILL AVBRYTA )
   CR ." FEGIS!" CR CR  SLUTA  ;

: SPELA   ( SPELA TILLS HJÄLTEN AVBRYTER ELLER BLIR TRÄFFAD )
   0 RÄKNARE !   ( NOLLSTÄLL RÄKNAREN )
   BEGIN
      KEY
      CASE
         13  OF   TRYCK   ENDOF
         65  OF   AVBRYT  ENDOF
      ENDCASE
   UNTIL  ;

: J/N   ( GE 1 FÖR J, 0 FÖR N )
   BEGIN
      KEY  DUP  223 AND  DUP DUP
      74 =  SWAP  78 =  OR
      IF    74 =  SWAP EMIT 1
      ELSE  2DROP  0
      THEN
   UNTIL    ;
```

```
: SPELA-IGEN?   ( SKRIV UT HUR MANGA VAR HJÄLTE KLARADE, OCH FRAGA
                  OM HAN/HON VILL FORTSÄTTA                      )
   CR  ." DU KLARADE "  RÄKNARE ?  ." GÅNGER."  CR CR
   ." VILL DU SPELA IGEN ? "  J/N  NOT  CR  CR  ;

: REGLER   ( SKRIV UT VILKA REGLER SOM GÄLLER )
   PAGE  7 SPACES
   ." ***** RYSK ROULETTE *****"  CR CR CR
   ." DIN UPPGIFT ÄR ATT SPELA RYSK ROULETTE"  CR
   ." MOT DEN TVELIVADE TSAREN"  CR
   ." DU KAN KOMMA UNDAN GENOM ATT SKRIVA"  CR
   ." BOKSTAVEN 'A' I STÄLLET FÖR ATT TRYCKA AV."  CR
   ." DU KOMMER DÅ UNDAN MED SKAMMEN,"  CR
   ." OCH BLIR HÅNAD AV DEN TVELIVADE TSAREN." CR  CR
   ." TRYCK PÅ NÅGON TANGENT FÖR ATT BÖRJA : "  KEY DROP
   CR CR  ;

: HEJDA   ( VÄLKOMNA TILL FORTSATT SPELANDE VID TILLFÄLLE )
   CR ." VÄLDIGT TRÅKIGT." CR
   ." DEN TVELIVADE TSAREN HÄLSAR DIG VÄLKOMMEN" CR
   ." VID SENARE TILLFÄLLE FÖR FORTSATTA FESTLIGHETER."
   CR CR  ;

: ROULETTE   ( HUVUDLOOP )
   REGLER
   BEGIN
      SPELA
      SPELA-IGEN?
   UNTIL
   HEJDA
   QUIT   ;
```

Du får själv klura ut hur ovanstående program fungerar. Det är konstruerat med JSP.

Utility- (hjälp-) modulen däremot, ska vi gå igenom lite närmare. Du har väl, liksom jag, irriterats över att behöva skriva CLEAR innan en ny skärm ska tas i bruk. Programmet i sej är inte särskilt avancerat, bara några få ord långt, men desto mer användbart. Dessutom kan delar av det vara användbart i andra sammanhang.

```
0 VARIABLE FRÅN
0 VARIABLE TILL

: #IN   ( TA IN ETT ENKELTAL FRÅN TANGENTBORDET )
   TIB @  SWAP  EXPECT
   BL WORD
   HERE NUMBER DROP ;

: FRÅN? ( TA REDA PÅ STARTSKÄRM )
   ." FRÅN VILKEN SKÄRM SKA RADERINGEN SKE ? "  10 #IN
   DUP DUP  1 < SWAP 1000 >  OR
   IF    QUIT   THEN  FRÅN !  ;

: TILL? ( TILL VILKEN SKÄRM )
   ." TILL VILKEN SKÄRM ? "  10 #IN
   DUP DUP FRÅN @  <  SWAP  1000 > OR
   IF  QUIT  THEN  TILL !  ;

: MELLAN-VILKA?   ( MELLAN VILKA SKÄRMAR SKA DET RENSAS ? )
   PAGE   7 SPACES
   ." ***** SUDDA SKÄRMAR *****"  CR CR CR
   FRÅN?  CR
   TILL?  CR  ;

: SUDDA-SKÄRMAR ( SUDDA SKÄRMARNA )
   TILL @  1+  FRÅN @  DO
      I BLOCK   768  BL FILL  I UPDATE
   LOOP
   FLUSH   ;

: SUDDA  ( HUVUD-DEFINITIONEN, ELLER MAIN SOM MAN SÄGER )
   MELLAN-VILKA?
   SUDDA-SKÄRMAR
   QUIT  ;
```

§IN tar in ett enkeltal från tangentbordet. Innan anropet sker, ska det maximala antalet siffror som får matas in anges. FRÅN? tar in ett värde, och kontrollerar rimligheten i svaret. TILL? gör samma sak, men kontrollerar att detta värde inte är lägre än från-värdet.

SUDDA-SKÄRMAR är det ord som sköter om själva suddningen. BLOCK ger addressen till den skärm som anges som parameter. Om skärmen inte finns i minnet läses den in. FILL suddar sedan 768 bytes från denna address med blanktecken (mellanslag). Sedan markeras att denna skärm har ändrats (uppdaterats), så att den sedan skrivs tillbaks till skivan när den buffert den upptar i minnet behövs för andra skärmar. Till sist sköter FLUSH om att skriva tillbaka de återstående skärmarna. SUDDA är det ord man skriver för att starta programmet. QUIT i denna definition gör att inget "ok" skrivs ut efteråt.

De numeriska felkoderna är som följer:

```
 1 - EMPTY STACK   ( stacken är tom )
 2 - DICTIONARY FULL
 3 - HAS INCORRECT ADDRESS MODE
 4 - ISN'T UNIQUE
 6 - DISC RANGE ?
 7 - FULL STACK
 8 - DISC ERROR ! (READ ERROR ?)
 9 - DISC DOOR OPEN/NO DISC
10 - DISC IS WRITE PROTECTED
11 - DISC ISN'T FORMATTED
12 - NO DISC !
```



```
15 - Här ligger en kommentar som används vid
     listning på printer
22 - COMPILATION ONLY, USE IN DEFINITION
23 - EXECUTION ONLY
24 - CONDITIONALS NOT PAIRED
25 - DEFINITION NOT FINISHED
26 - IN PROTECTED DICTIONARY
27 - USE ONLY WHEN LOADING
28 - OFF CURRENT EDITING SCREEN
29 - DECLARE VOCABULARY
```

FORTH på ABC 800 med DOS 6-1x för ABC 830 har tagits fram av Mats Knuts. Vi som använder UFD-DOS och/eller ABC 832 får vänta tills en sådan version tagits fram. UFD-DOS är inte särskilt lämplig att använda tillsammans med den nuvarande lagringsmetoden. Direktfiler, eller, ännu hellre, textfiler vore här det riktiga. Textfilerna har dessutom den fördelen att programmen utan problem kan överföras via tråd, eller skrivas via vilken texteditor som helst. Man får dessutom fritt format på programmen och är inte begränsad till skärm-formatet, något som har betydelse när man ändrar i programmen. Man behöver inte förlora det virtuella minne som skärmarna utgör. Man kan simulera detta via direktfiler som kontrolleras via de vanliga orden FLUSH, BLOCK, UPDATE m.fl. ord. Men detta är utökningar för framtiden.

Till nästa avsnitt, GO FORTH!

Robert Claeson <3492>

Synpunkter på fig-FORTH

Robert Claesson <3492> kommer under vintern att utveckla en ny FORTH enligt FORTH83-standard. Den som har upptäckt buggar i fig-FORTH,saknar funktioner eller har andra synpunkter kan ta kontakt med ▷

<3492>
Robert Claeson
Dommarvägen 47
961 40 Boden
0921-501 87

Systemvariabler i ABC800.

ABC800 datorer använder sig av en area högst upp i minnet för att lagra data som systemet använder sig av. Denna area omfattar minnesadresserna 64768 - 65535. Om man inte har UFD-doset är dock 65024-65279 ledigt för t ex egna assemblerrutiner. En omfattande beskrivning av adresserna finns i boken Bit för Bit. Här skall jag endast nämna en del adresser och lite om hur de kan användas. B betecknar en byte (ett tecken) och W betecknar ett ord (två tecken).

65280
Denna adress erhålles med SYS(10). Enligt Carl-Axel Petterson sätts adressen till 1 om CTRL-C har tryckts.

65286 W BOFA
Pekare till programmets början, dvs egentligen programinformationsblocket. Det är lika med värdet som fås med SYS(11).

65288 W EOFA
Pekare till sista byten i BASIC-programmet.

65290 W HEAP
Pekare till första lediga byte i minnet.

65292 W BOTTOM
Pekare till första byte av minnet som används till program m m. Om man höjer denna pekare kan man få minnesutrymme ledigt för t ex assemblerprogram.

65294 W TOP
Pekare till sista byten av minnet som används till program m m. Om man sänker denna pekare kan man minnesutrymme bli ledigt för t ex assemblerprogram.

65309 B FL
Diverse flaggor
Bit 2 = 1 då är blanktecken signifikant (EXTEND mode)
Bit 1 = 1 indikerar EXTEND mode
Bit 0 = 1 indikerar INTEGER mode

65310 B PREC
Flyttalsprecision. Innehåller antalet bytes lagringsutrymme som de två varianterna av flyttal upptar. Vid SINGLE precision är värdet 4 i 65310. Vid DOUBLE precision är värdet 8 i 65310.

65311 B DIGITS
Innehåller antal siffror som skrivs ut med PRINT, standardvärde är 6/16 beroende på flyttalsprecision. Ändras med BASIC-instruktionen DIGITS. Observera att max-värdet är 7/17 siffror.

65314 B DEFLOW
OPTION BASE. Innehåller lägsta vektor-index. Ändras med BASIC-instruktionen OPTION BASE.

65315 B INT
Interrupt-byte
Bit 3 = 1 anger SINGLE STEP mode
Bit 2 = 1 anger DIREKT mode, = 0 anger programkörning
Bit 1 = 1 anger TRACE mode
Bit 0 = 1 anger att CTRL-C har skrivits in

65318 B PRSTAT
Programstatus
Bit 7 Ny errorhantering används
Bit 6 HR-grafik används
Bit 5 Program högt i minnet
Bit 4 Flyttal finns i COMMON
Bit 3 Flyttal är alloklerade
Bit 2 Dubbel precision
Bit 1 Listskydd, att nollställa denna byte på fil gör normalt inte att man kan låsa upp listskydd
Bit 0 Fixed-up

65326 W VARBAS
Pekar på variabellistan

65332 B USERCS
Senast valda I/O kort.

65333 B TRCLU
Logisk enhet vid TRACE, innehåller nummret på den fil som TRACE § skriver på.

65348 B ERRCOD
ERRCODE, obs ERRCODE nollställs ej vid RUN.

65362 B
Aktuell kolumn för bildskärmen

65363 B
Aktuell rad för bildskärmen

65364 B WID
Aktuell radbredd för bildskärmen

65403 W DEVTBA
Pekare till länkad enhetslista. Denna lista används om man vill länka in egna enheter.

65405 W STMTBA
Pekare till länkad instruktionslista. Denna lista kan användas om man vill lägga in egna instruktioner.

65407 W FNKTBA
Pekare till länkad funktionslista. Liknar ovanstående men avser funktioner i stället för instruktioner.

65413 W CTRLCPTR
Pekare till CTRL-C. På adressen (normalt 65315) som denna systemvariabel pekare på sätts bit 0 till 1 när CTRL-C skrivs in.
Stäng av CTRL-C-funktionen = POKE 65413,0
Återställ CTRL-C-fuktionen = POKE 65413,35,255 (65315).

65415 CMDUNSA
Hopp till UNSAVE-rutinen. Hit hoppar datorn när den får kommandot UNSAVE. Genom att lägga in RET (Poke 65415,210) här kan man stänga av kommanot UNSAVE. Man kan också lägga in ett hopp via en egen rutin, som då utföres när kommandot UNSAVE ges.

65421 HRCLR1
Släck högupplösningsgrafik, anropas bl a av LIST-kommandot. Hopp till denna rutin innebär att systemet gör FGCTL 0. Funktionen kan stängas av genom att skriva RET i 65421 (POKE 65421,201). För att sätta på funktionen igen måste man återställa de ursprungliga värdena. Detta kan ske genom att först ta reda på vad det finns för adress där genom Pekare= PEEK2(65421) och sedan återskriva adressen med POKE 65421, Pekare, SWAP%(Pekare).

65424 CONSI
Hopp till rutinen som läser ett tecken (GET) och lägger resultatet i register A.

65430 USERTRAC
Hopp till en subrutin som skriver ut radnummer vid TRACE. Genom att i stället göra hopp till en egen TRACE-rutin så kan man använda en annan TRACE-rutin än den som finns normalt. Se Bit för Bit.

65433 B CASSPEED
Hastighet vid skrivning på CAS:. Normalt är innehållet 40 vilket avser 2400 bps. 0 är lika med 700 bps, d v s den hastighet som ABC80 normalt använder. Carl-Axel Pettersson rekomendar en test med t ex 25.

65460 W
Interrupt-adress för tangentbordet. Används vid interrupt från tangetbordet (när någon trycker på en tanget).

65494 W
Interruptvektor för klocka. Ett hopp hit sker var 1/100 sekund. Hoppet sker till en rutin som ökar innehållet i 65525 med 1.

Om man låter hoppet gå till en egen rutin kan man t ex låta datorn kontinurligt skriva ut klockan på bildskärmen. Se programmen SYSTIME.800 eller TIME.806 som utnyttjar detta. Programmen finns på klubbens monitor.

65506 B

Tangentbordsflagga. Denna byte innehåller 128 om en tangent är nertryckt. Erhålles också med SYS(5).
Nollställes med GET, INPUT och INPUT LINE. Kan givitvis också nollställas med POKE 65506,0

65507 B

Tangentbortsbuffert. Innehåller ASCII-väret för senast nedtryckta tanget. Jmf SYS(6).

65519 B YEAR
År för Time$ (0-99)

65520 B MNTH
Månad för Time$ (1-12)

65521 B DAY
Dag för Time$ (0-31, 0 = stopp klocka)

65522 B HOUR
Timme för Time$ (0-23)

65523 B MIN
Minut för Time$ (0-59)

65524 B SEC
Sekund för Time$ (0-59)

65525 B TIC
Räknare för Time$ (0-93)

65526 B

UFD-RESET. Om innehållet är lika med 165 nollställes ej ett ev aktivt UFD vid RESET. Observera att 65526 alltid nollställes vid RESET, så man måste skriva dit 165 efter varje RESET som sker för att få funktionen kvar.

65527 W

UFD-offset. Adress till ev aktivt UFD-bibliotek. UFD är ett användardefinerat bibliotek som kan användas om man har UFD-DOS eller ABCNET-DOS.

65528 B

UFD-drive. Drivenummer för ett aktivt UFD.

65529 B

RAM-floppy. Finns en RAM-rutin innehåller adressen 165 vill man att RAM-floppyn skall formateras (nollställas) skrivs 0 här och formateringen sker vid nästa RESET om drivrutinen finns på tillgänglig vid RESET.

PS Adresserna 65526-65528 är enbart aktuella för datorer utrustade med UFD-doset.

Carl-Axel Petersson <2229>

Bo Kullmar <1789>

ABC STOCKHOLM

Vidängsvägen 1
161 33 BROMMA Postgiro 40 42 52-9

Ordf Stig Löfgren <872>
sekr Bo Hjulström <557>,
kassör Kjell Järbin <700>.

Flera medlemmar i stockholmskretsen

Det under hösten utsända informations- och värvningsbrevet till ABC-klubbens stockholmssektion har gett en glädjande stort medlemsökning. 1983-11-09 var antalet 314. Alla som bor i Stor-stockholm eller inom bekvämt reseavstånd från Alvik är välkomna att bli medlemmar. Alla får plats i klubben. De fördelar du får via lokalklubben är i korthet följande:

Informationsmöten med föredrag och demonstrationer i olika ämnen. Som lokalmedlem kommer du in gratis, annars kostar det 10:- per kväll.

Öppet hus på klubblokalen varje tisdag kl 1830 - ca 2130. Där finns möjlighet att diskutera med andra medlemmar, köra program på klubbens datorer, köpa visst förbrukningsmateriel, läsa klubbens böcker och tidskrifter etc. Så småningom kommer vi att kunna köra ABC-klubbens monitor från en lokal dator till hjälp för alla som inte har telefonmodem.

I klubblokalen finns också en anslagstavla för klubbaktiviteter, köp och försäljning etc. Dessutom bjuder klubben på te.

Kommande stockholmsaktiviteter

Onsdag 25 jan 1984
Information om JSP (Jackson Structured Programming).

Onsdag 7 mars
Årsmöte i ABC Stockholm.

Där ej annat sägs äger alla informationsmöten rum kl 1900 i Vidängssalen, Vidängsvägen 1.

Aktiviteter 1984 under förberedelse
Februari
Studiebesök. (program ej fastställt)

Onsdag 11 mars
Informationsmöte. (program ej fastställt)

Kursverksamhet

Vi planerar att arrangera datororienterad utbildning i form av studiecirklar. Den första torde bli assemblerprogrammering. Problemet för närvarande är att vi saknar cirkelledare, och vi skulle mycket gärna vilja få kontakt med en person med administrativ och pedagogiskt intresse för organisation av kursverksamheten. Lärare och tekniska experter finns, men det är svårt att få tiden att räcka till även för den organisatoriska överbyggnaden.

Är det någon som har ideer till ämnen för informationsmöten och studiebesök eller som känner för att hjälpa till med kursverksamheten, går det bra att ringa till någon i styrelsen.

Tekniska projekt inom sektionen

Bosse <557> har en konkret ide till en hårdvarutillsats som ger cursor-styrning och funktions-tangenter via ABC 80:s V24-kontakt. En "standard" joystick (t ex Atari) kan användas som styrdon. Beskrivning planeras i ABC-bladet.

En klubbmedlem har tagit fram en billig 32 K minnesexpansion för ABC 80, som kan byggas för runt 200:-. Beskrivning kommer troligen i ABC-bladet eller i en ABC-rapport.

Att rädda flexskivor

På informationsmötet den 19 okt berättade Rune Sagnell <1602> om hur data är organiserade på 5 1/4 " flexskivor och demonstrerade en nytt program "FILTEST" med vars hjälp man kan läsa och skriva direkt i skivan. Programmet är skrivet av Kai Liljebladh <806> och är uppbyggt på ett sådant sätt att det är lätt att både lära sig och förstå skivans inre hemligheter. Det finns också ett antal funktioner för analys av felaktigheter, "reparation" av bitmap, biblioteket etc.

Ett ofta förekommande inslag på informationsmötena är och demonstration av program från ABC-kassetterna. Den 19 okt visades programmet i högprecisionsmatematik och hur man kan använda detta för bl a statistiska beräkningar. Det var ett verkligt intressant exempel på hur en mikrodator kan utnyttjas inom ett område som inte kan hanteras manuellt och där man tidigare fått ta till stordatorer.

ABC-Stockholm på Tekniska Museet eller hur QZ kan användas

Lördagen den 12 november kom ett meddelande på QZ från Tekniska Museets Ungdomsförening, som på fars dag - d v s dagen efter meddelandet - skulle ordna en s k aktivitetsdag med praktiska demonstrationer av olika slag. I år ville man gärna ha hjälp med att visa datorer och vad de kan användas till och vände sig av den anledningen till ABC-klubben.

Det var flera som läste meddelandet och gjorde sig beredda att agera. Förberedelsetiden var ju inte så lång, men en grupp formerades sig snabbt med Göran <2771>, Allan <3435> samt Kjell <700> med sonen Håkan <3673>. De gjordes ett antal snabba ryck och med klubbens och privata datorer ställde man upp på museet och tänkte visa administrativa program, grafik, frågesport och kommunikation för att ge en mångsidig belysning av datorn som praktiskt hjälpmedel.

Museets unga besökare visade sig dock vara inne på en något annan linje, och det tog inte många sekunder innan de anstormande entusiasterna hade slagit CTRL C och laddat in 'Starfighter' och satt full fart på rymdkampen!

Det som därutöver intresserade var främst dator-kommunikation med QZ och hela övningen var ett bra exempel på nyttan av denna form av samband.

I ABC-Stockholms bibliotek finns f n följande litteratur:

Böcker

ABC om användardokumentation
ABC om BASIC
ABC om mätsystem
ABC om programmering och dokumentation
Avancerad programmering på ABC80
Basic Computer Games
Basic Handboken
Bygg ut ABC80 med DataBoard 4680
Mikrodatorns ABC
The Basic Handbook
Your First Computer

Bruksanvisningar och manualer

ABC80 bruksanvisning
Datadisk 82-84 d:o
Datadisk 88 d:o
Flexskiveenhet ABC d:o
Servicemanual ABC80
Servicemanual ABC800
Z80-Z80A CPU Technical Manual

Ovanstående litteratur förvaras i klubblokalen och är tillgänglig under tisdagskvällarna. Ingen hemlåning tillåts.

Kort och Brett om

DEC-10.

Här kommer nu ytterligare 16 sidor av skriften Kort och Brett om DEC-10.

Skriften kommer att publiceras efter hand i mån om plats.

Den som vill ha tillgång till skriften i original kan beställa den direkt från QZ, 08-67 92 80. Tyvärr kan man inte beställa den genom ABC-klubben eller Q-Zentralen.

CP/M SuperCalc II

som kalkylprogram

SuperCalc är ett av de många kalkylprogram som finns att tillgå under CP/M på marknaden. Andra exempel är Multiplan, Knowledge Man och Visicalc (det senare kan endast köras under andra operativsystem på t ex Apple och TRS-80). För de nya 16-bitarsdatorerna finns t ex Lotus 1, 2, 3.

Programvarorna är alla exempel på de nya [illegible] generationsspråken eller superhögnivåspråken, vilket man nu vill kalla dem. Detta innebär användarvänliga programmeringsspråk, så som jag förenklat vill uttrycka det. Användaren märker inte att han i själva verket programmerar datorn när han eller hon använder kalkylprogrammet.

[illegible]dskärmen kan ses som ett stort rutat [illegible]betsblad, indelat i rader och kolumner. Man brukar även kalla det för ett elektroniskt skissblock. SuperCalc är uppbyggt med 63 kolumner i sidled och 254 rader i höjdled, där du i varje ruta som bildas kan skriva in text, siffror eller matematiska formler, som refererar till innehåll i andra rutor i matrisen. En kalkyl byggs upp så, att programmet hämtar innehållet ur andra rutor i matrisen utför någon ny beräkning och skriver svaret i den ruta där formeln är inmatad. Data blir på detta sätt relaterade till varandra och en ändring i en ruta påverkar alla övriga rutor i matrisen, som på något sätt är relaterade (kopplade till) den ruta där ändringen gjordes.

Man kan lätt ändra, stryka, spara eller skriva ut kalkylen. Rader eller kolumner kan läggas till eller tas bort i efterhand och i varje situation, när man arbetar med ett kommando, kan man få fram hjälptexter på svenska som visar hur man skall gå vidare med kommandot.

[illegible]g skall inte gå in för att förklara kraftfullheten i dessa typer av programvaror i detalj, det skulle kräva ett eget nummer av ABC-bladet, utan nöja mig med att visa ett exempel på en liten kalkyltillämpning och hur resultatet av ett antal kalkyler kan se ut presenterade av diagramprogrammet [illegible]RAPH.

Kalkylen avser installation av värmepumpar i en bostadsrättsförening innehållande 5 hus och 120 lägenheter för att återvinna energi ur den frisk luft, ca 20 grader varm, som evakueras ut via fläktsystemet.

Konsultföretaget som projekterade anläggningen, hade räknat ut besparingen, den årliga driftsreduktionen, det första året. Jag upplevde att jag ville se hur kalkylen tog sig ut under en 15-årsperiod eftersom investeringen skulle genomföras med hjälp av statliga energilån, som skulle förräntas och amorteras under en 15-årsperiod. Genom att ta hänsyn till alla kända fakta och göra en uppräkning med en inflationsfaktor (= årlig oljeprisökning) borde man få ett bättre underlag för att fatta beslut om investeringen.

I kalkylen intogs även en faktor för att kunna variera effektiviteten på anläggningen d v s 0 till 10 % avvikelse från förutsatt funktion på värmepumpar och elförbrukning.

Vid större avvikelse än 10 % träder leverantörens garanti in och anläggningen skall åtgärdas utan kostnad för beställaren men upp till 10 % felfunktion får man tåla i kalkylen. Eftersom staten aviserat att en snabbare upptrappning av den garanterade räntan kan komma att ske på energilån behövde även uppräkningsfaktorn för garanterad ränta kunna varieras.

Överst i kalkylen räknas den årliga driftsreduktionen det första året fram, där värmebesparing och energibehov redovisas i kWh. Oljepriset i kr/kubikmeter omvandlas till fjärrvärmetaxa i kr/kWh varefter årlig besparing i kr redovisas.

I den nedre tabellen finns inflationsfaktor, nu 10 % årlig inflation förutsatt. Årlig reduktion räknas fram och en korrigerad reduktion sedan hänsyn tagits till kostnader för service av anläggningen. Amorteringen på lån (annuitetslån med olika annuiteter på bottenlån och energilån) framräknas jämte garanterad ränta, varefter man ser den totala kapitalkostnaden per år. Längst ut till höger räknas årligt netto fram samt ackumulerad besparing under 15-årsperioden.

Att nu variera de olika faktorerna leder till att datorn producerar en oacceptabel mängd datalistor där det inte blir lätt att få överblick över vad man egentligen håller på med.

Grafikprogrammet kan lättare redovisa kurvor over det ackumulerade nettot, när inflationen sätts till t ex 8 eller 10 % och när upptrappningsfaktorn för garanterad ränta t ex fördubblas. Tre diagram redovisas här med 0 %, 5 % respektive 10 % förutsatt felfunktion på anläggningen. Man ser i det sista diagrammet att 8 % inflation och fördubblad garanterad ränta blir klart förlustbringande.

Genom möjligheterna att på två sekunder räkna om kalkylen kan man lätt se vilka parametrar kalkylen är känslig för. Störst betydelse får det oljepris jag har när jag startar kalkylen, årlig prisstegring är nödvändig för att en kalkyl överhuvudtaget skall gå runt, brytpunkten ligger vid ca 4 % årlig inflation o s v.

Hur gjorde man förr för att göra enkla investeringskalkyler ?

Ja, jag för min del avstod från att räkna på mer än några få alternativ med vanlig fickräknare. Utan fickräknare kan jag dessutom inte alls räkna.

Leder detta till någon förbättring ?

Man kan hävda att den som påstår något nytt, har bevisbördan för att det leder till någonting bättre. Jag har här avstått från bevisning genom att formlerna till kalkylen inte redovisats, de hade annars lika enkelt kunnat fås ut på papper på skrivaren.

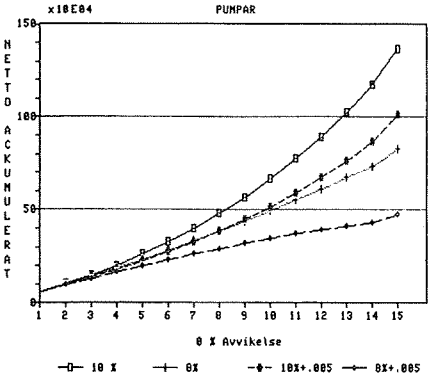
Frågan överlåter jag till dig, käre läsare att besvara.

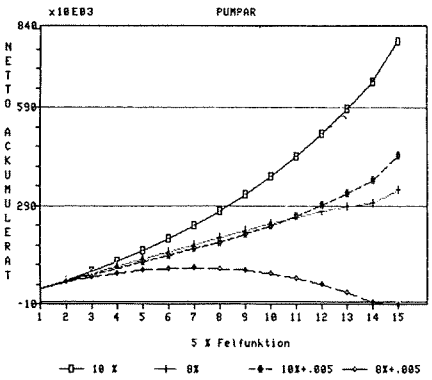
Odd Rolander

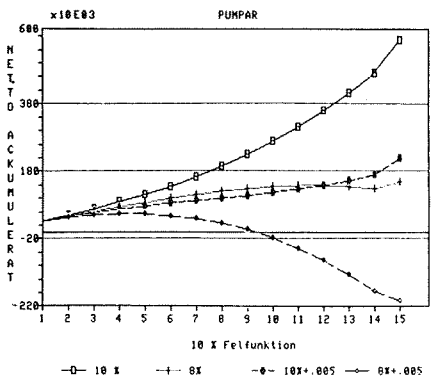
VÄRMEPUMPAR

Utrustn	Faktor	kWh	Korr kWh	Oljepris	Energipris	Kr/år	Driftred
VÄRMEPUMP	1	1,070,000.00	1,070,000.00	1,760.00	.22	235,400.00	
KULVERT	1	70,000.00	70,000.00	1,760.00	.22	15,400.00	
EL VP	1	370,000.00	370,000.00		.29	107,300.00	
EL EPATR	1	5,000.00	5,000.00		.29	1,450.00	
TARIFF		9,450.00				9,450.00	132,600.00

År	Faktor	Årlig red	Service	Korr red	Amort	Uppr fakt	Gar ränta	Kapitalkostn	Netto	Ackumulerat
År 1	1	132,600.00	.00	132,600.00	35,815.00	.03	40,050.00	73,865.00	58,735.00	58,735.00
År 2	1.1	145,860.00	20,000.00	125,860.00	36,210.95	.0325	43,387.50	81,598.45	44,261.55	102,996.55
År 3	1.1	160,446.00	22,000.00	138,446.00	43,178.37	.035	46,725.00	89,903.37	48,542.63	151,539.18
År 4	1.1	176,490.60	24,200.00	152,290.60	48,791.66	.0375	50,062.50	98,854.06	53,436.54	204,975.71
År 5	1.1	194,139.66	26,620.00	167,519.66	55,134.47	.04	53,400.00	108,534.47	58,985.19	263,960.91
År 6	1.1	213,553.63	29,282.00	184,271.63	62,301.95	.0425	56,737.50	119,039.45	65,232.18	329,193.09
År 7	1.1	234,908.99	32,210.20	202,698.79	70,401.20	.045	60,075.00	130,476.20	72,222.59	401,415.68
År 8	1.1	258,399.89	35,431.22	222,968.67	79,553.35	.0475	63,412.50	142,965.85	80,002.81	481,418.44
År 9	1.1	284,239.86	38,974.34	245,265.53	89,895.29	.05	66,750.00	156,645.29	88,620.24	570,038.74
År 10	1.1	312,663.86	42,871.78	269,792.09	101,561.68	.0525	70,087.50	171,649.18	98,122.91	668,161.65
År 11	1.1	343,930.25	47,158.95	296,771.30	114,787.30	.055	73,425.00	188,212.30	108,559.00	776,720.65
År 12	1.1	378,323.28	51,874.85	326,448.43	129,709.64	.0575	76,762.50	206,472.14	119,976.28	896,696.93
År 13	1.1	416,155.60	57,062.33	359,093.27	146,571.30	.06	80,100.00	226,671.30	132,421.37	1,029,118.30
År 14	1.1	457,771.16	62,768.57	395,002.60	165,626.25	.0625	83,437.50	249,063.75	145,938.85	1,175,057.15
År 15	1.1	503,548.28	69,045.42	434,502.86	155,441.10	.065	86,775.00	242,216.10	192,286.76	1,367,343.90
Summa					1,335,000.00		951,187.50	2,286,187.50	1,367,343.90	







CP/M på ABC-datorer

CP/M

några synpunkter och erfarenheter

CP/M är ett av världens spridda operativsystem, och det finns nog inget system som finns på så många olika datorer. Jag kommer att enbart behandla CP/M 2.2, som finns på såväl ABC-80 som ABC-800. CP/M 3.0 eller CP+ som endast finns på ABC-806 och nya DTC-2 kräver extraminne m m.

Fördelar med CP/M:

det finns

det är mycket spritt

det finns mycket programvara och språk. Genom CPMUG finns en hel del program till mycket lågt pris, ungefär som via ABC-klubben.

det finns nästan alla programspråk under CP/M för den som vill förkovra sig.

Nackdelar:

på ABC-80 och ABC-800 krävs extra hårdvara med extra minne.

diskhanteringen är långsam jämfört med ABC-DOS.

standardeditorn, ED är litet knölig innan man vant sig. Den är TTY-orienterad och TECO-liknande och träig om man är bortskämd med skärm-orienterade editorer. Det finns givetvis att köpa avancerade sådana editorer, t ex VEDIT.

CP/M är utrymmeskrävande på disk. Att köra på 830-disk innebär ett evigt skivbyte, och att köra på en FD2 torde vara etter värre.

Vad finns det då för möjligheter att köra CP/M på ABC-datorer? För ABC-80 finns det MyAB CP/M. Det består av ett litet kort, som man sätter in mellan CPU-kretsen och det stora kortet. Det är mycket enkelt att montera, man lirkar upp CPU-kretsen och lossar en skruv, som fäster moderkortet i lådan. Sedan sätter man ett distansrör mellan CP/M-kortet och moderkortet och trycker fast CP/M-kortet i CPU-hållaren. Till sist sätter man in CPU:n i CP/M-kortet. Sedan är det bara att montera ihop ABC-80, sätta CP/M-skivan i disken och köra. Det finns mycket bra monteringsanvisningar till alla MyAB:s kort. Om man har TKN-80 inmonterad, utnyttjar CP/M detta automatiskt.

På ABC-800 finns dels MyAB:s CP/M dels numera även Luxors CP/M. Båda kräver ett minneskort om 64 kB, som monteras mellan CPU-kort och video-kort. För att köra MyAB:s CP/M krävs dessutom att man har MyAB UNI-800-kortet som 64kB. På detta finns ett BOOTPROM. Använder man UNI-800/G finns det en monteringssats för att kunna köra HR-kortet i maskinen. Luxors CP/M nyttjar en klurig BOOT via ett 'fusk-bibliotek' i sektor 16, som i sin tur mappar om minnet och laddar in CP/M. Bootprogrammet undersöker även om det ABC-800 eller 802/806 och lägger ut olika kod i Bios, beroende på memory-mapping hårdvara. Luxor CP/M har stora fördelar, det finns dock även fördelar hor MyAB-CP/M.

1. Programmet ABCDISK under CP/M gör det möjligt att flytta ASCII-filer från ABC-DOS till CP/M. Programmet nyttjar CP/M BIOS och fungerar inte under Luxor CP/M.

2. Programmet CPMDISK flyttar filer från CP/M till ABC-DOS under ABC-DOS.

3. Källkoden till BIOS och BDOS finns med, så man kan se hur det är gjort och modifiera BIOS vid behov.

4. BASIC-minnet (29k på ABC-800) används som RAM-disk och kallas C:.

Nackdelen med detta är att det är litet struligt att generera system för andra konfigurationer. I detta fall är SYSGEN under Luxor CP/M ett synnerligen lätthanterligt system. När man startar SYSGEN får man upp aktuell konfiguration, och man kan sedan enkelt uppdatera vilken disk som är A:,B: C:,D: etc. Har man tillgång till en maskin med flera olika disk-stationer, är det synnerligen enkelt att göra olika system och att flytta filer mellan olika format. Samma skiva kan dessutom köras i ABC-800 med CP/M-kort, i ABC-802 och i ABC-806.

Kör man på en enda maskin, går det på ett ut, behöver man veva runt på olika maskiner med flera diskar är Luxor CP/M ett måste.

På ABC-80 finns bara MyAB CP/M, och likaså om man vill köra med 5" SSSD eller single side , double track (320kB). Vill man använda ABC-802 eller ABC-806 finns det endast Luxor CP/M eller CP/M 3.0 från MyAB. För ABC-802 kommer från MyAB snart ett minneskort med upp till 512 kB för inbyggnad. Whow!

CPMUG är uttytt CP/M Users Group. De har ställt iordning ett bibliotek med f.n. 77 st 8" SSSD skivor med program av varierande kvalitet. De kostar 50 kr. per skiva från MyAB. Vill man ha 830-skivor, får man varje CPMUG-skiva på två skivor för 100 kr. Eftersom programmen är helt fria får man ge bort dem hur som helst. Priset är ju i stort sett mediakostnaden.

Ett antal av de bättre programmen kommer från Ward Christensen, som tycks ha vigt sitt liv åt att skriva program till CPMUG. Här finns t ex MODEM, som är ett generellt filtransfereringsprogram för CP/M-datorer och REZ, som är en mycket avancerad disassembler. Från CPMUG finns diverse assemblers, en enkel Pascal, Ratfor, Adventure, ett enkelt Algol-system och mycket annat. De flesta program är i 8080 assembler, och en del måste yxas till för att kunna köras. Det är till stor del programvara för och av dator-amatörer.

Om man vill ha tag i en Pascal billigt, finns JRT Pascal för 36 dollar. Den har en del begränsningar och vid kompileringsfel stoppar den vid första påträffade fel. Den fungerar dock. Den säljs under villkoret att man får ge bort kopior till sina vänner, så om man slår i hop sig några stycken, behöver man inte bli ruinerad på Pascal. Det finns ett antal Pascal-system under CP/M. Det bästa och dyraste är nog Pascal/MT+ för c:a 5000:-.

Det finns en hel del C-kompilatorer under CP/M, även om 8080-arkitekturen inte är den bästa för C. En prisvärd sådan är C/80 från Software Toolworks. Den säljs av Elektrokonsult, N-3000 Drammen, Norge för 400 NKR. Den är baserad på Ron Cains Small C och innehåller det mesta vad beträffar int och char. I Byte fanns en artikel där man jämförde olika C-kompilatorer under CP/M. BDS-C verkar vara en annan prisvärd kompilator till överkomligt pris, som är utrymmessparande på disken. En del C-kompilatorer, tex. C/80 skapar en mellanfil på disken med Assembler-kod, och denna är ytterligt platskrävande.

För den som vill skriva assembler under CP/M och vill använda Z80-instruktioner finns det dels Z80ASM från CPMUG, dels UVMAC från Software Toolworks. Den senare är en Macro-assembler. Även Microsoft M-80 lär klara Z-80 kod.

Vidare finns givetvis Basic, främst Microsoft Basic om man har 4000:- över. Skall man få snabba program krävs dessutom BASCOM, Basic kompilatorn för ytterligare drygt 5000:-. Det finns ADA, LISP, FORTRAN, COBOL, FORTH och Modula-2 och mer kommer. Priserna varierar kraftigt, från 30 dollar och uppåt.

Man kan också använda ett CP/M-system som utvecklinssystem för andra mikroprocessorer då det finns korsassemblatorer för de flesta CPU:er från tex. SORCIM. De säljs av Applitron i Göteborg. Det finns t o m C-korskompilatorer under CP/M, men de är ganska ineffektiva.

De flesta program under CP/M kan levereras på 8" SSSD. Många går att köra direkt, under det att en del måste yxas till. För att få över dem till annat format krävs en ABC-800 med flera olika disk-stationer. Kan man få tillgång till en sådan maskin, finns det mycket att välja på. Läser man BYTE finns det sida upp och sida ned med annonser. Man får dock alltid vara beredd på att vänta och på att inget händer.

I Sverige kan man köpa CP/M-program dels från MyAB, dels från Applitron i Göteborg. MyAB kan leverera på alla ABC-format och de har även CPMUG. Applitron har ett stort urval av programvara, men levererar främst på 8" SSSD.

Som avslutning kan jag säga att CP/M är ett bra komplement till ABC-datorerna om man vill gå utanför BASIC och det som finns för ABC-DOS. Det kostar dock en hel del, så man måste vara motiverad. Dyrast blir det på ABC-80 och ABC-800. På ABC-802 och ABC-806 kommer man billigare undan genom Luxor CP/M.

Torbjörn Alm <116>

DIV:

Om TV-editorn

Efter att åtskilliga gånger försökt hämta in en fil från skivan med kommandot ;S med resultat att filen fått skrivas in på nytt eller reparerats med t.ex. programmet LÄSFIL2, har jag nu infört följande ändringar.

```
3135 IF LEN(M$)=0% GOSUB 3820
3820 ;CHR$(7)CUR(23,0)"Varning! Vill du spara en tom fil?"
3830 FOR I=1 TO 1000 : NEXT I : RETURN
```

Rad 3135 kollar om minnet är tomt och hoppar till subrutinen på rader 3820 och 3830. Klocksignal och text ger varning för eventuell överlagring på redan skriven fil, men tillåter efter ca 1 sek väntan att ändå gå vidare, men då förhoppningsvis mera medvetet.

Ändringarna har använts ett tag och fungerar bra. De lades in i TVMAIN samtidigt med ändringarna enligt ABC-bladet nr 2, därför radnr enligt ovan.

Stig Berlin < 231>

Denna ändring kan också göras i TVMAIN.800 (versionen för ABC800-datorer, men då måste rad nr 3135 enligt ovan ges radnummret 3130.

Bo Kullmar

Har någon en flexskiveenhet med dubbel densitet (FD2D/DD82) att sälja?

I så fall hör av Er till

<3063>
Henry Lidholm
Vårstavägen 99
140 32 Grödinge

0753-291 05 eller 0753-392 06

Spara attributminnet för ABC806

I en tidigare artikel om hardcopy för ABC800-datorer skrev jag att jag inte visste hur man kan spara undan attributminnet. Det har jag nu fått reda på och kan därför redovisa det här.

När man läser av ett tecken i bildminnet kan man samtidigt få tag på attribiuttecknet genom INP(53). Detta skall enligt uppgfit gå bra både i assembler och BASIC. Jag har ännu bara provat det i BASIC.

I BASIC gär man så här för att lagra undan bildminnet:

```
Tecken=PEEK(30720) : Attribut=INP(53)
```

För att skriva tillbaka det gör man:

```
OUT 53,Attribut : POKE 30720, Tecken
```

Använd ej Attribut som variabel, eftersom detta är ett BASIC-ord.

Nedan följer två BASIC-rutiner som omfattar hela bildminnet:

```
300 DIM Screen(2048),Attribut(2048)
2000 DEF FNSave806screen
2010    Curpos=PEEK2(65362)
2020    FOR I=1 TO 2048
2030        Screen(I)=PEEK(I+30719) :
Attribut(I)=INP(53)
2040    NEXT I
2050    RETURN 0
2060 FNEND


3000 DEF FNPrint806screen
3010 POKE 65362,Curpos,SWAP%(Curpos)

3020    FOR I=1 TO 2048
3030        OUT 53,Attribut(I) : POKE
I+30719,Screen(I)
3040    NEXT I
3050    RETURN 0
3060 FNEND
```

Ovanstående rutiner är dock ganska långsamma. Ungefär 6 sekunder tar varje rutin. Om man använder WHILE-loopar för att få I som lokal variabel, så tar det ytterligare någon sekund. Detta beror på att en WHILE-loop inte är så optimerad för snabbhet som FOR-NEXT-looparna.

Dessutom tar dessa rutiner upp 8 kbyte av minne. Skriver man om rutinerna i assembler så bör de bli avsevärt snabbare. Dock kvarstår då problemet med att det tar upp mycket minne för att lagra undan både bild och attributminnet, även om lagringen kan göras effektivt så att man enbart tar i anspråk 4 kbyte. En lösning på detta kan vara att skriva en assemblerrutin som lagrar undan blidminnet på en fil i RAM-floppyn. Då får man först testa om drivrutinen för RAM-floppyn finns laddad. Om så inte är fallet, ja då föreslår jag att man helt enkelt struntar i att lagra undan bild och attributminnet på detta sätt. Man gör helt enkelt som man gör på de andra 800:orna.

Bo Kullmar

Speciell teckenmode för ABC806

ABC806 kan ställas om med kommandot OUT 34,134 så att tangentbordet lämnar ett speciellt tal när en tangent trycks mer och ett tal när en tangent släpps upp. Detta gör att tangentbordet inte längre lämnar ASCII-koder utan VARJE tangent lämnar ett tal beroende på vilken tangent som har tryckts ner. Siffran 1 på den alfanumeriska delen av tangentbordet ger alltså inte samma kod som siffran 1 på den numeriska delen av tangentbordet. Återställning till normal mode görs med OUT 34,6.

Denna speciella "up/down mode" kan användas om man vill flytta eller byta funktioner för tangenter. Därvid måste man dock skriva en assemblerrutin för att tolka tangentnedtryckningarna på det sätt som man önskar. Då måste man också själv hålla reda på om någon shift tangent har tryckts ned eller om capslock-tangenten är nertryckt. Lysdioden i capslock kan i "up/down mode" tändas med OUT 34,136 och släckas med OUT 34,8.

För att testa detta kan man skriva ett enkelt basicprogram:

```
10 OUT 34,134
20 FOR I=1 TO 30
30GET Ö$ : ; ASC(Ö$);
40 NEXT I
50 OUT 34,6
```

FOR-NEXT loopen finns med så att man kan komma ur "up/down mode" utan att initierat det från tangentbordet, vilket ju i detta läget är omöjligt.

Dessutom finns följande funktioner om också inte står i bruksanvisningen:

```
OUT 34,9   Ljud, 5 ms klick
OUT 34,10  Ljud, 20 ms klick
```

Bo Kullmar

Funktionstangenter i ABC800-datorer

Pf 1 ger decimalt värdet 192 och Pf 8 ger 199, medan SHIFT + Pf 1 ger 208 samt SHIFT + Pf 8 ger 215.

Detta verkar inte särskilt logiskt! Om man tittar på vilka binära koder som Pf-tangenterna generar så kommer förklaringen fram!

```
Bit 0      är alltid  1
"   1      "  "       1
"   2 - 3  ger ett tal decimalt 0 - 3
"   4      är alltid  0
"   5 - 7  ger ett tal decimalt 0 - 7
```

Bit 2 - 3 anger alltså om någon shift-tangent är nedtryckt.

```
Om decimalt:   Nedtryckt tangent:
0              Enbart Pf-tangent nedtryckt
1              Pf+SHIFT
2              Pf+CTRL
3              Pf+SHIFT+CTRL
```

Bit 5 - 7 anger vilken Pf-tangent som är nedtryckt. Om man adderar 1 till värdet så får man fram Pf-nummret.

På tangetbordet ABC55 som normalt används till ABC802 generas Pf-koderna med hjälp av vanliga tangenter i kombination med SHIFT och CTRL. På samma sätt sker även generering av Pf-koder på tangentbordet ABC77, som normalt används till ABC806, men eftersom detta tangentbord också innehåller vanliga Pf-tangenter så är det inget som man använder.

Av Bo Kullmar

Hastigheten för klockan i ABC800

Två olika sätt att ändra klockans hastighet i ABC800-datorer har medlemmar kommit på.

Carl-Axel Pettersson, <2229>, föreslår följande:

```
OUT 99,208,99,X,99,Y
```

X kan t ex vara:

```
37 = Stänger av klockan
133 = Klockan går 16 ggr fortare
165 = Normalt värde
```

Y är tidskonstant som normlat är 125.

En annan medlem har hitta följande metod:

Klockans hastighet ändras med OUT 99, 133,99,N. N kan vara 0 - 255. Klockan kan stängas av med OUT 99,1.

Bo Kullmar

Problem med ABC806

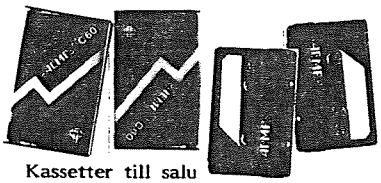
På de först producerade ABC806 har upptäckts två fel vid speciella tillämpningar enligt Luxor Datorer Serviceinformation, 1983-09-06 2-01.

1. Vid körning mot expansionslåda kan det uppstå problem att läsa status från databussen.

2. Störningar på inkommande klocka från tangentbordet. Datorn missar tecken från tangentbordet. Problemet visar sig vid körning med hög hastighet över V24:.

Om man har en ABC806 som berörs av ovanstående fel kan man få maskinen gratis lagad via sin återförsäljare. Det framgår också av nämnda servicemeddelande hur felet åtgäras, men detta måte ske av en tekniker.

Bo Kullmar



Kassetter till salu

ABC-Klubben har köpt ett antal ALME C60 kassetter som vi inte kan använda till ABC-Kassetter eftersom de inte går att snabbkopiera. Detta beror på att mekaniken är dålig i en del av kassetterna.

ABC-Klubben säljer därför ut kassetterna för **300 kronor per sats om 100 kassetter.** Kassetterna går att använda, men är av dålig kvalite. De som är mekaniskt felfria kan gå att använda till data. Klubben lämnar ingen garanti på kassetterna, en del kan alltså vara dåliga. Vi säljer dem därför billigt.

Kassetterna erhålles genom att sätta in beloppet på klubbens postgirokonto för **publikationer 62 93 00-5.** Glöm ej den egna adressen.

ABC-Klubben/Styrelsen

SUPER SMARTAID -kraftfullt hjälpmedel för ABC 80

Starkt beroendeframkallande och med uppenbar risk för tillvänjning; det är omdömen som brukar mana till försiktighet i användningen i många fall. Beskrivningen kan emellertid helt berättigat appliceras även på OWOCO AB:s nya produkt SUPER SMARTAID. Och då är riskmomenten obefintliga, medan nyttan och de positiva effekterna är uppenbara.

SUPER SMARTAID är ett tillbehör till ABC 80 och betecknas av tillverkaren, Huddinge-företaget OWOCO AB, som ett programmeringshjälpmedel. Men det bjuder i realiteten på bekväma funktioner och god hjälp både före, under och efter själva programmeringsfasen. Autostart, KEY-funktioner, spooling, programkörning i "single step" och JOB-strömmar är några av dessa funktioner. De ger ABC 80 ett helt nytt och intressant arbetsregister.

SUPER SMARTAID omfattar 10K-byte programvara. Den ligger lagrad i minneskapslar av PROM-typ. ABC 80:s eget arbetsminne tas alltså inte i anspråk, utan det kan användas ograverat för programmering. AID:en har dessutom ett eget arbetsminne (CMOS-RAM) på 2K-byte för lagring av olika uppgifter. Allt som ingår i SUPER SMARTAID är "förpackat" i en bastant plåtlåda, klar att anslutas till den 64-poliga ABC-bussen.

SNABB INSTALLATION

Installationen är enkel. Man lossar de bakre gummifötterna som ABC 80:s tangentbord normalt vilar på. Därefter passas hjälpmedlets honkontakt in i kontaktdonet baktill på ABC 80. En plåtskena nedtill på AID:en ansluter då mot bottenplattan på tangentbordet och två små, förhandsborrade hål i plåtskenan ger möjlighet att skruva fast gummifötterna i datorn igen. Därmed sitter även AID:en fast. I mitt fall var passningen inte helt perfekt, men sedan hålen i plåtskenan hade förstorats något, gled skruvarna lätt in på plats.

När vi slår på spänningen är SUPER SMARTAID genast initierad och funktionsberedd. Datorns klarsignal "ABC 80" på bildskärmen ersätts nu med beskedet "SMARTAID" som tecken på att hjälpmedlet står till vårt förfogande.

Vid initieringen gör SUPER SMARTAID ett försök att ladda en fil och utföra ett JOB med namnet SUPER.JOB. Om JOB-filen är så utformad, att den startar ett BASIC-program, har vi på så sätt tillgång till en autostartfunktion.

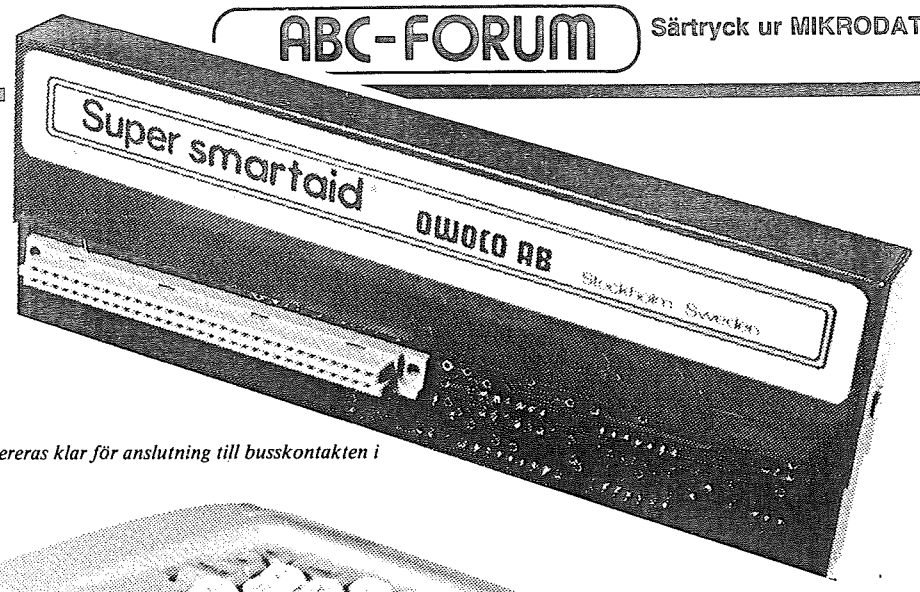
Ett exempel: Vi vill ha autostart av programmet LIB.BAC och se storleken på filerna (alternativ "S" i LIB-programmets meny). På textskivan lägger vi då in följande textrad: "RUN LIB__M S". Denna rad har vi sparat under namnet SUPER JOB. Det hela resulterar i att SUPER SMARTAID startar och kör det önskade programmet vid initiering av AID:en. (En förkla-

```
SMARTAID
HELP
Ü         PEEK      FIND      CHANGE    VAR
SYS       KEY       CONT      START     RESUME
SPOOL     DISP      TLOAD     TSAVE     TMERGE
LIB       DIR       HELP      TAB       EX
AUTO      OLD       JOB       PEW       POW
TIME      NOTIME    STACK     BYE       RUN
LIST      NEW       SCR       CLEAR     LOAD
MERGE     REN       ED        SAVE      UNSAVE

SMARTAID
```

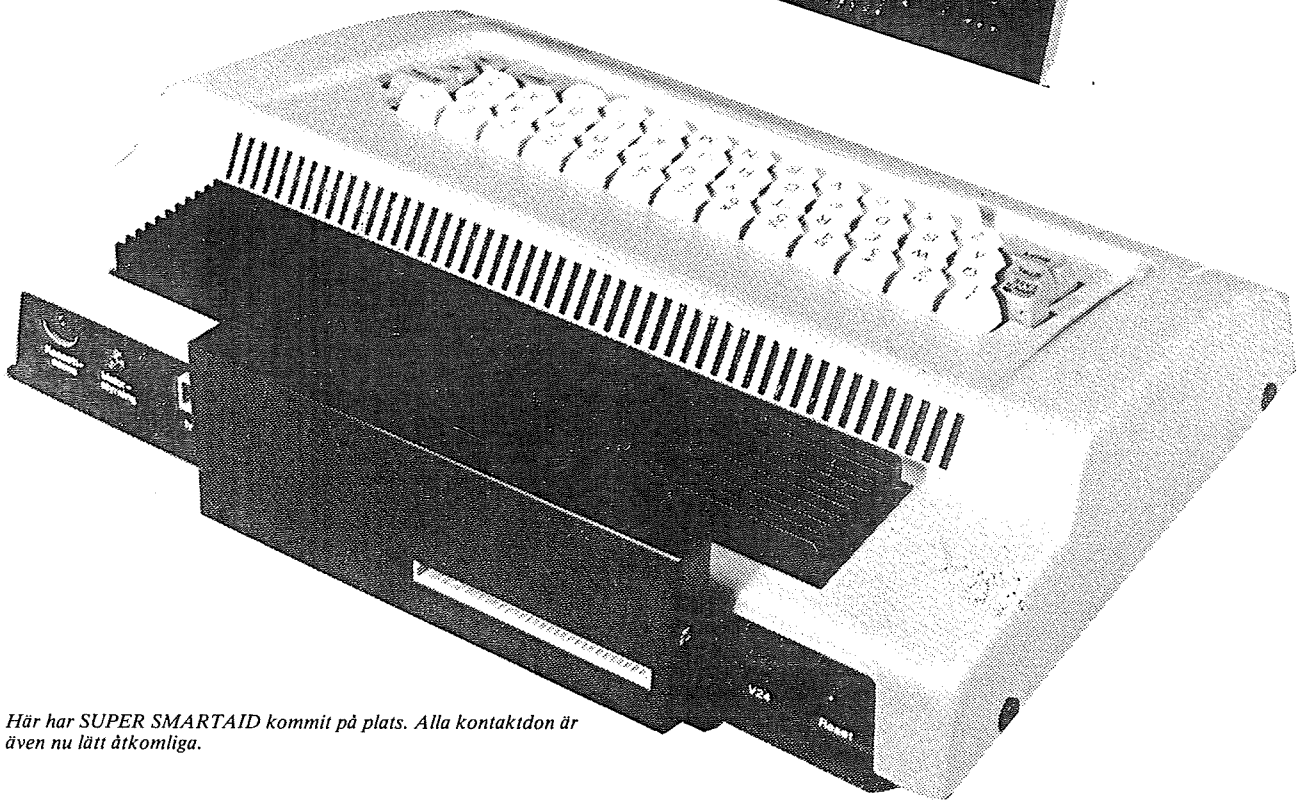
Figur 1
"HELP" hjälper oss att komma ihåg vilka kommandon som SUPER SMARTAID erbjuder.

```
SMARTAID
SYS
SUPER SMARTAID

MEMMAP: PROG 1924   VAR  822    FREE 27205

SYSVAR: BOFA 32768 EOFA 34692 HEAP 35515
        STCK 62720 VROT 34693 RAM  32768
        BKGD 0     CMOS 22000 DVRT 28339
TIME  : 19:43:32
DEVICE: DR0, DR1, DR2, DR3, DR4, DR5,
        DR6,    , CAS,    , PR , IEC
SMARTAID
```

Figur 2
SUPER SMARTAID ger omfattande systeminformation som svar på kommandot SYS.



SUPER SMARTAID levereras klar för anslutning till busskontakten i ABC 80:s tangentbord.



Här har SUPER SMARTAID kommit på plats. Alla kontaktdon är även nu lätt åtkomliga.

ring: __M är koden för CTRL-M = RETURN).

TEXTBEHANDLING

Det utmärkande för SUPER SMARTAID är att den förses ABC 80 med en rad nya direktfunktioner. För att skapa och spara den textrad, som förekom i föregående stycke, använder vi lämpligen AID:ens egen texteditor. Inskrivning av texten startar vi genom att som första programrad skriva en asterisk (*). Så här:

```
10 *
20 RUN LIB__M S
```

Med ED-kommandot tar vi därefter bort rad 10. Den återstående, önskade, raden sparar vi sedan genom att skriva TSAVE SUPER. JOB. Snabbt och enkelt utan att behöva ladda in ett textbehandlingsprogram från någon yttre enhet! Raden sparas med det här förfarandet utan att föregås av radnummer.

Lika enkelt är det att med direktkommando läsa en textfil. En sådan hämtas genom att skriva TLOAD (filnamn). I detta fall tillfogas radnummer till texten och editering kan göras med ED-kommando. Vill vi slippa radnummer, kan textfilen läggas ut på skärmen med kommandot DISP (filnamn). I det fallet påverkas inte innehållet i datorns internminne av våra åtgärder.

EGNA FUNKTIONER

Hela SUPER SMARTAID:s kommandorepertoar listas på skärmen med HELP-kommandot. (Se figur 1.) Därutöver finns en rad funktioner, huvudsakligen för bildskärmseditering. Dessa styrs med ungefär 20 olika CTRL-(tecken)-kombinationer. Men dessutom håller AID:en fortlöpande reda på en rad systemdata, som vi kan få upp på skärmen, närhelst vi ger kommandot SYS. (Se figur 2.) Om nu inte allt detta skulle räcka till, har vi möjlighet att tilldela enskilda tangenter (eller CTRL + tangent) nya funktioner helt efter vårt eget huvud. Det gör vi genom att använda KEY-funktionen.

Skriver vi exempelvis KEY__T = ''TIME__M'', tilldelas kombinationen ''CTRL-T'' funktionen TIME och lägger till RETURN. Detta medför att vi lägger ut datorns realtidsklocka längst upp till höger på skärmen. Dessförinnan har vi lämpligen ställt klockan, vilket vi gör med TIME hh,mm,ss.

De KEY-funktioner, som vi själva har definierat (Ex: KEY__N = ''NOTIME__M''), sparas i AID:ens eget CMOS-RAM. SUPER SMARTAID inrymmer också en laddningsbar NICAD-ackumulator som fulladdad klarar av CMOS-minnets strömbehov under ungefär 60 dygn. De KEY-funktioner som vi

har lagt in finns alltså direkt tillgängliga även nästa gång vi slår på datorn. Ackumulatorn får sin laddning under den tid som datorn är ansluten till elnätet (datorn påslagen).

NYA MÖJLIGHETER

Med kommandot KEY kan vi göra en listning på bildskärmen av de definitioner som ligger lagrade i CMOS-minnet. (Se figur 3.) Allt i en KEY-funktion som följer efter första "_M" ignoreras. Därför kan vi, om vi så vill, skriva en förklarande kommentar efter KEY-definitionen. Exempel på detta framgår av figur 3.

Att KEY-definitionen bara kan innehålla "_M" en gång innebär på sätt och vis en begränsning av vad som kan åstadkommas direkt med KEY-kommando. Låter vi emellertid som i figur 3 en KEY-funktion enbart starta en JOB-fil, kringgår vi lätt den begränsningen. En sådan fil kan klara många successiva uppgifter och se ut t ex som figur 4 visar. Där startar den ett program (TEST.BAS) och matar direkt in åtta svar på INPUT-satser i början av programmet. Finessen är alltså att allt detta kan åstadkommas med att bara trycka ned två tangenter CTRL + J. Och heter filen SUPER.JOB, får vi till och med automatstart.

PROGRAMMERINGS-HJÄLP

När vi sätter oss ned för att programmera, sköter datorn själv om radnumreringen, sedan vi gett kommandot AUTO. Gör vi syntaxfel i en lång programrad och får felmeddelande, är inte raden räddningslöst förlorad som med den "nakna" ABC 80. Med CTRL-tangenter kan vi flytta upp markören på raden igen, rätta felet och därefter mata in den korrekta lydelsen. Både INSERT- och DELETE-möjligheter finns vid editering. Vi kan välja på ett flertal sätt för att flytta markören över bildrutan och för att mata in text i inmatningsbufferten. CTRL-L rensar skärmen helt eller delvis.

```
SMARTAID
KEY
 J="JOB SUPER.JOB_M" ! Startar fördefinierad JOB-fil.
 T="TIME_M" ! Visar realtid i övre högra hörnet av skärmen.
 N="NOTIME_M" ! Tar bort utskriften av TIME från skärmen.
 ü="ü- M"     ! Listar från programmets början.
SMARTAID
```

Figur 3
Med KEY får vi besked om vilka KEY-funktioner som ligger lagrade i CMOS-minnet.

```
SMARTAID
TLOAD SUPER.JOB
SMARTAID
ü
10 RUN TEST.BAS_M J_M 12_M 3_M 25_M ;"HEJ"_M ;"HOPPSAN"_M
   ;"GÅR"_M ;"BRA"_M

SMARTAID
```

Figur 4
Exempel på hur en JOB-fil kan utformas.

Vill vi veta vilka variabler som har tagits i anspråk i ett program, ger vi kommandot VAR. Med CHANGE kan vi i ett enda moment göra generella ändringar i ett helt program. Exempelvis ändra variabeln B till B% på alla ställen där den förekommer. Det är ett bekvämt sätt att använda i stället för att hela tiden brottas med att behöva skriva %-tecknen manuellt.

REN-kommandot får utökad repertoar med SUPER SMARTAID. Det är t ex möjligt att numrera om enbart ett avsnitt i ett program, medan resten behåller sina radnummer. Den möjligheten kan även användas i text-mode och ger då möjlighet att flytta hela textavsnitt inom den textmassa som ligger i internminnet (BLOCK MOVE).

Med kommandot DEL (DELETE) tar vi bort en eller flera rader ur programmet med ett fåtal tangentnedtryckningar.

MÅNGSIDIGT

Även LIST-funktionen är förbättrad. Som snabbkommando kan vi skriva "Ü RETURN". Listning sker då (första gången) från första programraden. Nästa gång vi använder samma kommando, startar listningen med den skärmbild, där föregående listning avbröts. Listningen kan rullas (scrollas) nedåt eller uppåt på skärmen med hjälp av piltangenterna. Skriver vi "LIST-" eller "Ü-", sker listningen alltid från första raden. Det kan vara praktiskt att, som i vårt exempel (figur 3), lägga in den varianten som en bekväm KEY-funktion.

FIND är ett kraftfullt kommando att använda när vi söker efter en variabel, en sträng eller ett radnummer i ett program eller i en text. Alla rader där det sökta begreppet återfinns listas då i sin helhet. Med mellanslagstangenten eller RETURN släpper vi fram *en* sådan rad i sänder, till dess att vi når slutet på programmet. Editering kan vi göra direkt utan att behöva gå ur FIND-funktionen. Det hela är mycket värdefullt vid felsökning och "avlusning" av program.

Vid vanlig programlistning kan editering ske över flera rader i "ett svep". Ett antal fördefinierade CTRL-tangenter ger oss då möjlighet att direktförflytta markören ett stycke framåt eller bakåt i texten till ett önskat tecken.

STEGVIS

När programmet är klart att köras, kan vi göra det på vanligt sätt med RUN. Men vi kan också starta från valfri rad. I det fallet skriver vi START (radnr) i stället för RUN.

Efter stopp i programmet (vid STOP eller ERROR) går det att återuppta körningen med bibehållna variabelvärden. Det sker med CONT eller RESUME. Datorn kör då automatiskt fortsättningen av programmet i "step-mode". Det innebär att exekveringen sker en rad i sänder i den takt som vi själva bestämmer. Även i det fallet står flera möjligheter till vårt förfogande.

Med CTRL-D får vi den programrad som står i tur att exekveras utskriven i sin helhet. Det sker längst ned på bildskärmen. Med den stegvisa funktionen kan vi läsa rad för rad i den takt vi trycker fram nya och direkt se vad de utför. Det är en funktion som har klara pedagogiska fördelar.

Väljer vi i stället CTRL-T, får vi tillgång till TRACE-funktion. Enbart radnumren visas då på nedersta skärmraden, när vi stegar oss framåt i programmet. CTRL-S, slutligen, använder vi vid stegvis körning, när vi inte behöver något av den hjälpinformation som CTRL-D eller CTRL-T kan förse oss med. Vid normal körning (utan STOP eller ERROR) kan vi direkt sätta datorn i STEP-MODE med tangentkombinationen CTRL-C.

När det är dags att spara vårt program på flexskiva, kan vi först ge kommandot LIB (eller DIR om vi har 8-tums flexutrustning). Då får vi direkt upp information om vilka filer som redan finns på skivan och hur mycket utrymme som återstår. Programmet i datorns primärminne påverkas inte av den åtgärden.

SPOOLING

Med CTRL-SHIFT-O kan vi när som helst "dumpa" skärminnehållet på en ansluten skrivare. Printerparametrarna ligger för övrigt lagrade och klara i CMOS-minnet, sedan de en gång har matats in.

Lista programmet på skrivare kan vi göra på vanligt sätt, med LIST- (eller Ü-) (filnamn) PR:, men här bjuder SUPER SMARTAID återigen på valmöjligheter. AID:en klarar nämligen också spooling. Det är därmed möjligt att överföra textfiler till printer samtidigt som ABC 80 är användbar till andra uppgifter, t ex programmering. Vill vi nyttja den funktionen, skriver vi SPOOL (filnamn), vilket startar utskriften.

ABC 80 delar nu sin tid mellan förgrunds- och bakgrundsjobb. Utskrift på printern pågår hela tiden och datorn har till sy-

nes tid med operatören bara lite då och då. Skriver vi exempelvis med jämn och snabb takt in ett program eller en text, kommer resultatet upp "skurvis" på skärmen. Tecknen kommer med viss fördröjning, flera samtidigt vid varje tillfälle. Detta verkar ovant och något förvirrande, åtminstone i början. Men det fungerar ovedersägligen. Våra tangentnedtryckningar lagras först i en typeahead-buffert för att därefter tas om hand av datorn, när den "får tid". På grund av det här beteendet sätter jag nog ett litet frågetecken i kanten beträffande möjligheten att verkligen kunna dra nytta av SPOOL-funktionen i praktiskt arbete.

MER FINESSER

Med ovanstående beskrivning har jag någorlunda utförligt exemplifierat hur SUPER SMARTAID kan vara till snabb och omedelbar hjälp under ett körpass. Men ytterligare möjligheter återstår. Med PEEK eller PEW (PEEK a Word) kan vi direkt lista och läsa en respektive två byte i taget i primärminnet. Liksom vid listning av program kan vi scrolla minnesinnehållet (vid PEEK) framåt eller bakåt med piltangenterna. Med POW (adress) skriver vi in två byte direkt i primärminnet, medan POKE används som vanligt.

Bland ytterligare specialiteter bör jag slutligen nämna även kommandot OLD. Med det kan vi återkalla och köra ett program även efter NEW eller RESET. En annan finess är körning av FOR-NEXT-loopar även som kommando, utan radnummer.

SLUTOMDÖME

I praktisk användning (under några månaders tid) har SUPER SMARTAID visat sig helt problemfri. Det mest dramatiska som har inträffat är att CMOS-minnet vid ett par tillfällen har tappat sitt innehåll. Det inträffar om ABC 80 "spårar ur" i samband med någon, oftast olämplig, åtgärd. Vid sådana tillfällen skrivs "CMOS JEOPARDIZED" för övrigt ut vid nästa initiering av AID:en. Det är ett besked om att data i CMOS-minnet har gått förlorade eller är otillförlitliga.

Med den stora funktionsrepertoar som SUPER SMARTAID har, tar det en hel del tid att lära sig att hantera alla finesser med någorlunda fulländning. De 20 CTRL-tangenternas funktion (och ett varierande antal egna KEY-funktioner) kan vara svåra att ha aktuella i huvudet. Men man lär sig snart nog att behärska några favorithandgrepp som klarar elementära men nyttiga grundfunktioner. I övrigt listar man lämpligen övriga funktioner med HELP-kommandot, när det egna minnet sviker.

Slutligen går det bra att rådfråga den bruksanvisning som finns. Den omfattar 25 maskinskrivna A4-sidor. I den beskrivs, omfånget till trots, hjälpmedlets funktioner tämligen koncentrerat och där ges exempel på hur man kan göra. OWOCO AB betecknar själv (juni 1983) bruksanvisningen som preliminär. Men den är helt tillfyllest för den som redan är någorlunda bevandrad med den terminologi som förekommer i datorsammanhang. Nybörjaren kan kanske ha lite svårare att tillgodogöra sig hela innehållet.

Samma omdöme kan för övrigt gälla SUPER SMARTAID som helhet. Alla — inte minst nybörjare i datorsammanhang — har stor nytta av flertalet av hjälpmedlets bekväma och snabba funktioner. Men det är främst den något mer avancerade tangenttryckaren och programmeraren som kan använda och ha nytta av hela den breda funktionsrepertoaren. Det är med andra ord ett hjälpmedel med "högt i tak". Något för flertalet att växa i.

MYCKET PENGAR — ÄNDÅ BILLIGT

Finns det då inte några nackdelar bland alla de positiva egenskaperna? Ja, det skulle i så fall vara priset. Den som kompletterar sin ABC 80 med SUPER SMARTAID får räkna med att spendera nära 2 800 kronor plus moms. Det är alltså den summa som går åt för att förvandla ABC 80 till en verkligt intressant maskin. Men det är en summa som måste anses i hög grad överkomlig med hänsyn till resultatet — en uppdaterad och i många avseenden "ny" och kraftfullare dator.

Karl-David Höglund
ABC-Forum
Box 8040
900 08 UMEÅ

Meddelande till ABC-klubbens medlemmar!

Som framgår av denna artikel lovordas SUPER SMARTAID av författaren Karl-David Höglund.
Han säger vidare att om något kan räknas till nackdelarna så skulle det möjligen vara priset.
Vi har tagit fasta på detta och kan glädja medlemmarna med ett sänkt pris till 1.985:- inkl moms.
Beställ Din SUPER SMARTAID direkt från OWOCO AB, Kvarnbergsvägen 25, 141 45 Huddinge tel. 08 - 774 02 90.
Du kan också kontakt någon av OWOCO:s återförsäljare som Du får tel.nr till genom att ringa OWOCO.

PS. Vi har infört nya finesser till SUPER SMARTAID (okt. -83). Även SMARTAID 3 har fått sänkt pris 995:- inkl moms.