TV-editorn

```
*****************************
*                           *
*          T V              *
*                           *
* A screen oriented editor  *
*                           *
* IMPLEMENTED AT LYSATOR    *
* LiTH, Linköping           *
*                           *
* BY  ANDERS ISAKSSON       *
*     ANDERS SUNDQVIST      *
*     ANDERS STRÖM          *
*     ----------------      *
*                           *
*****************************
```

Hur använder man TV-editorn

Det är säkert många medlemmar som ännu inte upptäckt vilken utmärkt nyttighet ni fått tillgång till genom TV-editorn som fanns på ABC-kassett nr 2 med kompletteringar och modifieringar på ABC-kassett nr 3. Kanske beror det på att den medföljande beskrivningen var i knappaste laget.

ABC-80 har en inbyggd editor. Den har alla som skrivit något program på ABC-80 kommit i kommit i kontakt med och lärt sig hur man med höger och vänsterpilarnas hjälp ändrar en programrad. Tyvärr kan den inbyggda editorn bara användas för Basicprogram.

När man programmerar på en dator så brukar man använda någon form av editor för att skriva och rätta i programmet. Det kan liksom på ABC-80 vara en editor speciellt gjord för det programspråk man programmerar i. Det kan också vara en generell texteditor användbar för flera programspråk och godtyckliga textfiler. I dessa fall består källkoden till programmen av en textfil. Man använder editorn såväl för att skriva in programmet som för att göra tillägg, ändringar och rättelser.

En sådan generell editor kan även användas för att bearbeta godtyckliga textfiler. Det kan vara texter av mest skiftande slag såsom brev, skrivelser, notiser, artiklar, tabeller, rådata för input till dataprogram etc. Man kan exempelvis använda en editor för att skriva in snygga rubriker i tabeller som framställts med ett dataprogram (det går ofta snabbare på det sättet än att lägga ned möda på att progammera utskrift av snygga rubriker). Man kan givetvis också använda editorn för att rätta siffervärden i tabellen samt för att korrigera data som registrerats i textfiler. Kort sagt en editor användes när helst man vill skriva text, göra ändringar i eller tillägg till en befintlig text.

Att läsa en textfil

Om bara vill kunna läsa en kortare textfil, på kassett eller flexskiva, går det kanske både enklare och snabbare att använda programmet VISA som fanns på ABC-kassett nr 1. Gör RUN VISA och ange filnamnet på den fil du vill läsa. (Om filen ligger på kassett räcker det att ange CAS: som filnamn, därvid läses den fil som först påträffas på bandet.) Om man trycker ner en tangent så visas första raden i filen, varje ny tangenttryckning visar ytterliggare än rad. Det kan vara lite knepigt i början när man läser filer på kassett, om man väntar för länge med att trycka tangent så spårar programmet ur. Det beror på att bandspelaren när den väl startat läser raderna i den takt dom påträffas på bandet. Till slut blir det spill när CAS-buffertarna blir överfulla. Ett annat problem är att man aldrig kan gå tillbaka i texten till något som stod tidigare i filen. Då måste man i stället börja om läsningen från början. Dessa problem slipper man med TV-editorn.

Uppstart av TV-editorn

TV-editorn består av tre delar, filerna TV.BAC, TVSUBR.ABS samt TVMAIN.BAC eller motsvarande programfiler som ersätter dessa. Man behöver bara initiera det första programmet med RUN TV, sedan skall resten ske automatiskt. Det förutsätter att alla programmen finns tillgängliga på någon av diskarna. TV läser in de maskinkodsrutiner som editorn arbetar med från filen TVSUBR. ABS och lagrar dessa fr o m adress 63488 och gör sedan CHAIN till TVMAIN.BAC. Därvid kommer texten "Reenter with list" upp på skärmen.

Efter avslutad inladdning blankas skärmen och editorns sk markör, de tre tecknena <*> i ursprungsversionen, i senare modifieringar det enda tecknet "ruta" = ASCII 127, skrivs ut i övre vänstra hörnet. Denna plats motsvarar början av den text editorn kommer att arbeta med. Samtidigt skrivs en asterisk följt av en blinkande cursor i början på rad 22. Editorn är nu klar för användning.

Kassettversionen

En av de stora fördelarna med TV-editorn är att den går utmärkt att använda med enbart kassettutrustning. (Den här texten har jag skrivit och redigerat på en ABC-80 som jag tagit med mig under en semestervecka, visserligen med extraminne och 80 kol bildskärm, men ändock med enbart kassettutrustning). Editorn initieras med RUN TVCAS, som gör CHAIN till CAS:TVSUB.BAC vilken lägger in maskinkodsrutinerna med Poke. (Genom att läsa in de värden som pokas in från DATA- satser i stället för som ofta sker lägga dem direkt i POKE-satserna sparar man mycket utrymme vilket gör programmet kortare och snabbare att läsa in.) TVSUB gör sedan CHAIN till CAS: TVMAIN.BAC. För att inläsningen skall gå så snabbt som möjligt bör man lägga de tre filerna direkt efter varandra på ett band. Det är viktigt att filerna lagras under just det namn som tidigare program ger CHAIN till annars spolar bandet bara förbi.

Anm. Kassettversionen av TV-editorn kan också användas på flexskiva. Eftersom programmen gör CHAIN till enheten CAS: kan man få Error 21 när programfilerna inte hittas på bandet. Gör då RUN TVSUB resp RUN TVMAIN om filerna ligger på flexskiva. Att använda kassettversionen för att starta upp TV-editorn rekommenderas för den som inte får den andra versionen att fungera, det gäller bl a om du har DataDisc 88 med ett annat DOS. Då kan det vara praktiskt att ta bort enhetsnamnet CAS: ur de programrader som gör CHAIN i TVCAS resp TVSUB. När TV-editorn väl har laddats in fungerar båda versionerna lika.

Så här börjar du

När cursorn står och blinkar efter asterisken på rad 22 är TV-editorn klar att ta emot dina order som här kallas kommandon. Ett kommando består vanligtvis av en bokstav eventuellt föregånget av annat tecken eller siffror.

Låt oss börja med inläsningskommandot som ser ut så här ;Y alltså semikolon följt av bokstaven y (stor eller liten bokstav spelar ingen roll i kommandon). Skriv alltså ;Y och tryck sedan på den röda högerpiltangenten två gånger. Denna tangent användes när man vill beordra utförandet av det eller de kommando man skrivit in efter asterisken i början på rad 22. Vid nertryckning av denna tangent skrivs tecknet sol, ASCII 36, ut. Två nedtryckningar innebär verkställighetskommando. Flera kommandon kan skrivas efter varandra i en lång följd. De verkställes först när man tryckt två högerpilar i följd. Högerpilen har även en annan funktion som vi återkommer till.

När du tryckt tangenten för andra gången svarar editorn med frågan "Input file:". Ange filnamn exempelvis TVEDIT.REM som låg på ABC-kassett nr 3 och tryck Return. Nu läses hela filen in och om inläsningen av gått bra kommer skärmen efter en stund att visa de 21 första raderna i filen med markören först på första raden. Längst ner står det --more-- för att visa att filen innehåller flera rader. Markören användes för att markera var i filen man befinner sig, det behövs när man vill skriva in ny text på viss plats eller ta bort viss del av befintlig text.

Markörförflyttningar

TV-editorn arbetar inte med radnummer som editorn för Basicprogram gör. Den arbetar i stället med en markör för att hålla reda på var man befinner sig i texten. Den som kan editorn TECO på DEC-10 och PDP-datorer kommer att känna igen flera av kommandona.

Vissa kommandon utföres omedelbart, utan särskilt verkställighetskommando. Med höger och vänstarpilarna flyttar du markören framåt och bakåt i filen, med RETURN-tangenten flyttar du markören till början av nästa rad och med tangenten "tyskt Y", ASCII 94, flyttar markören till början av föregående rad. Vill du läsa den text som kommer efter --more-- tryck mellanslag så "bläddrar" du frammåt i filen. Du kommer tillbaka till början igen genom att trycka vänsterpil. CTRL-B placerar markören mitt på den sida som visas och CTRL-L centrerar en textsida kring markören utan att flytta dess läge. Du kan alltid flytta markören till början av filen med kommandot J men kommandot utföres först sedan du givit verkställighetsorder.

Du kan läsa av markören läge genom kommandot .= alltså punkt följt av likhetstecken. Det siffervärde som skrivs längst ned anger markörens plats räknat från filens början. Vill du se hur många tecken hela filen innehåller ger du kommandot Z= samt verkställighetskommando.

Det finns ytterligare två kommandon som flyttar markören nämligen C och L. C och L kan föregås av ett talvärde här betecknat med n. nC flyttar markören n positioner framåt i filen och nL flyttar markören n rader framåt alltså samma resultat som att trycka n gånger på RETURN-tangenten. Förflyttningen utföres först efter verkställighetskommando. När talvärdet på n skall vara 1 räcker det att skriva C resp L, vilket tolkas som 1C resp 1L. Ett radslut i filen består av de två tecknena <CR> och <LF>, ASCII 13 resp 10. (Prova vad som händer när markören står i slutet av en rad och du ger kommandot C. Då ser du att de två tecknena <CR> och <LF> skrivs ut som CTRL-M och CTRL-J, med CTRL representerat av tyskt Y.) Du återställer texten genom att trycka RETURN. Som talvärde på n kan storheten Z, antalet tecken i filen användas. ZC flyttar markören till slutet av filen, platsen omedelbart efter sista tecknet.

Borttagning av text

Det finns två kommandon D för delete och K för kill som kan användas för ta bort viss del av texten i en fil. nD tar bort n tecken omedelbart efter markören. Tänk på att radslut i filen består av två tecken! nK tar bort n rader från markören räknat. Kommandot ZD tar bort all text som står efter markören. Det sammansatta kommandot JZD tar bort all text i filen och är mycket användbart för att snabbt tömma editorn på gammal text innan man läser in en ny textfil. Tekniken att ta bort text är att först placera markören omedelbart framför den text som skall bort och därefter använda D och K-kommandona. Är du osäker ta hellre bort för lite och upprepa sedan tills du är nöjd.

Inskrivning av text

Med kommandot I (I för insert eller insättning) skriver man in ny text med början på den plats markören står. Det går till så att man omedelbart efter I skriver den text man vill lägga in. Det går bra att skriva flera rader på en gång. Avsluta sedan texten med högerpil som "ekas" med soltecknet. När du trycker högerpil för andra gången förs texten in på plats och markören placerar sig omedlebart efter den just inskrivna texten. Sedan kan du fortsätta med annat kommando eller ett nytt I följ av ny text.

Den text du skriver efter I lagras temporärt i en särskild inskrivningsbuffert som kan innehålla max 500 tecken. Skulle du råka överskrida den gränsen så gör det inte så mycket. När bufferten blir full töms texten automatiskt över på plats i huvudtexten. Du kan råka tappa något av de sist inskrivna tecknena.

Observera att du inte får glömma det inledande I-kommandot när du vill skriva in text. Annars tolkar editorn det du skriver som en kommandosträng som den sedan försöker utföra när du ger två högerpilar. Det är då stor risk att något blir tokigt.

Under inskrivningen av text kan du använda vänsterpilen för att ta bort felinslagna tecken, även efter det du tryckt Return för att börja ny rad. Med CTRL-X kan du ta bort hela den sist inskrivna raden. Du kan ta bort ytterliggare en rad med nytt CTRL-X. Tänk på att inte skriva för långa rader! TV-editorn accepterar visserligen i princip hur långa rader som helst men du får problem när du överför texten till fil på flexskiva eller kassett eftersom ABC-80 inte accepterar längre rader än 117 tecken exkl <CRLF> vid inläsning från fil. Tänk också på att även den sista raden i en text måste avslutas med RETURN, annars får man inte något <CRLF> i slutet av raden villket medför att man tappar den raden vid återläsning.

Skriva texten på fil

När du använder editorn ligger all texten lagrad i RAM-minnet. Skulle du råka ut för ett kortvarigt strömavbrott förlorar du allt vad du åstadkommit om du inte lagrat över textinnehållet på fil. Det gör du med kommandot ;S alltså semikolon följt av S och två högerpilar. Då får du frågan "Output file:" som du besvarar med lämpligt filnamn och Return. Då skrivs filen ut på flexskiva resp kassett. Du bör göra det till en god regel att med jämna mellanrum lagra över det du skrivit eller redigerat på fil för att inte tappa någon information. När du skrivit på fil så kan du direkt fortsätta att editera vidare, därför bör det inte innebära något större avbrott att då och då skriva texten på kassett eller flexskiva.

Printerutskrift

Om du har en printer ansluten kan du texten utskriven genom att ange filnamn PR: eller V24: beroende på vilket enhetsnamn du använder för din printer. Om du använder en särskild printerrutin t ex ABCV24 måste denna laddas in innan du startar upp TV-editorn. TV-editorn är mycket lämplig att använda när du vill göra printerutskrifter av textfiler med enbart kassettutrustning. Det beror på att hela filen läses in på en gång och du slipper att omväxlande läsa från kassett och skriva på printer. (Vissa printrar bl a min Anadex DP 8000 bär sig underligt åt när man läser från kassett samtidigt som printern är inkopplad.)

SÖkning och textutbyte

Det som gör TV-editorn särskilt kraftfull är kommandona S och R, S för search eller sökning R för replace (utbyte). Om man vill söka efter en textsträng t ex "Basicprogram" så ger man kommandot SBasicprogram<högerpil><högerpil>. Därvid letar editorn upp den första förekommsten av textsträngen "Basicprogram". All sökning sker från markören och framåt i texten. Skulle den inte finna någon match skrivs "Search failed: Basicprogram" ut. Framför S kan man skriva ett siffervärde t ex 2. Därvid letar editorn efter den andra förekomsten av textsträngen i stället. Den textsträng man söker får bestå av högst 20 tecken. När man vill söka vidare efter samma textsträng räcker det att ge kommandot S följt av två högerpilar, textsträngen behöver inte upprepas igen. Sökningen sker i exakt mode, stor eller liten bokstav är här en väsentlig skillnad. Stava rätt editorn söker slaviskt efter det du skrivit.

R-kommandot användes när du vill byta ut en viss textsträng mot en annan. Du skriver:
Rgammaltext<högerpil>nytext<högerpil><högerpil>
för att få den först påträffade förkomsten av "gammaltext" utbytt mot "nytext". Inget hindrar av strängen "nytext" är tom, då tas gammaltext bort utan att ersättas mot någon ny. Genom att skriva ett talvärde t ex 5 före R så ersättes "gammaltext" med "nytext" på de 5 först påträffade ställena. Vill man byta ut texten på samtliga platser den förekommer så räcker det att skriva ett tillräckligt stort tal framför, det gör inget om talet är för stort. Denna möjlighet gör R-kommandot mycket kraftfullt t o m så att det kan vara färädiskt att använda det om man inte tänker sig för. Prova t ex kommandot:
J1000Ren<högerpil>två<högerpil> <högerpil>
och se efter på vilka underliga ställen "två" hamnar.

Det går mycket bra att editera Basicprogram med TV-editorn. Basicprogrammet måste vara i BAS-format dvs ha lagrats på fil med LIST. Då kan R-kommandot kan vara mycket användbart. Ett exempel: Man har ett program som skriver en hel del text på skärmen men vill i stället ha utskriften på printer. Då kan man försöka med följande kommando:
100R;"<högerpil>;§L%,"<högerpil> <högerpil>
för att få texten skriven på filen L% i stället för på skärmen. Man får givetvis lägga till en rad där filen först öppnas.(§ = brädgård)(r)

Q-register

Ibland vill man kunna flytta eller kopiera ett textstycke från ett ställe till ett annat. Då kan man använda det sk Q-registret som finns med i TVMAIN2.BAC på ABC-kassett nr 3. Med kommandot nO lägger man in n st rader från markören räknat Q-registret. Vill man ta bort texten på den ursprungliga platsen ger man även kommandot nK. Sedan flyttar man markören till den plats där man vill ha in texten som nu ligger i Q-registret och ger kommandot G vilket lägger in texten på den nya platsen. Q-registret (variabeln Q$) är dimensionerat till max 2000 tecken. Är du tveksam om det räcker så använd kommandot .= (se ovan) och läs av markörens position dels vid starten och dels vid slutet av det textstycke du vill flytta. Ett litet exempel på användning av Q-register finns i filen TVEDIT.REM på ABC-kassett nr 3.

Återinträde med List

Det finns en mycket nyttig funktion inbyggd i TV-editorn. Det är rad 10 i TVMAIN som gör att man kan återinträda i editorn med bibehållen textinformation. Om man stoppat med CTRL-C och sedan ger kommandot LIST så kommer man tillbaka in i editorn. I versionerna TVMAIN och TVMAIN2 fungerar detta endast för ABC-80 med checksumma 11273.
Om du har checksumman 9913 eller 10042 kan du få det att fungera genom att ändra rad 10 på följande sätt: Gör LOAD TVMAIN eller TVMAIN2. Gör sedan LIST-9. Står det några REMsatser så ta bort dom. Skriv sedan B=PEEK(65053)*256+PEEK(65052)+17 <RETURN>. Skriv därefter PRINT PEEK(B) <RETURN>. Om du gjort rätt skall du få värdet 97. Skriv sedan POKE B, 95 <RETURN>. Gör därefter SAVE TVMAIN resp SAVE TVMAIN2. Sedan kan du ladda TV-editorn på nytt fast nu med de nya versionen av TVMAIN.
(Rad 10 ser i kompilerad form ut så här:

24 10 0 135 19 133 203 64 10 58 42 254

167 202 85 13 195 97 13 203 64 0 187 13

Första byten 24 anger att raden innehåller 24 bytes, de två följand byten anger radnummer 10. Nästa två byte 135 19 gör att man vid exekvering hoppar över följande 19 bytes dvs resten av raden. Längre fram står 195 97 13 som vid återlistning medför hopp till adress 3425 dvs 13:97. Har du annan checksumma -(9913,10042)- skall hoppet göras till 3423 dvs 13:97. 97 skall alltså bytas ut mot 95 !)

I kommande versioner av TVMAIN som kommer på ABC-kassetter kommer vi att lägga in en rad 11 avsedd att ersätta rad 10 för de som har andra checksummor än 11273. Rad 11 gör ingen skada (annat än att texternas maxlängd minskar med 24) om den ligger kvar vid checksumma 11273, men rad 10 måste tas bort om du har checksumman 9913 eller 10042.

Övning ger färdighet

Det bästa sättet att lära sig TV-editorn är genom praktisk övning. Det är min förhoppning att denna artikel skall hjälpa dig igång om du inte redan gjort det. Använd de enkla kommandona innan du blir van. Du kommer snart att upptäcka att du med TV-editorn fått en ordbehandlingsmöjlighet på din ABC-80. Har du dessutom skaffat dig 80 kol bildskärm så lär du nog komma att uppskatta TV-editorn ännu mer.

Gunnar Tidner

PROGRAMMET EXTDEL

Den här artikeln stod i förra numret av bladet,men eftersom källkodslistan till programmets maskinspråksrutiner blev konstigt utskrivna,införs artikeln på nytt med vissa ändringar.

På ABC-kassett nummer 2 finns bland alla program ett som tar bort rader ur ett annat program. Jag har döpt det till 'EXTDEL'. EXTDEL syftar på funktionerna extract & delete.

Programmet som jag visade vid ett möte i Vidängssalen utnyttjas några maskinspråkssnuttar, som jag skall beskriva. Först några korta ord om programmets handhavande. Ladda in det program som är föremål för redigering. Ladda därefter in EXTDEL med MERGE och RUN. Programmet utnyttjar s.k. blockmoveinstruktionen LDIR, vilket är en förkortning av: Load Decrement, Increment and Repeat. Denna instruktion använder HL och DE registren som pekare varvid HL pekar på källan och DE på destinationen. När instruktionen utförs laddas position DE med innehållet i position HL innehållet i DE och HL ökas med 1 BC minskas med 1 - detta upprepas tills BC är 0. Storleken av det minnesblock som flyttas anges alltså av BC-registrets innehåll.

Algoritmen för programmet

Vi kan tänka oss två fall, som motsvarar programmets funktioner. Vid extract skall ett minnesblock flyttas ner till BOFA. Vid delete skall två minnesblock flyttas så, att de kommer att ligga konsekutivt från BOFA och uppåt.

De data som rutinerna kräver är storleken av de minnesblock som skall flyttas och adressen till byte 1 i vart och ett av dessa.

Frågan är då hur tillför vi dessa data till vår maskinspråksrutin? Den lösning jag valt finns i programrad 65090 och i källkod nedan.

```
ORG -128
LD (-2),DE
RET
LD (-4),DE
RET
LD (-6),DE
RET
```

Den som upprörs i sitt innersta av att jag anger adresser decimalt kan lätt räkna om dem till HEX. Rutinen ovan är egentligen tre rutiner för lagring av tre tvåbytestal i pokearean. Dessa tal kan skickas som argument i en CALL-instruktion. I Basic-programmet beräknas lätt storleken hos det (de) minnesblock som ska flyttas. Jag kommer inte att ytterligare beskriva hur kommunikationen mellan Basic och maskinspråk går till i detta fall.

Nu kommer vi till själva move-rutinen. (I programmet på rad 65310 - extract)

```
ORG -100
LD BC,(-2)
EX DE,HL
LD DE,(BOFA)
LDIR
LD A,1
LD (DE),A
RET
```

Programmet avslutas med att en etta läggs in efter det flyttade blocket. Vid delete blir det hela mer komplicerat. Flyttningen måste ske i två steg med två LDIR. Här följer källkoden till raderna 65370-65380.

```
ORG -100
EX DE,HL
LD DE,(BOFA)
LD BC,(-2)     ; STORLEK BLOCK 1
LDIR
LD HL,0
ADC HL,DE
LD DE,(-6)
EX DE,HL
LD BC,(-4)     ; STORLEK BLOCK 2
LDIR
LD A,1
LD (DE),A
RET
```

Problem

När EXTDEL används kan det bli problem om man gör MERGE direkt efter. Det beror på att programslutpekaren inte ställs om av EXTDEL. Denna ställs dock om av kommandot CLEAR, RUN med flera. Om man önskar kan man lägga till några instruktioner som sätter pekaren på adress 65056 lika med programmslut.

Eftersom HL-registret kommer att peka på den sista byten i programmet sedan blockmove gjorts så behöver HL ökas med ett och detta innehåll läggas på adress 65056. Detta sker med:

```
INC HL
LD (65056),HL
RET
```

Den ändring man behöver göra i programmet är att ta bort den avslutande 201 (RET) på raderna 65310 och 65380 och skriva in 35,237,99,32,254,201 i stället.

Kjell-Åke Johansson

```
100 REM Basicrad
110 REM Björn Gustavsson 1981-06-
    29
120 DIM A$=200%,B$=50%,C$=50%
130 ; CHR$(12%)," BASICRAD BG-81
    " : ; ,CHR$(151%)STRING$(15%,
    35%)
140 POKE 65408%,33%,64%,254%,231%
    ,205%,144%,24%,33%,0%,0%,208%
    ,111%,201%
150 ; : ; "Mata in en BASIC-rad:
    "; : INPUTLINE A$ : ; : ;
160 ; CHR$(12%); : ;
170 ; 'BASIC-rad:'
180 ; : ; A$
190 J%=65088%
200 FOR I%=1% TO LEN(A$)-1%
210 POKE J%,ASC(MID$(A$,I%,1%)) :
    J%=J%+1%
220 NEXT I%
230 Z%=CALL(65408%)
240 IF Z% ; 'BASIC-error:'Z%-128%
    : GOTO 150
250 R%=PEEK(65208%)
260 ; 'Kompilerade radens längd:'
    R% : ;
270 B$='' : C$=''
280 FOR I%=65208% TO 65207%+R%
290 A%=PEEK(I%)
300 B$=B$+SPACE$(4%-LEN(NUM$(A%))
    )+NUM$(A%)
310 IF A%<32% OR A%>127% A%=38%
320 C$=C$+'    '+CHR$(A%)
330 IF LEN(B$)>=40% ; B$+C$ : B$=
    '' : C$='' : GET A$
340 NEXT I% : IF LEN(B$)>0% ; B$+
    SPACE$(40%-LEN(B$))+C$
350 ;
360 GOTO 150
```

BASICRAD

Programmet BASICRAD som presenteras här används när man vill ta reda på hur en BASIC-rad lagras i intern form. Man matar in en rad, som sedan görs om till internformat och visas på skärmen. Använt tillsammans med Arne Stockmans listning av programvaran i ABC80 kan man lära sig en hel del.

När programmet startas frågar det om en BASIC-rad. Mata in en rad och tryck <RETURN>. Nu visas första delen av raden i internformat. Tryck ned en tangent för att mata fram fler rader. Innehållet i varje byte skrivs ut, och under detta motsvarande tecken. Om tecknet är ett kontrolltecken (ASCII: mellan 0 och 32 ELLER 127 och 255) skrivs i stället "&".

Programmet fungerar på följande sätt: Raden läses in på vanligt sätt och pokas sedan in i inmatningsbufferten, som börjar vid adress 65088. Sedan anropas följande enkla assemblerrutin:

```
LD HL,254:64    ;Adress till
                ;inmatningsbuffert
RST 32          ;Ignorerar in-
                ;ledande mellanslag
CALL 24:144     ;BASIC-tolkens
                ;kompileringsrutin
LD HL,0:0       ;Sätt felkod till
                ;inget fel
RET NC          ;RETurn till BASIC-
                ;programmet om inget fel

LD L,A          ;FEL! Ladda HL
                ;med felkod
RET             ;RETurn till BASIC-
                ;programmet
```

Adresserna är skrivna som Arne Stockmans disassembler skriver adresser, dvs 254:64 betyder 254*256+ 64=65088. Assemblerrutinen anropar BASIC-tolkens rutin för kompilering av inmatade rader till internformat. Den kompilerade raden lagras från adress 65208 och uppåt. BASIC-programmet **peek**-ar då fram en byte i taget och skriver ut.

Björn Gustavsson

(Vid checksumma 9913 och 10042 ändras CALL 24:144 till CALL 24:142)(red)

ANNORLUNDA STRÄNGHANTERING

eller

HUR MAN TROLLAR BORT BILDER OCH MASKINSPRÅKSPROGRAM SÅ ATT DE KNAPPAST SYNS

eller

VAD GÖR ABC80 MED PROGRAMTEXTEN?

Bernt Figaro/Sven Wickberg

Bernt Figaro i Örebro är en av ABC-bladets flitiga korrespondenter. Han har sänt in flera flexskivor fulla av program och texter. Både den inbitne ABC80-entusiasten och den blyge nybörjaren i genren kan ha intresse av de synpunkter han för fram.

Originalen har dock bedömts vara för långa och något för invecklade att ta in i bladet. Med Bernts medgivande görs här ett sammandrag med några av de intressantaste avsnitten. Den som tänder extra på det här kan kontakta Bernt direkt för att få mera kött på benen.

BASIC KONTRA MASKINSPRÅK

De flesta skriver sina program i BASIC, men för vissa tillämpningar bör eller måste man skriva i maskinspråk. Ett sådant maskinprogram består av en rad siffror, som måste stoppas undan någonstans i arbetsminnet och som sätts i funktion genom Z=CALL(X), där X är numret på den minnescell (adressen) där sifferraden börjar.

Vi tar inte upp hur man gör maskinprogram - det är en historia för sig. Men om man nu har en programsnutt man behöver använda, fordras med normala metoder också en programsnutt i BASIC som lägger in maskinprogrammet i minnet. Denna BASIC-del är under alla förhållanden ganska skrymmande, och tar väl mycket plats både i programutskrift och arbetsminne. I slutändan av denna artikel skall vi rationalisera bort den.

Men låt oss börja från början:

HUR LAGRAS MITT PROGRAM I ABC80?

När vi laddar in ett BASIC-program i vår dator, skrivs den in i arbetsminnet med början i den adress som på dator-jarjong kallas BOFA (Beginning Of File Area, eller något i den vägen). I en vanlig 16 K ABC80 är BOFA normalt 49152, med utökat minne till 32 K vanligen 32768.

Man kan genom diverse konstgrepp "höja golvet", t ex för att reservera plats för maskinspråk, men det hoppar vi också över.

Apparaten håller själv reda på BOFA i minnesscellerna 65053 och 65054, så vill man vara helgarderad kan man i stället för rad 90 i nedanstående program skriva

P=PEEK(65052)+256*PEEK(65053)

så vet man säkert att P=BOFA.

Hur det ser ut i arbetsminnet när vi skrivit in ett program kan vi lätt ta reda på med följande lilla loop:

```
90 P=49152 : REM...BOFA
100 A=PEEK(P+I)
110 ;P+I;I,A,CHR$(A)
120 GET W$
130 I=I+1
140 GOTO 100
```

P är begynnelseadressen. På rad 100 läser man vad som står i den, och på rad 110 skriver man ut adressen, antal steg framåt (I), adressinnehållet och motsvarande tecken enligt ASCII-koden.

Med rad 120 stegar man fram utskriften adress för adress.

(För ASCII-koder under 32 visar skärmen ingenting, eftersom de inte motsvaras av något tecken. 32 är ju mellanslag, som inte heller syns separat.)

Syftet med denna operation är att analysera olika programdelar som placeras i rad 10, t ex

```
10 A$='ABC'
```

Kör nu programmet och stega fram till I=16, vars ASCII-kod skall vara 13. Det ser då ut så här (om man utelämnar bl a adressnumret):

I	A		Anm.
0	17		Antal steg i raden
1	10		Radnummer
2	0		(två bytes)
3	131		ABC80-kod
4	194		"
5	242		"
6	65	A	" (strängnamnet)
7	132		"
8	131		"
9	203		"
10	39		"
11	3		" (antal tecken i strängen)
12	65	A	Strängens innehåll
13	66	B	"
14	67	C	"
15	184		Kod (sträng slut)
16	13		Programrad slut

Som synes tar den enkla programraden upp hela 17 minnesceller!

För den intresserade kan det löna sig att systematiskt gå igenom olika programinstruktioner och skrivrutiner och se hur de ser ut. ABC80-koderna kan ibland vara mycket kortare än deras utskrifter på skärmen. Se t ex på PRINT, DATA, STRING$(10%, 45%).

Beträffande det sista kan påpekas, att man vinner åtskillig plats i minnet på att använda heltal där så kan ske, även i uttryck som CHR$(), MID$() osv, där annars omvandling till heltal sker automatiskt vid körning.

Jämför (sätt in i rad 10) P=1 och P%=1%. Övertygad?

Vad har man nu för nytta av ovanstående kunskaper?

Jo, man kan lagra informationer i sina programrader på annorlunda sätt.

Ta A$ som exempel. Från tangentbordet kan vi skriva in bara vissa tecken i en teckensträng. Tecken med ASCII-kod mindre än 32 eller större än 126 går inte, och inte heller grafik-tecken. Normalt har vi heller inte nytta av dem, eftersom teckensträngarna skall producera text.

Men om man nu med stort besvär har komponerat en bild på skärmen, som man vill använda också i andra program? Eller om vi har en maskinspråksrad som vi gärna ville överföra till fler program?

Då är det perfekt att lagra alltihop i en lämplig teckensträng. Denna kan sedan sparas och MERGAS med vilka program man vill.

ETT EXEMPEL

Gör följande modifikation av det tidigare uppgivna programmet:

```
10 A$='*****'
95 ;"TITTA/SKRIVA (T/S)";:GET W$
97 ;W$:IF W$="S" THEN 300
300 FOR J=0 TO 4
310 READ A
320 POKE P+12+J,A
330 NEXT J
340 ;"DATA INSKRIVNA"
350 GOTO 95
400 DATA .....
```

Vi måste passera rad 90 som ger oss BOFA (P=49152) men sedan kan vi välja att hoppa till den nya rutinen 300. Vad händer där? Jo, vi läser ur DATA-raden 5 siffervärden (ett i taget) och petar in dem med POKE i A$ med början i den position där teckenvärdena i A$ börjar.

När det är gjort får man beskedet "DATA INSKRIVNA" och så är man tillbaka på valraden 95. Väljer man nu T kan man som förut stega sig igenom minnesadresserna och se att de värden man har bestämt har hamnat på rätt plats.

FÖRSTA PROVET

Man skall inte vara för originell i första omgången, så låt oss ge DATA-satserna värdena 65,66,67,68,69, dvs ASCII-värdena för ABCDE.

Kör programmet och se att just dessa värden går in. Gå ur programmet och skriv LIST 10 så ser vi att A$='ABCD'.

Det var inget märkvärdigt, utan snarare ett komplicerat sätt att skriva en sträng på.

ANDRA PROVET

Nu kommer det intressanta. Ändra DATA-satserna till: 151,100,101,102,103 och kör.

Vi finner värdena på plats, men när vi tittar på rad 10 finner vi att A$ är fullt med grafiktecken! ASCII-värdet 151 sätter efterföljande värden i grafisk mod. Vi har demonstrerat en metod att skriva "oskrivbara" tecken i strängen.

Bernt har intresserat sig mycket i för grafiska representationer och experimenterat fram en rad hjälpprogram med vilka man relativt enkelt kan komponera en graf på skärmen och stoppa undan den i en strängvariabel. Bilden framkallas med PRINT A$.

TREDJE PROVET

Låt oss vara nyfikna. Vad händer med tecken 1-31 i DATA-satserna. Du får själv expermentera med dem, det ger oväntade resultat. Här är några.

Tecken 7 ger "pip" - som vanligt. 10 matar upp texten en rad, och 13 för texten till radbörjan. Med lämpliga doser kan man alltså ge en "rak" teckensträng viss formatering.

Tecken 12 suddar skärmen, det visste vi också. Lägger man 12 sist i en teckensträng och kör LIST, får vi aldrig se strängen, eftersom den suddar ut sig själv! Om man sätter en sjua i den utsuddade delen, hör man att den är kvar.

Man finner att också tecknen 17-23 gör efterföljande tecken grafiska, och att tecknet 1 tar bort grafik, precis som 135.

Tecknen för 7,10,12 och 13 kommer vid listning eller utskrift av A$ att synas endast genom sin verkan. De tar ingen synlig plats i utskriften. Prova med bara sjuor i DATA-raden. Övriga tecken skrivs som mellanslag. På vissa printrar kommer inte heller grafiken med i utskriften. Men tecknen finns där i alla fall.

FJÄRDE PROVET

Nu har vi kommit fram till det som var målet för övningen: man kan också lagra sina maskinprogram i strängen.

Arbetsgången är exakt densamma. Maskinprogrammet har du utarbetat på vanligt sätt. Placera in det i DATA-raden, du kanske behöver flera. Gör A$ så lång som behövs. Du kan med KOD-programmet lätt se adressen där strängvärdena börjar och slutar, och du får modifiera inläsningsdelen därefter.

Saken är klar, maskinspråket inläst: vad händer nu?

Nu kommer det roliga: Ta bort allt utom raden med A$! Det ensliga programmet med bara A$=en rad tecken, mer eller mindre synliga och läsliga, kan sparas och MERGAS in i andra program. Av praktiska skäl bör den stå först. Då vet man exakt var maskinspråket börjar. Man måste ju ha raden Z=CALL(...) för att få i gång programmet.

Detta är mycket praktiskt. Man spar bra mycket minnesutrymme och man behöver inte POKEa på något annat ställe i arbetsminnet. Instruktionerna i vår sträng följer snällt med huvudprogrammet och finns alltid på sin kända plats.

På samma sätt gör man med andra sparade strängar.

Men en sak får man INTE göra:
editera dessa specialsträngar! Försöker du med det blir de helt förstörda, eftersom BASIC-tolken inte klarar annat än de vanliga skrivtecknen och genast tvättar bort alla andra.

Så en slutanmärkning: varför jobba med textsträngar, som behöver så mycket plats som 10 platser för bara strängdefinitionen? Varför inte lagra i REM-, DATA- eller PRINT(;)-satser i stället?

På det svarar Bernt att REM och DATA är opålitliga. Svens kommentar är att efter REM och DATA lagras allt i textformat, utan att transformeras till ABC80-kod. Kanske har det något med resultatet att göra.

Däremot kan man lagra i ;-satser utan besvär. Ett problem i sammanhanget är dock att när man kör sitt program kommer PRINT-satserna också att exekveras och därmed synas på skärmen, vilket man kanske inte är intresserad av.

Slutligen har A$ fördelen att vid behov kunna tas fram, t ex som en bild, med ;A$.

Samarbete ger bättre ABC80 för skolan

OWOCO AB har på uppdrag av LIBER Läromedel AB och LUXOR Datorer AB tagit fram ett programmeringshjälpmedel som heter **SCHOOLAID**(R). I detta samarbete står OWOCO AB för ide och konstruktion samt produktion och LIBER AB marknadsför produkten på utbildningsmarknaden.
SCHOOLAID är det senaste tillskottet i OWOCO's serie "AID'ar" med SMARTAID typ 2 som förebild med dess grundläggande funktioner (avancerad bildskärmseditor, automatisk radnumering, tar bort flera rader i en följd, listar alla variabler i ett program och startar programmet från valfritt ställe däri) och dess enkla utanpåliggande montering, som har nykonstruerats för att uppta mindre plats mellan tangentbord och bildskärm.

Ur OWOCO's SMARTAID 3 och SUPERSMARTAID (den senare som snart levereras) har funktionerna till **SCHOOLAID** sammanställts för att motsvara önskemålen man haft från LIBER och skolsidan.
Bl.a. följande exempel:

Temporärt stopp vid programkörning utan att programkörningen avbryts, bra exempelvis vid felsökning.

Med ett kommando kan användaren få fram en hjälplista över **SCHOOLAID**'s och ABC80's kommandon.

Möjlighet att spara valda programdelar på ex. skrivare, söka efter en variabel, radnummer eller valfri text och få de radnummer där dessa finns på. Byta variabelnamn på en redan befintlig variabel. Återuppta programkörningen efter det att programmet avbrutits, med alla variabelvärdena bibehållna.

Hämta tillbaka eller "rädda" program som har tagits bort oavsiktligt med RESET eller NEW.

Lista minnesinnehållet på bildskärmen i form av decimaladress, decimalt innehåll och ASCII-format, vilket underlättar vid närmare studier av datorns minnesinnehåll. Få systemtiden utskriven i timmar, minuter och sekunder som också enkelt kan nollställas för enklare tidtagning (tiden kan även enkelt fås utskriven i ett program).

Kommando som visar ett bibliotek över innehållet på diskett. Information om systemet kan erhållas med ett kommando där bl.a. visas programmets längd, kvarvarande minnesutrymme, totala minnesåtgången av ett program efter körning, systemvariablerna som pekar på olika adresser ex. BOFA, EOFA, HEAP mm. Bildskärmens innehåll kan även "dumpas" på en skrivare.

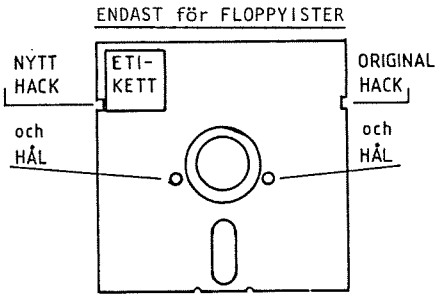
SCHOOLAID har dessutom en egen printerrutin vilket möjliggör att använda den och ABC80 tillsammans med en skrivare, utan att behöva ladda in printerrutin i arbetsminnet eller ansluta sig till en flexskiveenhet.

SCHOOLAID är även anpassad till MULTI-USER system och olika tolkar på ABC80, SCHOOLAID kan därmed användas både till äldre och nyare ABC80. **SCHOOLAID** anslutes till ABC80's busskontakt samt skruvas fast, den är utrustad med egen busskontakt för anslutning av flexskive-enhet etc.

Till **SCHOOLAID** finns också ett instruktionshäfte i A5-format.

LIBER räknar med att de flesta ABC80 i skolorna kommer att utrustas med **SCHOOLAID** då detta innebär en avsevärd förbättring av ABC80's kapacitet, dessutom kommer en utökning till 32k minne att erbjudas.

Utförligare information kan fås från LIBER DATORER Telefon 08/739 90 00.



Vi hoppas få rum i nästa nummer med texten till denna bild.

Mera om TV-editorn

Sedan artikeln om TV-editorn publicerades i ABC-bladet 1982:1 har vi hört TV-editorn lovordas av många entusiastiska användare och vi har även fått flera brev med påpekande och frågor. På ABC-kassett nr 4 ligger en ny version av TVMAIN kallad TVMAIN3. Den innehåller en del nya kommandon och har fått några tidigare otympligheter borttagna. De viktigaste nyheterna är:

Negativa argument kan användas i vissa kommandon
Makrokommando
Kontroll att Q-registret inte spiller över
Återinträde med LIST fungerar även med checksummor 10042 och 9913

De olika versionerna

På ABC-kassetter har distribuerats versionerna TVMAIN, TVMAIN2 och TVMAIN3. Behövs flera versioner?

De olika versionerna skiljer sig åt beträffande tillgängliga kommandon och buffertstorlek. /ill du jämföra de olika versionerna kan du skriva ut dem på skärm eller printer genom att först ta bort eller hellre ersätta med REM-sats rad 10 samt rad 11 i TVMAIN3.

Ta hänsyn till buffertstorleken

TV-editorn är så konstruerad att allt tillgängligt minnesutrymme som inte behövs för programmet kommer att utnyttjas som textbuffert. Det är naturligtvis önskvärt att ha en så stor buffert som möjligt. Buffertstorleken bestämmer hur stor text som kan behandlas. Vill man skriva, läsa eller bearbeta en text som innehåller fler tecken än buffertstorleken måste man dela upp texten i två eller flera delar och lagra varje del som en separat fil.

I TV-editorn är variabeln M$ textbuffer, denna variabel har dimensionerats till det värde som variabeln M9% har. Det är lätt att ta reda på hur stor buffert man med olika versioner av TVMAIN får tillgång till med den utrustning som användes genom att stoppa programmet (med CTRL-C eller kommandot ;H) och sedan skriva ut värdet genom att skriva PRINT M9% <RETURN>. Vill man se hur mycket text som finns i bufferten gör man sedan återinträde med LIST (se nedan) och ger kommandot Z=. Skillnaden mellan M9% och detta värde är det utrymme man har kvar för ytterliggare text.

Beroende på den textbearbetning man vill utföra och den utrustning man har kan man behöva kompromissa mellan buffertstorlek och tillgång till nyttiga kommandon. Har du 16 K extraminne så kan du använda TVMAIN3 i de fall texten inte överstiger c:a 70 sektorer. Varje radslut <CRLF> räknas som två tecken i bufferten. Av dessa lagras enbart <CR> vid skrivning på fil.

Buffertstorleken påverkas av Q-registrets maximala kapacitet. I Q-registeret kan lagras in max 2000 tecken. Detta värde kan ändras genom att dimensionera om strängvariabeln Q$ i rad 1010 till annat värde. Gör även motsvarande ändring på rad 2696 i TVMAIN3. Observera att när du gjort någon ändring i TVMAIN så går det inte att återinträda med list. Glöm inte att göra SAVE om du vill behålla ändringen.

Laddning av TV-editorn

Det finns de som misslyckats med att ladda TV-editorn. På ABC-kassett nr 2 låg ursprungsversionen av TV-editorn, filerna TV.BAC, TVSUBR.ABS och TVMAIN.BAC. De kunde bara användas om man hade 5 tums flexskiveutrustning. Laddningen startas med RUN TV. Programmet TV består av raderna 1000 till 1060, resten är enbart REM-satser.

```
1000 POKE 65408%,33%,146%,255%,205%,
     94%,109%,17%,240%,255%,205%,27%,
     6%,56%,4%,33%,255%,255%,201%,
     33%,0%,0%,201%
1010 POKE 65520%,84%,86%,83%,85%,66%,
     82%,32%,32%,65%,66%,83%
1020 IF CALL(65408%) THEN 1040
1030 ; "Can't find TVSUBR" : STOP
1040 ; CHR$(12%);"-wait-"
1050 ; CUR(10%,0%);"Reenter with LIST"
1060 CHAIN "tvmain"
```

Tittar man närmare på de siffervärden som pokas i raden 1010 så ser man att de motsvarar namnet på filen, "TVSUBR ABS". Rad 1000 pokar in en liten rutin i maskinkod som sedan anropas i rad 1020. Denna rutin letar efter filen TVSUBR.ABS och läser från filen de maskinkodsrutiner som sedan utnyttjas av TV-editorn. Rutinerna lagras i DOS-buffert 3-5, dvs fr o m adress 63488.

Christer Johnson <681> har påpekat att denna inlagring inte fungerade på hans DataDisc 82 som hade ett DOS märkt "6-81". När han ändrade 5:te byten i rad 1000 (rad 300 i TV80) från värdet 94% till 89% fick han laddningen att fungera.

Om laddningen inte fungerar skrivs texten "Can't find TVSUBR" ut på skärmen och programmet stoppar. Filen TVSUBR.ABS måste finnas tillgänglig i någon av drivarna. Därefter sker i rad 1060 chain till TVMAIN som också måste finnas tillgänglig. Lämpligen döper du den version av TVMAIN som du vill använda till TVMAIN.BAC och lägger den på den programskiva du använder för TV-editorn.

Använder du enbart kassettutrustning så skall programmet TVCAS.BAC som låg på ABC-kassett nr 3 användas i stället för TV.BAC. I kassettversionen läses TV-editorns maskinkodsrutiner från filen TVSUB.BAC som alltså ersätter TVSUBR.ABS. Lämpligen gör du i ordning en systemkasset för TV-editorn där du i tur och ordning lägger filerna TVCAS.BAC, TVSUB.BAC samt TVMAIN.BAC. Det sista programmet skall vara den version av TVMAIN du normalt vill använda, i förekommande fall omdöpt.

TVCAS och TVSUB kan som nämndes i föregående artikel användas även för flexskive utrustning. Har du 80 teckens bildskärm (MyAB:s eller GeJo:s) skall du använda TV80.BAC, på ABC-kassett nr 3, i stället för TV.BAC. Även TVCAS fungerar för 80 teckens bildskärm.

Om du vill ladda in TV-editorn igen efter att ha kört ett annat program, räcker det normalt att enbart ladda in TVMAIN på nytt. Maskinkodrutinerna ligger oftast kvar oförändrade. Samma gäller om du vill övergå till en annan version av TVMAIN.

Negativa argument

I TVMAIN3 kan man ha negativa argument till kommandona C, D, L, K och Q. Exempel:

-15C flyttar markören 15 steg bakåt i texten.
-d tar bort det tecken som föregår markören,
-2d tar bort bort två tecken, användes lämpligen när man markören står på början av en rad och man vill ta bort radslutet <CRLF> på föregående rad. Sekvensen -2DI<mellanslag> lägger även till ett space efter sista ordet på föregående rad. -20000D tar snabbt och effektivt bort allt från textens början fram till markören.
-3L flyttar markören till början av rad 3 rader högre. Observera att om markören ej står på början av rad så inräknas även den rad som markören står på.
-3K tar bort motsvarande rader/radbitar och **-3Q** lägger in dem i Q-registret. Kommandosekvensen -KK tar bort två rader om markören står på början av rad men bara en rad om markören befinner sig in på raden.

Du rekomenderas att innan du uppnått vana inte använda -3K eller -3Q förrän du först sett effekten av -3L. Glöm inte att flytta markören tillbaka igen efter testet, -3K tar bort 3 rader före det läge markören har när kommandot utföres.

Makrokommando

I TVMAIN3 har man tillgång till makrokommando. Som nämndes i artikeln i ABC-bladet 1982:1 kan man foga flera kommandon efter varandra som utföres först när man avslutar sekvensen med två högerpilar i följd. När man behöver samma sekvens flera gånger kan det vara praktiskt att först definiera sekvensen som ett makrokommando som sedan utföres varje gång man ger ett särkilt exekveringskommando. Kommandot M följt av den aktuella sekvensen av kommandon och avslutat med två högerpilar definierar sekvensen som ett makrokommando. Därvid inlagras sekvensen i det s k M-registret, variabeln M4$, max 60 tecken. Där ligger det kvar tills man definierar en ny makrosekvens med nytt kommando M. Makrokommandot exekveras genom kommandot ;M (semikolon M).

En mycket kraftfull (och potentiellt farlig) finess är att makrokommandot kan göras rekursivt genom att den kommandosekvens man definierar som makro får avslutas med ;M. När vid exekveringen av makrokommandot ;M påträffas så ges nytt exekveringskommando och utförandet av kommandosträngen i makrokommandot upprepas ända tills något stoppvillkor blir uppfyllt. Stoppvillkor kan vara att någon sökning misslyckas eller markören vill flyttas utanför textens gränser.

Q-registret spiller över

När man i TVMAIN2 försöker lägga in för mycket text i Q-registret (mer än 2000 tecken) så spiller texten över i bufferten med följd att delar av den tidigare texten där blir ersatt med ny text som spillts över från Q-registret.

I TVMAIN3 har lagts in en kontroll som ger felutskrift när man försöker lägga in mer än 2000 tecken i Q-registret. Kontrollen görs i IF-satsen i andra delen av rad 2696. För att eliminera felet i TVMAIN2 kan man lägga in samma rad där.

Återinträde med LIST

Som förutskickades i förra artikeln har i TVMAIN3 lagts in en rad 11 som vid checksummor 10042 och 9913 ger samma funktion som rad 10 vid checksumma 11273. Du kan ta bort den rad du inte behöver men om du växlar mellan tangentbord med olika checksummor kan det vara bra att ha den kvar. Har du kvar rad 10 och har checksumma 10042 eller 9913 så kan du göra återinträdet med LIST 11 (eller ED 11). Prova! Många har undrat hur rad 10 egentligen ser ut, den går ju inte att lista. Svaret är att det finns ingen rad skriven i BASIC som kompilerar till den kompilerade formen som återgavs i artikeln i ABC-bladet 1982:1. Därför kan programmet inte lagras i BAS-format.

Programmakarna har utnyttjat en "feature" i BASIC-tolken som gör att återlistningsrutinerna i BASIC-en initierar exekvering av del av raden tolkad som maskinkod. Den i detta sammanhang väsentliga funktionen är att hopp sker till adress 3425 (13:97) resp 3423 (13:95) vid checksumma 11273 resp de båda andra. (Adress 3423 motsvarar 13:95. Ett litet tryckfel har här insmugit sig föregående artikel). Inhopp på adressen i fråga ger RUN-mode utan föregående CLEAR (nollställning av alla variabler). Rad 1000 i TVMAIN är också väsentlig i detta sammanhang, raden medför att raderna 1010- 1190 bara genomlöpes första gången vilket är väsentligt.

Rad 10 resp 11 kan extraheras, numreras om och sparas separat och genom MERGE fogas in i andra program som då även bör ha en rad motsvarande rad 1000 för att man skall få funktionen återinträde utan nollställning av variabler.

Finns det plats på disken?

När du arbetar med flexskiveutrustning med en text som du vill skriva på diskett kanske du ställs inför frågan om det du skrivit ryms på disketten. Det kan vara bra att veta hur många sektorer som finns lediga innan man skriver filen. Här kan programsegment TVLIB.MRG som ligger på ABC-kassett nr 4 vara till hjälp. Detta segment är avsett att genom MERGE komplettera TVMAIN3. Faciliteten kostar dig 688 tecken i minskad bufferstorlek.

Vill du ha denna möjlighet förfar du så här: Gör LOAD TVMAIN3, sedan MERGE TVLIB.MRG och därefter SAVE TVMAIN3B. TVMAIN3B är en version som ger dig till gång till två ytterliggare kommandon:

;0 skriver ut hur många sektorer som är lediga på drive 0,
;1 ger dig motsvarande för drive 1.

Kommandotabell

På mångas begäran publiceras en kommandotabell som sammanfattar alla kommandon som (hittills) finns i TV-editorn. Tabellen är ej avsedd att vara en fullständig beskrivning av tillgängliga kommandon utan mera som en minneslista eller lathund att ha lättillgänglig när man arbetar med TV-editorn.

Kommandon i TV-editorn

Tecknet * användes för att i kommandona nedan beteckna <högerpil>. Kommandona skrivs här med stor bokstav men kan lika gärna ges med liten bokstav.

n betecknar talvärde som användes som argument vid vissa kommandon.
z betecknar antalet tecken i bufferten (användes som argument)
. anger markörens position räknat från textens början.

Direkta kommandon som utföres omedelbart för att flytta markören och visa olika delar av texten i bufferten:

<RETURN>	flyttar markören till början av nästa rad
<tyskt Y>	flyttar markören till början av raden resp till början av föregående rad
<högerpil>	flyttar markören ett steg åt höger
<vänsterpil>	flyttar markören ett steg åt vänster
<mellanslag>	visar den del av texten som står efter --more--
CTRL-B	flyttar markören till mitten på den sida som visas
CTRL-L	centrerar texten på skärmen symmetriskt kring markören

Övriga kommandon som utföres först när man tryckt tangenten <högerpil> två gånger i följd:

I<text>*	insättning av <text> på markörens plats
J	flyttar markören till textens början
nC	flyttar markören n steg åt höger (åt vänster om n är negativt)
zC	flyttar markören till textens slut
nL	flyttar markören till början av raden n rader längre ned (längre upp om n är negativt)
nS<text>*	söker efter n:te förekomsten av <text>
.=	skriver ut markörens position
z=	skriver ut antalet tecken i bufferten
nD	tar bort (delete) n tecken efter markören (före om n är negativt)
zD	tar bort allt efter markören
JZD	tömmer hela bufferten
nK	tar bort (kill) n rader efter markören (före om n är negativt)
nR<gammaltext>*<nytext>*	byter ut (replace) första n påträffade <gammaltext> mot <nytext>
nI*	lägger in tecken med ASCII-värdet n på markörens plats
nQ	lägger in i Q-register n rader efter markören (före om n är negativt)
G	sätter in texten i Q-registret på markörens plats
;Y	läs in text från fil och lägg den efter befintlig text i bufferten
;S	skriv innehållet i bufferten på fil
;X	skriv innehållet i bufferten på fil och exit från TV-editorn
;H	stopp
M<kommandosträng>**	definiera <kommandosträng> som makrokommando
;M	utför makrokommandot
;0	visa antalet lediga sektorer på drive 0
;1	visa antalet lediga sektorer på drive 1

Om man försöker utföra ett kommando som inte finns implementerat i aktuell version av TVMAIN kommer texten Illegal com: <kommando>

Kontrolltecken i texter

S k kontrolltecken, dvs tecken med ASCII-värde mindre än 32, ibland kallade "icke skrivbara tecken", användes för att styra olika printerfunktioner. De vanligaste är ASCII 13 <CR> carridge return eller vagnretur, ASCII 10 <LF> linefeed eller ny rad, ASCII 12 <FF> formfeed eller frammatning till ny sida, ASCII 9 <HT> horisontal tabulator, ASCII 8 <BS> backspace eller bakåtstegning. Det finns printertyper som använder även andra kontrolltecken än dessa för att ställa in printern eller styra speciella funktioner vid utskrift. Någon enhetlig standard tycks inte finnas.

Det finns kontrolltecken som ABC-80:s filhanteringssystem inte klarar av att lagra textfil. Behöver man sådana kontrolltecken måste man lägga in dessa medan texten finns i bufferten och därifrån skicka utskriften direkt till printern. Det går att lägga in alla kontrolltecken i text som finns i bufferten. Ett kontrolltecken t ex <HT> som inte kan läggas in med det vanliga insättningskommandot I<text><högerpil> kan enligt vad Lars Frej <2260> påpekat läggas in med kommandot 9I<högerpil>.

Gunnar Tidner <1306>

FORTH för ABC80

av Torbjörn Alm (medlem 116)

Jag har under 2 veckor haft möjlighet att prova den fig-FORTH för ABC-80, som framtagits av Robert Johnsen, Uppsala Universitet, och som ABC-klubben förvärvat rättigheterna till. Programvaran kommer på nästa ABC-kassett och ställs till medlemmarnas fria disposition.

Den har överträffat mina förväntningar, och den är en mycket fin produkt. Den är enkel att använda, och innehåller en del extra finesser, som nyttjar ABC-80:s speciella möjligheter. ABC-80 FORTH baserar sig på fig-FORTH, som har tagits fram av orth Interest Group, San Carlos för fri spridning. Att lägga in den i en viss dator är givetvis förbundet med en hel del arbete. Programmet är c:a 7 k Byte långt. Antal labels är precis på gränsen till vad avancerad assembler för ABC-80 tål (>500).

ORTH är ett mycket eget språk, som uppfanns av Charles Moore 1973, när han skulle styra ett radio-teleskop med hjälp av en PDP-11, utrustad med teletype och magnetband, en föga avundsvärd uppgift. Man arbetade samtidigt med en annan dator, och enbart med assembler. Genom att skapa FORTH, som kan betecknas som ett maskinoberoende (i stor utsträckning) lågnivåspråk, kunde man lösa sina problem. Sedan dess har språket spritt sig, inte minst tack vare tillkomsten av fig-FORTH.

Utöver fig-FORTH eller FORTH-79, finns det POLYFORTH, som säljs av FORTH Inc., som drivs av just Charles Moore. Det finns vissa olikheter mellan dessa dialekter, men i stort sett kan man flytta FORTH-program mellan olika maskiner. Det tycks finnas en tredje variant, MMSFORTH för TRS-80, men jag vet inte vad den liknar.

Vad är nu FORTH? Språket är ett "threaded nguage". Koden består av en följd av adresser till rutiner, som i sin tur kan vara i masinkod, s.k. primitiver eller i FORTH-kod, sekundära funktioner. En av ideerna med fig-FORTH är att 90% av koden är sekundära funktioner, och de kan direkt flyttas mellan olika processorer, nder det att c:a 10% är processorunik assemblerkod.

Ett FORTH-system innehåller bl.a. följande delar: ett user interface med rutiner för inmatning och utmatning till skärm och till skrivare, en interpreter, som exekverar FORTH-koden, en kompilater, som översätter FORTH källkod till internkod samt ett system för virtuellt yttre minne. Dessutom finns det en editor för att editera FORTH-program samt en assembler för att kunna skriva primitiva funktioner.

FORTH förutsätter enbart att det finns något yttre medium på vilket man kan läsa och skriva block. Dessa block slås sedan ihop till sk. screens, vilka brukar vara 1 k byte, men i ABC-80 har valts 768 bytes (3 block), då det är enklare att hantera på skärmen.

För att editera program har man i ABC-80 FORTH inte mindre än 3 olika editorer till sitt förfogande:

fig-FORTH editor, som ingår i fig-FORTH installationsmanual, och finns beskriven i denna.

POLYED, som är ursprungligen definierats för POLYFORTH, och som finns beskriven i Leo Brodies bok: Starting Forth.

VEDIT, som är en enkel och lätthanterlig bildskärmseditor.

Varför är FORTH så populärt?

Språket är mycket maskinnära utan att vara masinkod, och programmen blir mycket effektiva. Det andra är att det är ett rent strukturerat språk. Någon GOTO finns inte, utan man arbetar med funktioner, sk. FORTH-ord, vilka i sin tur byggs upp av egna ord eller ord i systemet. Det finns LOOP-funktioner, IF-THEN-ELSE m.m. och det är enkelt att skapa en CASE-sats, vilket jag visar senare. En mycket trevlig egenskap är att man kan skriva in en funktion på skärmen, ladda funktionen i kompilatorn och sedan omedelbart testa den, samt om dem gick fel, editera och testa igen. Det gäller även FORTH-funktioner i assembler. Man kan jämföra med arbetet med program i assembler!

Efter hand får man ett bibliotek med screens med fungerande rutiner, som sedan kan kombineras ihop. Alla funktioner anropas med sitt namn, och enda kravet vid kompilering av en funktion är, att de refererade funktionerna är definierade.

Man kan definiera konstanter och variabler med namn, och man kan ha konstanter direkt i koden. Det går även att beräkna adresser till funktioner och lägga dem på stacken, för att sedan exekvera dem.

Den assembler, som finns till FORTH ABC-80 kan assemblera 8080-kod. För den som vill ha en fullständig Z80-assembler, rekommenderas Loeligers bok: Threaded Interpretative Languages.

Start av FORTH:

FORTH ABC-80 är ett .ABS-program, och startas över CMDINT.

1. Sätt FORTH system-skiva med Forth-scr.een och FORTH.ABS i drive 0.
2. Gör BYE. När CMDINT laddats, skriv FORTH.
3. När FORTH laddats, svarar den med att överst på skärmen skriva Z80 FORTH 1.1
4. Skriv 6 LOAD 10 LOAD RETURN . Nu laddas en del användbara funktioner. Bland annat laddas funktionen WHERE, som kan användas för att söka fel vid inläsning av program samt laddrutiner för fig-FORTH editor, POLYED samt assembler. Screen 10 innehåller programmet PRINT-BLOCKS, med vars hjälp screens kan skrivas ut på skrivare, 2 per sida.
5. Om man har diskar med enkel densitet, skriver man 0 DENSITY ! RETURN.
6. Nu är det klart att köra.

Skärmeditor VEDIT.

För att använda VEDIT och editera screen 70, skriver man 70 VEDIT. Screenen listas på skärmen med radnummer i vänsterkant och rutor i högerkant. Om man skriver på tangentbordet, läggs tecknen in där markören befinner sig. Om man vill förflytta markören använder man pilarna för förflyttning åt vänster eller höger, RETURN för att gå nedåt och ü för att gå uppåt, på samma sätt som i TV-editorn. Om man gör CTRL/E´ lämnar man VEDIT, och för att spara screenen på disken gör man FLUSH RETURN.

VEDIT kan inte användas för att spränga in text i eller mellan rader. För detta behöver man tex. POLYED. För att ladda POLYED skriver man POLYED RETURN efter det att man gjort 6 LOAD. För att sedan aktivera editorn, skriver man EDITOR RETURN. Då kommer de speciella editeringskommandona att aktiveras. Därefter skriver man 70 LIST RETURN. Som synes skall varje kommando eller följd av kommandon avslutas med att man gör RETURN för att de skall exekveras. När alla kommandon är klara, skriver FORTH ut ok som kvittens och är klar för nästa kommando. Det gäller även editorerna.

POLYED-kommandon:

WIPE blankar aktuell screen.

L listar aktuell screen.

5 T skriver ut rad 5 och pekar ut 5 som aktuell rad.

U TEXT lägger in TEXT under aktuell rad. Underliggande rader flyttas ned. Rad 20 försvinner.

P TEXT ersätter aktuell rad med TEXT.

F XXX söker från aktuell rad efter XXX. Vid träff skrivs raden med XXX ut, och markören står efter XXX.

R YYY ersätter det som stod i söksträngen. (XXX) med YYY.

E tar bort XXX.

FLUSH återskriver screenen till disken.

Det finns ytterligare kommandon, som jag ännu inte hunnit stifta bekantskap med. POLYED finns beskriven i Brodies bok Starting Forth.

Hantering av printer och V24-utgång.

ABC-80 FORTH innehåller dels en V24-handler, som i princip är standard V24-handler, och som kan mata ut vid valfri baudrate, dels länkning till parallell-printerkoden, som skall finnas i 7800H, dvs Centronics snitt eller P40. Det finns en intern variabel, PR-TYPE, som normalt är =1, dvs Centronics, men om man skriver 2 PR-TYPE !, avses V24. Baudrate anges med kommandot 2400 BAUD. Om man sedan sätter printerflaggan genom att skriva PR-ON, kommer att utmatning till skärmen även att gå till printer. Dessutom finns det 2 funktioner, ASCII-IN och ASCII-OUT, vilka kommunicerar via V24-utgång, oberoende av printer-flaggor.

Exempel på FORTH-verb.

FORTH ABC-80 innehåller över 200 kommandon, och skulle jag beskriva dem alla, blev det en hel bok. Eftersom det finns flera bra sådana, nöjer jag mig några få:

+ Addera de två översta talen på stacken, och lägg tillbaka resultatet. Likheten med en HP-kalkylator är slående. På samma sätt fungerar - * och / .

. trycker talet på toppen av stacken.

/MOD utför en heltalsdivision och lägger tillbaka såväl kvot som rest.

/* gör först en multiplikation, sedan omedelbart en division. Vitsen ligger i att mellanresultatet hålls i dubbel precision.

É Hämtar ett 16-bits ord, vars adress finns på stacken.

! lagrar ett 16-bitars ord på stacken enligt en adress på stacken.

: NAMN <instruktioner > ;
definierar funktionen NAMN och lägger in den i ordlistan.

' (kallad tick) används för att lagra adressen till en funktion.

' ABC CFA
lägger adressen till ABC på stacken. Om man sedan gör EXECUTE, exekveras ABC. Här finns en skillnad mot POLYFORTH, som inte framgår av Brodies bok.

Användbara hjälpfunktioner i ABC-80 FORTH:

VLIST listar namnet på alla funktioner som hittills definierats. Om du gjort PR-ON så erhålls lista även på printer.

3 30 INDEX
listar första raden på skärmarna 3-30. En god regel är att första raden (rad 0) på varje screen skall innehålla en kommentartext om screenen. Kommentar inleds med (och avslutas med).

3 30 PRINT-BLOCKS
(finns på screen 10) listar screen 3-30 på printer, 2 screens per sida.

CTRL/< RETURN
dumpar skärmen på printer. Denna finns på screen 8 och laddas av 6.

Trace av FORTH-kod.

Denna funktion finns inte med på kasetten, men har visat sig mycket bra. Om man lägger upp den på en screen och läser in den före de rutiner man vill spåra, erhåller man en utskrift med mamn och toppen av stacken för varje rutin, om variablen TFLAG sätts =1.

```
0 ( TRACE                                )
1 FORTH DEFINITIONS
2 0 VARIABLE TFLAG
3 :(TRACE) ( IF TFLAG IS ZERO,THEN
4            NO TRACE OCCURS      )
5 TFLAG E'E IF CR R 2-NFA DUP ID.
6 CE'E 31 AND 32 SWAP - SPACES
7 -2 4 DO SPE'E I+E' 8 .R-2 +LOOP
8 THEN ;
9 : :  ( REDEFINE : )
10 ?EXEC !CSP CURRENT E' CONTEXT!
11  CREATE ' (TRACE) CFA DUP E'E
12  HERE 2 - ! , RB ;
13 ;S
```

Funktionen visar vad som kan göras med FORTH.

Ett annat exempel är definitionen av en CASE-sats.

```
( CASE                                   )
FORTH DEFINITIONS DECIMAL
: CASE:
<BUILDS SMUDGE RB
DOES> SWAP 2 * + E'E EXECUTE
;
;S


( EXAMPLE )
: 0PET ." APA" ;
: 1PET ." BABIAN" ;
: 2PET ." ANNAT" ;
CASE: DJUR 0PET 1PET 2PET ;
0 DJUR ( SKRIV APA )
1 DJUR ( SKRIV BABIAN )
2 DJUR ( SKRIV ANNAT )
```

CASE: är exempel på en högnivå-definition, som är en avancerad feature hos FORTH. Det är ingen tvekan om att FORTH är ett språk, som sätter fantasin i rörelse hos alla hackers. Det finns nästan outtömliga möjligheter att skriva kompakt kod. Tyvärr tenderar den lätt att bli write-only, dvs. ingen, ej ens författaren kan läsa den.

Här är Fig-FORTH!

Fig-FORTH som presenterades i ABC-bladet 1.5 (sommarutgåvan) har utvecklats av docent Robert Johnsen vid Fysikalisk-kemiska Institutionen, Uppsala universitet. Den kan köras på en standard ABC-80 utrustad med 5-tums flexskiveutrustning. Programmet är skrivet i assembler och köres under CMDINT.SYS.

Torbjörn Alm har provat programvaran och skrev en recension för ABC-bladet 1.5 som p g a ombrytningsfel återges på nytt.

Programvaran distribueras på **kassett till ABC-klubbens medlemmar.** På ABC-kassett nr 4 ligger FORTH16.ABS för ABC-80 utan extra minne. Den kan köras även på ABC-80 med 32 K RAM men därvid utnyttjas inte hela RAM-minnet effektivt. För den utbyggda versionen finns FORT32.ABS som läggs på ABC-kassett nr 5. FORTH utnyttjar inte det vanliga filhanteringssystemet på ABC-80 utan lagrar program och data på s k screens. Varje screen upptar 3 sektorer på disketten och lagras fr o m spår 3 sektor 1 och uppåt. Man bestämmer själv hur stort utrymme man vill reservera för screens. Om man reserverar plats för 75 screens så ryms hela programbibloteket, tiotalet egna screens samt FORTH.ABS och CMDINT.SYS på en skiva med enkel densitet. Programbibloteket till FORTH distribueras i form av textfilerna SCREEN.TXT och SCREEN2.TXT. SCREEN.TXT innehåller screens nr 3 till 36. SCREEN2.TXT som läggs på ABC-kassett nr 5 innehåller screens nr 34 till 69 samt en fullständigare version av screen nr 7. Screens nr 34 till 36 i SCREEN.TXT kommer att skrivas över av andra screens från SCREEN2.TXT. Det kan vara skäl att spara screen 36 genom att flytta den till annan plats med nummer högre än 69.

Fortsättning på sidan 23

FORTH-Litteratur:

1. **Starting FORTH,** Leo Brodie, Forth Inc. Prentice-Hall.
Detta anses allmänt, och det med rätta, vara den absolut bästa introduktionen till FORTH. Boken är lättläst och har humor. Den är rikt illustrerad av författaren. Här finns POLYED beskriven.

2. **fig-FORTH Installation Manual.** Forth Interest Group, San Carlos.
Här finns dels stora delar av interpreten i FORTH-kod, så att man sätta sig in i vad den har för sig. Här finns även beskrivningen av fig-FORTH editorn.

3. **ABC-klubben RAPPORT 2, Fig-FORTH on ABC-80** implemented by Robert Johnsen. Juni 1982.
En rapport 46 sidor skriven på engelska och innehållande bl a installationsmanual för ABC-80 med köranvisningar och screenbeskrivningar samt utförlig referenslista. Dessutom innehåller rapporten utdrag ur ref 2 i form av Glossary (verb-beskrivningar) och beskrivning av Fig-FORTH editorn.

4. **Threaded Interpretative Languages,** R. G. Loeliger. Byte Books (McGraw-Hill). Denna bok beskriver på maskinnära nivå hur en FORTH-interpreter byggs upp av ett stort antal små rutiner och hur de samverkar. Här finns också en Z80-assembler i FORTH, som bör gå att göra om till ABC-80 FORTH. Den innehåller mycket få fel. Jag har själv implementerat denna FORTH, och fann 2 fel! Den kan dock inte nyttjas till produktion, eftersom den saknar väsentliga delar. För den som vill veta mer om hur FORTH arbetar internt, rekommenderas denna bok varmt.

5. **FORTH DIMENSIONS** är en tidskrift, som ges ut av Forth Interest Group.
Den innehåller artiklar och FORTH-program. Jag vet inget om denna själv, men Robert Johnsen, som gjort ABC-80 FORTH, har hämtat en hel del godbitar ur denna, varigenom ABC-80 FORTH är bättre än standard fig-FORTH på flera punkter.

ABC80 blir RAM-maskin

Jag har låtit bygga om min ABC80 till RAM-maskin. Jag fick hjälp av en kompis som är snitsare. Den ombyggnadssats jag valde var Ge-Jo's. Ge-Jo står för Göran Johansson i Ljung, en kille som gjort en hel del intressanta saker för ABC80. Nåväl nu gällde det att montera in en RAM-tillsats, vilket visade sig betydligt svårare än att stoppa in 16K expansionsminne. Det är stor skillnad mellan bygganvisnigarna till RAM-tillsats och expansionsminne. Byggbeskrivningen för expansionsminnet är klart bättre. En del punkter i beskrivningen skulle nog delas upp i flera. En ytterligare komplikation är att beskrivningen inte stämmer för alla tangentbord. Man letar alltså förgäves efter en viss kondensator om man har ett äldre tangentbord, men i sanningens namn ska sägas att Göran Johansson är en hygglig kille som gärna svarar på frågor. Varför ska man nu bygga om sin ABC80 till RAM-maskin? Jag kan se flera skäl som kan delas upp i två huvudgrupper. Ett skäl kan vara att man vill ha möjlighet att utnyttja något annat operativsystem exempelvis CP/M. Ett annat skäl kan vara att göra förändringar i Basic-tolken. För de som undrar hur en ABC80 fungerar efter en ombyggnad ska jag försöka förklara.

Det är möjligt att köra sin ABC80 ungefär som vanligt efter en ombyggnad. En ombyggd ABC80 fungerar dock annorlunda eftersom RAM-minnet är tomt när strömmen slås på måste någonting hända, som kopierar Basic-tolken från PROM-kapslarna till RAM-et. Detta sker också automatiskt, varefter man kan köra som vanligt. Det är inte bara Basic som kopieras utan även printer-prom och den del av Basic som ligger på controler-kortet i floppyn.

Mer minne

Ombyggnaden leder till att man får 32K RAM-minne för program och data ovanför bildminnet. Under bildminnet får man nu RAM-minne inom ett adressområde som tidigare varit tomt. Om man inte har floppy och 40 kol skärm så får man alltså 32 K minne ovanför bildminnet och 15 K mellan Basic-tolken och bildminnet. Har man floppy så får man 8 K minne mellan diskfilhanteringsdelen av Basic-tolken, dessutom får man 3 K direkt under bildminnet. Vad ska man nu använda det nya minnet till? Ja när det gäller de 32 k som ligger över bildminnet så används dessa som vanligt. Man kan ha större program, deklarera stora matriser osv. Det är alltså ingen skillnad mellan dessa 32 K och ett minne som expanderats till 32 K utan att maskinen ändrats för övrigt. Det som är spännande är alltså att utnyttja minnet under bildminnet och att utnyttja möjligheten att ändra i Basic-tolken. Det är då fråga om att utnyttja helt nya möjligheter som bara ges då man bygger om till RAM-maskin. Det nya minnet kan exempelvis användas för lagring av maskinspråk, men det är faktiskt möjligt att ladda in Basic-program under bildminnet. För att kunna göra detta måste man ändra BOFA till 16384. Den ändringen sker med kommandot 'POKE 65053,64'. Jag har exempelvis laddat in Ge-Jo's program Subrutin, varefter jag ändrat BOFA med 'POKE 65053,128'. Och det fungerar! Nästa steg är att lägga Lathund och ABCV24 mellan 16 K och 24 K.

Jag har ändrat promptern till K-Å J och i stället för 'ERR 21' får jag 'FEL 21'. Dessa förändringar åstadkommer jag genom att poka in resp text i stället för befintlig. Principen är densamma vid alla förändringar av de typer som nämts ovan. Problemet består då endast i att hitta resp textsträngar i BASIC-tolken, vilket kan vara nog så svårt. Lämpliga hjälpmedel förutom tålamod är Disassemblerlistan och något program som letar efter textsträngar i BASIC-tolken. Detta program ska skriva ut första adress i resp sträng. När man skaffat sig en förteckning över var i tolken som RUN, GOTO, LIST osv ligger kan man börja ändra.

Att ändra sin BASIC

När Basicen ligger i RAM-minnet så kan man ändra i den på ett ganska enkelt sätt. Man använder sig antingen av POKE-kommandon (när det är små ändringar) eller program bestående av POKE-satser. Exempel på ändringar som man kan göra är: förändrad promter, ändrade namn på kommandon och nya namn på Basic-satser.

BASIC på Svenska

En möjlighet som öppnar sig är att göra en BASIC på Svenska. Det gäller att översätta kommandon och satser till Svenska motsvarigheter. En komplikation är att de Svenska översättningarna måste bestå av lika många tecken som de Engelska motsvarigheterna. Man kan tänka sig följande exempel på översättningar:
LET=GÖR, GOTO=HOPP, POKE=FYLL, PEEK=KIKA etc.

Vad kan man ha för nytta av Svensk BASIC?

Nyttan är förmodligen inte så stor, men kanske en Svensk BASIC kan ha ett visst värde i undervisning. Framför allt bör det bli lättare att använda datorn på mellanstadiet i grundskolan. En ABC80 med endast RAM-minne som fått sin BASIC ändrad till Svenska, kan köra program lagrade i BAC-format som vanligt. Det förhåller sig annorlunda med BASIC-program i BAS-format. Ett BAS-program kan bara köras om det innehåller samma satser som BASIC-tolken. En Svensk BASIC kan alltså användas för kryptering. Gör så här, läs in det BAC-program som ska krypteras och spara det sedan med LIST. Nu innehåller den sparade filen dina översatta BASIC-satser.

En mera nyttig användning av en ändrad BASIC skulle kunna vara att göra program skrivna i andra BASIC-dialekter körbara på ABC80. Det är naturligtvis inte möjligt att ändra funktionen av en sats genom att byta stavning av den, men när två BASIC-dialekter skiljer sig åt genom att använda olika satser för samma funktion , kan det räcka med att ändra på det sätt som vi beskrivit.

Att ändra funktioner i BASIC

Jag har ännu inte kommit så långt att jag börjat ändra BASIC-tolkens funktioner och jag har inte heller försökt lägga till nya. Jag har däremot haft svårt att avhålla mig från att fundera över vad som skulle vara möjligt. En intressant sak skulle vara att bygga ut BASIC i utrymmet över 16 K. Det man måste göra är att beskriva de nya satsernas funktion, ge dem namn och operationskoder. Med operationskod avser jag den interna halvkompilerade kod, som ABC80 arbetar med internt.

DP/M och andra operativsystem

På denna front finns för närvarande inte så mycket att säga eftersom Göran Johansson inte är färdig med CP/M för sina ombyggda ABC80. Han räknar att bli färdig till hösten. För min del är det dock inte någon brådska - jag ska lära mig FORTH!

Kjell-Åke Johansson

Kassett ett

CASTODSK.BAS
Detta program kopierar en fil från kassett till floppy. Det kan alltså endast köras av den som har floppy. Vid inladdning från kassett i system utan floppy, eller om floppyn ej är del av systemet, får man felmeddelandet ERR 8 LINE 380. Detta felmedelande beror på att instruktionen på rad 380 ej kan tolkas av den Basic som finns i tangentbordet. Vid instruktionen NAME sker ett hopp till den utbyggnad av Basic som finns på controler-kortet i floppyn. Felmeddelandet ovan är alltså i sin ordning när systemet saknar floppy.

Vid körning av programmet bör man vara uppmärksam på att det kan förstöra filer på floppyn ifall dessa råkar ha samma namn som filer på kassetten, som ska kopieras. Mot detta skyddar man sig genom att alltid kopiera till tom floppy.

Tänk på att CASTODSK endast kopierar text-filer. Det innebär att program som sparats med SAVE ej kan kopieras med programmet. Det kan vara bättre att använda CASDISK som numera ligger som första fil på ABC-kassetterna.

MAXEN.BAS
Happening, rolig och enkel roar barn. Körs på grundutrustning liksom alla program tom OHMDATA.

SLALOM.BAS
Det gäller att ta sig fram mellan portarna. Du kan tävla med dig själv eller med dina vänner. Det fungerar bäst om tangentbordet är låst i versal-läge och röd lampa lyser.

REAKTION.BAS
Testa din reaktionsförmåga, du kan tävla med dina vänner.

GUBBEN.BAS
Gissa ord. Missar du blir du hängd.

KRIG.BAS
Krigsspel

MÖNSTER.BAS
Ritar (vackra?) mönster på skärmen. Bra xempel på vad grafiken kan - och inte kan!

LUFFAR.BAS
Luffarschackprogram

VIRRE.BAS
Underhållningsprogram av orginellt slag. Virre är en man som rör sig på olika sätt. Virre kan hoppa tala och mycket annat som du kan styra. Virre är de små barnens förtjusning, men har många vuxna anhängare.

MYSKO.BAS
Är mycket mysko - grafik och ljud och överaskning! Elaka tungor förmäler att programmet innehåller en bug!

KULGRAF.BAS
Grafikprogram som ritar ett regelbundet mönster på skärmen. Mönstret tätnar tills skärmen är helt vit varvid mönstret börjar upplösas och försvinna osv i all oändlighet.

SLOOP.BAS
Musik ur ABC80 är det verkligen möjligt? Kan man verkligen få ut något annat än pang! choff! oioioi! ur ABC80? Du får avgöra den saken själv genom att lyssna på Sloop.

EURFRÅG.BAS
Frågesport för familjen. Vem vet mest om Europa?

MATTEN.BAS
Program för att öva aritmetikfördigheter. Programmet är på mellanstadienivå och kan användas i skolan.

ROTEN.BAS
Kan köras på grundutrustning. Programmet beräknar kvadratroten av ett tal med valfritt antal decimaler, dock högst 20. Programmet är självinstruerande och lätt att använda, även för absoluta nybörjare.

PRIM.BAS
Program som delar upp ett tal i primfaktorer. Programmet är intressant eftersom det var en vanlig uppgift för de första datorerna. Programmet klarar av att dela upp tal med upp till 28 siffror. Det tar lång tid att dela upp stora tal.

LÅN.BAS
Hur fördelaktiga är dina lån?

STÅLTEMP.BAS
Du kan undersöka olika legerade ståls behov av förvärmning. Du kan variera stålets sammansättning dimension etc. Du kan variera innehåll av svavel, fosfor och kol samt flera legeringsmetaller.

OHMDATA.BAS
Undervisningsprogram i elektricitetslära. Du tränar begrepp och beräkningar t ex resistans,spänning och effekt.

ABCMINI.BAS
Kräver floppy och modem som extrautrustning dessutom krävs terminal-program ex ABCV24.BAS, vilket finns på ABC-kassett nr 3. Programmet medger mottagning av textfiler från klubbens monitor i Alvik och från liknande utrustning på andra ställen.

CASMINI.BAS
Kräver modem som extrautrustning. Programmet medger överföring av filer från klubbens monitor till din ABC80.

ABCTRANS.BAS
Kräver floppy och modem som extrautrustning. Programmet som nu finns i en version för 80 kolumners skärm, medger filöverföring i båda riktningarna mellan din utrustning och klubbens monitor.

LOGAIN.BAS
Kommunikationsprogram som liknar ABCTRANS och följaktligen kräver modem och floppy. Programmet är dock utformat med tanke på kommunikation med stordatorer av fabrikat DEC -Digital Euqipment Corp. Programmet innehåller exempelvis en automatisk inloggningsrutin. Det som gör programmet intressant är att klubben snart kan erbjuda sina medlemmar möjlighet att utnyttja QZ's DEC10-dator till en starkt rabatterad taxa.

MAKETEXT.BAS
Kräver floppy som extrautrustning. Programmet är en mycket enkel editor. För den som inte vet vad en editor är kan jag berätta att en editor är ett program som gör det möjligt att skriva text på en fil. MAKETEXT kan användas för att skriva brev mata in data i en fil som kan bearbetas av ett annat program.

Du använder MAKETEXT så här:
Ladda in MAKETEXT och skriv RUN. zzz

VISA.BAS
Kräver floppy. Programmet gör det möjligt att på ett bekvämt sätt läsa textfiler rad för rad. Du kan se hur programmet fungerar om du gör så här. Du har VISA.BAS på en diskett! Skriv RUN VISA och ge när programmet svarar VISA.BAS som indata. Du kan nu läsa filen genom att trycka ned ex mellanslagstangenten.

FILSUM.BAS
Programmet beräknar antalet rader och ASCII-summan av tecknen i en textfil på skiva. Det är ofarligt att använda, det är alltså fritt fram för experiment.

FILCOM.BAS
Filhantering på floppy. Programmet jämför två filer. Programmet är ofarligt och kan ej skada några filer. Förutsättningen för att jämförelse ska kunna ske är att filerna är text-filer, vilket innebär att program måste sparas med LIST för att kunna jämföras. Du kan prova FILCOM på följande sätt. Ladda in FILCOM med LOAD och spara sedan med LIST XYZ. Du får då en version av programmet under namnet XYZ.BAS. Du kan nu köra FILCOM, skriv RUN och ge FILCOM.BAS respo XYZ.BAS som indata till programmet. Om filerna är lika så får du meddelande om detta. Skulle det vara någon skillnad skrivs de rader som är olika ut.

CONCAT.BAS
Kräver floppy som extra utrustning. Programmet konkatinerar filer, dvs slår samman två eller flera filer till en. Det kan du ha användning för om du skrivit text med MAKETEXT och vill lägga två text-filer efter varandra i samma fil.

Varning!
Tänk på att det namn du ger din utfil måste vara skillt från det namn dina in-filer har. Är namnet du ger utfilen upptaget förstörs den filens innehåll.

SORTCORE.BAS
SORTCORE sorterar en textfil internt i RAM-minnet. Programmet är skrivet för flexskiva men fungerar även för kassett.
Först frågar programmet efter infil. Se till att den textfil som skall sorteras ligger klar att läsas på flexskiva eller kassett.
Därefter frågas efter recordlängd, d v s den maximala radlängden som posterna i infilen har.
Sedan frågas efter "sorteringsnyckel(startposition,längd)". Svara då med två tal separerade med komma. Det första talet skall vara första position i det fält man vill ha raderna sorterade efter och det andra talet fältets längd.

Exempel:ett adressregister där varje rad har fyra fält, namn (efternamn förnamn), gatuadress, postnummer, postadress, med fältlängderna 20, 25, 5 resp 13.
Recordlängden blir då 63.
Vill man sortera alfabetiskt efter efternamn anger man sorteringsnyckel 1,20 och vill man ha det i postnummerordning anger man nyckeln 46,5.

Programmet talar om hur många records, d v s poster som kan sorteras. Får du ERR 3 så minska värdet på N% i rad 30 och försök igen.

Programmet frågar sedan efter utfilens namn. Trycker man bara RETURN får utfilen samman namn som infilen.

När textfilen lästs in från flexskiva (eller kassett) stannar programmet och väntar på tangentnedtryckning. Om man vill kan man nu passa på att byta skiva (bra när utfilen fått samma namn som infilen och man inte vill få den överskriven). Använder man kassett så gör man nu bandspelaren klar för inspelning innan man trycker tangent. Därefter startar sorteringen. När den är klar skrivs sorteringsstatistik ut och den sorterade utfilen skrivs på flexskiva resp kassett. När det är klart skrivs utfilens namn ut.

KONVERT.BAS
Program som konverterar vissa tecken i en fil. Programmet verkar föga andvändbart men visar hur man kan lösa ett problem, som är generellt.

DÖP.BAS
Programmet används för att märka skivor med volym-namn.

IDMÄRK.BAS
Märker eller 'döper' en floppy. Liknar alltså programmet DÖP, men kan dessutom kolla eventuell märkning av skivan.

CKOLON.BAS
Lib-program för floppy. Programmet är skrivet i maskinspråk och mycket snabbt. Du startar programmet med RUN CKOLON varvid programmet går igång och snabbt ger Libben plus resterande utrymme på floppyn i drive 0 och 1. Har man många filer på drive 1 så att filerna på drive 0 hinner rulla förbi kan man ge kommandot RUN C:0 . Programmet fungerar nämligen så att en enhet med namnet C: definieras.

TIME.BAS
Program som sätter systemtid. Programmet gör CHAIN till programmet MENY, vilket ingår i kassett 2.

KASSETT NR 2

CASDISK.BAC
Program som kopierar filer från band till disk. CASDISK kopierar alla filtyper såväl text-filer som ABS- och BAC-filer. Du som använt CASTODSK för att kopiera från band till disk bör övergå till CASDISK för att slippa obehagliga överaskningar.

CASLIB.BAS
Programmet som går på grundutrustning ger en snygg utskrift av biblioteket i två spalter. CASLIB läser blockvis och kan alltså klara alla filer.

MASKEN.BAS
Programmet som nått herostratisk ryktbarhet,kanske inte främst på grund av märkvärdiga egenskaper, utan fastmer på grund av den beramade 'stölden'. Masken är dock värd en bekantskap, så prova och hjälp masken Max att hitta något att äta! Av speciellt intresse är den repetitionsfunktion som finns i programmet.

LABYRINT.BAS
Ett tredimensionellt labyrintspel i två dimensioner.

GOMOKU.BAS
Ett luffarschackprogram. Vinner alltid (?).

Kassett två

INVASION.BAS
Spelprogram, luftangrepp, med möjlighet att slå tillbaka.

HINDLOPP.BAS
Cursorn försöker ta sig över skärmen och möter uppdykande hinder.

DECROM.BAS
Programmet omvandlar decimala tal till romerska. Det är enkelt att använda och allmänbildande.

BLINKA.BAS
Mer musik i ABC80, liknar SLOOP i programmeringshänseende.

GUBBENOA.BAS
Ännu mer musik i ABC80!

NYMEMDMP.BAS
Disassembler för ABC80. Programmet som har en mängd funktioner är mycket användbart för den som vill lära mer om maskinspråk.

EXTDEL.BAS
Program som gör delete eller extract av rader i ett annat program. Programmet finns beskrivet i nr 1 1982 av ABC-bladet. Där finns också källkoden till de maskinspråksrutiner som används i programmet.

HUVUD.BAS
Detta program ställer en rad frågor som du bör besvara varje gång du skrivit ett program. Resultatet blir ett programhuvud som du kan få ut på disk eller kassett.

MENY.BAS
Dynamiskt meny-program tyvärr bekajat med något fel som får datorn att låsa sig. Programmet är ett exempell på vild programmering med sådana läckerheter som inhopp i loopar, flera subrutiner efter varandra utan föregående END osv. Vi har dock inget att invända mot programmiden som sådan och ser den gärna fullföljd på ett strukturerat vis om någon känner sig hågad.

TIME.BAS
Sätter systemtid och gör CHAIN till MENY.

LÄSDISK.BAS
Blockvis läsning på disk, du väljer drive, spår och sektor. Ett bra program för dig som vill veta hur lagringen på flexskiva går till. Programmet kan dessutom knappast skada dina skivor så var ej orolig för att pröva.

FILOMV.BAC
Minne 16 Kbytes för flexskiva Programmet omvandlar en BAS-fil till BAC-fil eller tvärt om. Typiskt bra om man har lagrat ett program med list, men råkat få en rad längre än 117 tecken! Du som är osäker på vari skillnaden mellan BAS och BAC filer består bör dock först inhämta denna visdom.

DISKST.BAS
Programmet som egentligen borde ha hetat DISKSTAT

REGISTER.BAS
Sekvensiellt registerprogram med utskriftmöjlighet.

DIRFIX.BAS
Räddar libben vid haveri.

OMVANDLA.BAS
Programmet kan köras på grundutrustning och är ett omvandlingsprogram mellan olika talsystem. Omvandling kan ske mellan binära, oktala, decimala och hexadecimala tal.

TNSET.BAS
Program av (c) Göran Sundqvist tel.0756/-30310 Minne 16 Kbytes och skrivare Testutskrift av alla tecken på en skrivare, med uppställning i ASCII-tabell.

SNYGGBAS.BAS
SNYGGBAS ÄR ett program för vettigare utskrift av BASIC-programmen än vad ABC-80 's "operativsystem" bjuder på ! Filen, som skall skrivas ut, lagras i ASCII-format med LIST-kommandot.

COPYLIB.BAC
Detta program ersätter ABS-programmet med samma namn och ger dessutom möjlighet att kopiera blockvis från disk till kassett. Programmet är viktigt eftersom det är en förutsättning för att vi ska kunna distribuera alla slags program på kassett. Tillsammans med CASDISK blir det ett paket som gör alla ABC80 med bandspelare kompatibla.

COPY.BAC
Program för kopiering av filer på disk. Programmet ersätter COPYLIB då enstaka filer ska kopieras mellan DR1: och DR0: eller när en fil ska kopieras med nytt namn på samma skiva. Observera att programmet ej innehåller något skydd mot att du 'bombar' en fil genom att ge samma namn åt utfilen som en redan existerade fil.

LIB.ABS
LIB-program i assembler att köras under det speciella operativsystemet för ABS-program, som du laddar in med BYE. Därefter kan du ge kommandot LIB .

DOSGEN.ABS
Detta dosgen-program bör användas av dig som har enkel densitet annars kan du få problem vid ombyggnad till dubbel densitet. Det kan exempelvis visa sig omöjligt att läsa gamla disketter som formaterats med det gamla dosgen.

TV.TXT
Innehåller kort information om TV-editorn med förteckning över vanliga kommandon.

TV.BAC
Detta program startar TV-editorn i floppyversionen för 40 kol skärm. TV.BAC läser in TVSUBR och gör CHAIN till TVMAIN, alla tre filerna måste alltså finnas på disken.

TVMAIN.BAC
Huvudprogram i den s k TV-editorn och finns i flera versioner, varav den som ligger på ABC-kassett nr 2 är den första. TVMAIN måste sparas i BAC- format eftersom programmet innehåller en funkton på rad 10 som ej kan återläsas om programmet sparas med LIST.

TVSUBR.ABS
Masinspråksrutiner i ABS-format för TV-editorn, vilket innebär kompakt lagring av maskinspråk jämfört med DATA-satser och i all synnerhet POKE-satser. TVSUBR måste kopieras till skiva med CASDISK, det går ej med LOAD eller CASTODSK.

Kassett tre

CASDISK.BAS
Samma program som på kassett nr 2 men nu i BAS-format.

MUSIK.BAS
Orgel av tangentbordet, tyvärr med ganska svagt ljud. Det bättre sätt att få musik ut ur ABC80 vi hoppas återkomma på en senare kassett.

LANTERNA.BAS
Här får du lära dig vilka ljus olika farkoster ska föra i mörker till havs. Programmet prövar dina kunskaper genom att visa olika farkosters ljus som de syns i mörker. Din uppgift blir att gissa vilket slags farkost det är fråga om. Eftersom du inte vet vilken kurs de fartyg du möter har kan det vara nog så svårt. Du har dock hjälp av en programfunktion, du du kan rotera de farkoster du ser. Tyvärr bara i steg om en grad. Detta kan du dock ändra på genom att ändra värdet på variabel z

UBÅT.BAS
Stridsvagn, ubåt med torped, vanliga båtar med sjunkbomber.

BRAMUSIK.BAS
Musik på det vanliga viset som du känner igen från tidgare kassetter.

ELMÄTARE.BAC
Ett program!

CHECK.BAS
Med det här programmet kan du testa din ABC80's fuktioner. Programmet har följande möjligheter; tangentbordstest, bildskärmstest, rom-test, kassett-test och printer-test.

ASSIGN.BAS
Detta program visar enhets-tabellen och gör det också möjligt att ändra enhetsnamn.

CASPROMP.BAC
Registerprogram för kassett som kan användas för t ex medlemsregister i mindre föreningar, men naturligtvis också för personligt bruk som att hålla ordning på grammofonskivor och dyligt.

CASBLOCK.REM
Text som beskriver programmet CASBLOCK.

CASBLOCK.BAS
Program som kan rädda en fil med för långa rader ERR 20. Handhavande är ganska omständligt men lärorikt, vi hoppas att det manar till efterföljd till CAS-fantomernas båtnad.

MORSE.BAS
Program för den som vill träna telegrafi. Programmet påverkar V24-porten och kräver viss extra hårdvara t ex en summer ansluten till V24-porten. Närmare beskrivning finns på annat ställe i tidningen.

ABCV24.BAC
Här får du en printer-terminal rutin som gör kommunikation via V24-porten. Programmet simulerar en 80 kol skärm och kommer med en ny version av ABCTRANS att kunna köras på ABC80 med 80 kol som 80 kol terminal möjlig.

CASSEND.BAS
Kassettprogram för filöverföring till exempelvis klubbens monitor.

TVEDIT.REM
Textfil med information om TV-editorn, återfinns utökad på sid 6-7 i ABC-Bladet nr 1 1982.

TVCAS.BAC
Program för start av TV-editor på kassett, TVCAS gör CHAIN till TVSUB och TVMAIN i nu nämd ordning.

TVSUB.BAC
Motsvarar TVSUBR.ABS i floppy-versionen av TV-editorn.

TV80.BAC
Motsvarar TV.BAC men fungerar på 80 kol skärm i floppy-versionen.

TVMAIN2.BAC
Utvecklad version av TVMAIN på kassett nr 2. Denna version kräver mer minne än den första versionen och bör ej användas av den som bara har 16 k. En nyhet är möjligheten att förflytta textstycken, som införts på denna version.

RADJUST.BAS
Program för justering av radlängd i en textfil inför utskrift. Medger utskrift med rak högermarginal på icke-proportionell skrivare. Tillsammans med TV-editorn ger RADJUST medlemmarna ett kraftfullt ordbehandlingsprogram utan stora kostnader.

REGCASPR.BAS
Registerprogram för kassett.

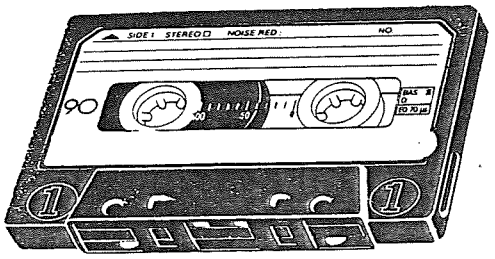
DISASEM.REM
Textfil med information om disassemblerprogrammet som kan skrivas ut på printer eller läsas med TV-editorn. För den som inte vet vad assembler och maskinspråk är kan det hela förefalla mycket mystiskt. Särskillt sedan man startat programmet! Tyvärr kan vi inte här ge en fullständig kurs i maskinspråk, utan måste återkomma i ämnet på annan plats. Var dock inte alltför orolig det finns proffs som inte kan maskinspråk.

DISASEM.BAS

OPLIST.BAS
Detta program skriver ut en lista på opcoder och mnemoniks.

OPCODE1.TXT
Denna fil innehåller de 253 första opcoderna med mnemo-koder. Om du är intresserad av dem kan du läsa filen med VISA som fanns på kassett ett.

OPCODE2.TXT
Denna fil innehåller mnemoiks för de opcoder som kräver mer än en byte



SVENNE

Många har undrat vem Svenne är. Man har gissat hit och dit. En del har gissat på 495, men det är fel. Det är inte heller 1352 och naturligtvis inte 681! För övrigt kan jag avslöja att Svenne inte alls heter Svenne. Han vill bara vara anonym.

Nå till saken. Härom kvällen när jag stod med tandborsten i högsta hugg så ringde Svenne. Jag visste det innan jag svarade. Telefonen låter på ett alldeles speciellt sätt när Svenne ringer. Naturligtvis hade Svenne problem, jätteproblem, och bara jag kunde lösa dem, påstod han. Jag vet av erfarenhet att när Svenne börjar med att smickra så är det besvärliga frågor att vänta. Jag la alltså ifrån mig tandborsten och satte mig.

Svenne:
Jo du förstår jag har läst om maskinspråk och undrar hur man gör egentligen.

K-Å:
Jasså inget annat!

Svenne:
Jag har hittat ett maskinspråkprogram i en Amerikansk tidskrift. Det är för Z-80, så det ska väl gå på ABC80, eller hur? ABC80 har ju Z-80 processor.

K-Å:
Du har rätt ABC80 har Z-80 processor!

Svenne:
Just det, och nu ska du tala om för mig hur jag knappar in det här programmet!

K-Å:
Vad är det för ett program och hur ser det ut?

Svenne:
Det är en Pascal-kompilator. Det ser ganska lattjo ut! Men det är ju maskinspråk! Nu vill jag veta hur man gör jag får bara ERR 11.

K-Å:
Vill du läsa upp något ur programmet.

Svenne:
Hjärtans gärna! Här till exempel LD HLK, EF10H! När jag skriver det får jag ERR 11, vad gör jag för fel?

K-Å:
Du gör flera fel. För det första ska du inte skriva någonting av det där programmet för det förstår inte ABC80. Anledningen är att programmet är ett s k källprogram. Det är endast en källa till ett maskinspråksprogram. Det där som ser så lattjo ut måste översättas för att bli maskinspråk

Svenne:
Jamen det måste väl gå att få in det på något sätt.

K-Å:
Både ja och nej. Den där Pascal-kompilatorn kommer inte att fungera på ABC80 om den inte är skriven för ABC80.

Svenne:
Ja . men det är ju samma processor! Ett maskinspråksprogram för Z-80 ska väl fungera på alla maskiner som har Z-80!

BREV

Följande brev fick vi 5 maj 1982 med kommentaren att LUXOR ej svarat och att ABC-bladet fick publicera det

Bernhard Granholm
Ljunggatan 39 C
664 00 GRUMS

GRUMS 1982-03-28

Följande systematiska fel har upptäckts på ABC80. Om kod för DATA-rad finns i variabellistan så kan datorn uppfatta denna som början på en DATA-rad och läsa följande tecken med en READ-sats.

Här är en beskrivning på det inträffade, alltså hur det kan fungera i ett program. I bifogat utdrag ur programmet finns på rad 220 en ONERRORSATS. Av bekvämlighetsskäl har jag satt loopen med READ-sats med fler omlopp än vad som finns variabler i DATA-raderna (utrymme för fler variabler förberett. När DATA-raderna är slut kommer ERROR 30 vilket tas om hand av ONERROR-satsen. Detta fungerade utills efter en ändring i programmet det i stället blev ERROR 10 LINE 230 (Texten får ej plats i strängen). Jag dimensionerade upp strängvariabeln och fick så tag i den. Efter CTRL C och manuella kommandon skrev jag den på diskett. Därefter jämförde jag med innehållet i arbetsminnet och hittade "variabeln" i variabellistan med början på adress 38792. Adressen före innehåller decimaltalet 147, som är koden för DATA-rad och i detta fall ingår 147 i pekare till adress 38803 (nästa variabel) (38803 = 147 + 256 * 151). Datorn är försedd med 16k extra arbetsminne.

Felet försvann när jag lade till en rad i programmet, t ex en REM-sats. Felet återkommer varje gång programmet har en längd så att adresspekaren innehåller talet 147.

För jämförelse sänder jag med ett utdrag ur arbetsminnet med en riktig DATA-rad som börjar t ex vid adress 38322 med textbörj an BORLÄNGE fr o m adress 38326).

```
10 REM BGm
20 REM * D 82-03-27
30 REM NDFMNÄT 1982 *
40 DIM C(25%,3%),CB=20%,CB(25%)=20%
50 DIM C%(25%)
60 DIM D(12%),DB(12%)=20%
70 RESTORE 2200
75 READ M
80 FOR I%=1% TO 7%
90 FOR J%=1% TO 3%
100 READ F(I%,J%)
110 NEXT J%
120 NEXT I%
130 J%=0%
140 GOSUB 1240
150 ; CHR$(12%)
160 ; 'Detta program ger störningsprognos'
170 ; 'för FM-nätet i Norra Dalarna.'
180 ;
190 RESTORE 2280
200 GOSUB 1300
210 FOR I%=1% TO 25%
220 ONERRORGOTO 340
230 READ CB
240 ; CB;TAB(15%);
250 GET VB
260 V%=INSTR(1%,'JjYy',VB)
270 IF V%=0% 320
280 ; TAB(15%)'J';
290 J%=J%+1%
300 CB(J%)=CB
310 C%(J%)=I%
320 ;
330 NEXT I%
340 C%=J%
350 GOSUB 1300
360 FOR I%=1% TO J%
370 ; I%TAB(5%)CB(I%)
380 NEXT I%
390 GOSUB 1300
400 ; C%' stationer att ta hänsyn till.'
410 R5%=PEEK(65011%)-J%-3%
420 FOR I0%=1% TO J%
430 V%=10%
440 IF C%(I0%)<8% GOSUB 2460 : GOTO 480
450 FB=CB(I0%)
460 GOSUB 2320
470 GOSUB 2360
480 FOR L%=1% TO 3%
490 IF C%(I0%)>7% 520
```

Programlista i INTERNKOD RAD 2290

Del av program med READ-satsen RAD 230

```
38311  78
38312  44
38313  71
38314  92
38315  82
38316  68
38317  65
38318  76
38319  69
38320  78
38321  13
38322  31
38323  242
38324  8
38325  147
38326  66
38327  79
38328  82
38329  76
38330  91
38331  78
38332  71
38333  69
38334  44
38335  72
38336  92
38337  76
38338  74
38339  69
38340  83
38341  44
38342  84
38343  82
38344  65
38345  78
38346  83
38347  84
38348  82
38349  65
38350  78
38351  68
38352  13
38353  16
38354  252
38355  8
38356  147
38357  82
38358  92
38359  82
38360  66
```

Variabeln skriven med METRIC 8500

151 94 153 103 0 5 0 25 0 3 0 242 67 157 151 160 0 192 155 9 0 246 67 167 151 6 156 25 0 6 0 245 67 177 151 170 158 25 0 2 0 244 68 187 151 222 158 12 0 5 0 24 6 68 197 151 31 159 12 0 6 0 240 77 206 151 16 112 0 0 130 241 73 222 151 14 0 2 46 128 210 151 173 128 7 0 1 0 241 74 238 151 0 0 232 128 226 151 195 128 3 0 1 0 248 70 6 152 113 160 120 0 5 4 10 0 10 0 105 130 210 151 153 129 25 0 1 0 242 86 16 152 80 0 206 162 1 0 241 86 22 152 0 0 241 67 38 152 0 0 202 130 210 151 1 57 130 0 0 1 0 81 82

Variabeln i ASCII-kod

```
38780  66
38781  90
38782  152
38783  212
38784  185
38785  58
38786  145
38787  13
38788  1
38789  248
38790  67
38791  147
38792  151
38793  94
38794  153
38795  103
38796  0
38797  5
38798  0
38799  25
38800  0
38801  3
38802  0
38803  242
38804  67
38805  157
38806  151
38807  20
38808  0
38809  102
38810  155
38811  9
38812  0
38813  246
38814  67
38815  167
38816  151
38817  122
38818  155
38819  25
38820  0
38821  6
38822  0
38823  245
38824  67
38825  177
```

147 tolkas som början på DATA-rad

38803

Pekar dit (Nästa variabel)

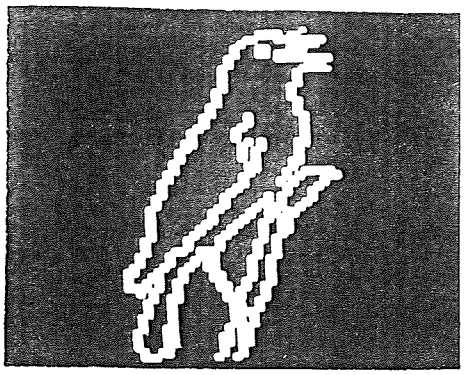
Variabellistan

```
2200 DATA 10.7
2205 DATA 92.2,96.7,98.95
2210 DATA 90.6,94.9,97.9
2220 DATA 88.9,93.7,95.7
2230 DATA 88,91.1,93.4
2240 DATA 91.4,96.2,99.7
2250 DATA 90.2,94.3,98.65
2260 DATA 89.2,92,97.55
2270 DATA 32078,31775,32272,32007,31745,32258,31787
2280 DATA MORA,SVEG,SÄRNA,IDRE,GRÖVELSJÖN,DREVDAGEN,GÖRDALEN
2290 DATA BORLÄNGE,HÖLJES,TRANSTRAND
2300 DATA RÖRBÄCKSNÄS
2310 DATA ÄNGE,ÖSTERSUND
2320 REM * FILNAMN
2330 IF LEN(FB)>8% FB=LEFT$(FB,8%)
2340 FB=FB+'.DAT'
2350 RETURN
2360 REM * LÄSER FRÅN FIL
2370 ONERRORGOTO 2440
2380 OPEN FB ASFILE 3%
2390 INPUT #3%,N8
2400 FOR I%=1% TO 3%
2410 INPUT #3%,B(I%)
2420 GOSUB 2530
2430 NEXT I%
2440 CLOSE 3%
2450 RETURN
2460 REM *
2470 FOR I%=1% TO 3%
2480 B(I%)=F(V%,I%)
2490 GOSUB 2530
2500 NEXT I%
2510 N8=B8(V%)
2520 RETURN
2530 ; CUR(V%+R5%,I%*7%+9%)B(I%) ; RETURN
```

Del av programmet med DATA-radesna

Vid förfrågan i början av september hade inte L.svarat ännu. Samt "Jag har provkört programmet också i en ABC80 med normalt minne.Det händer samma sak".BG..

claes s.



- ett grafikprogram med ljud för nybörjare

av Ferdinand Mican

Pausbird

När jag såg programsnutten "Pausfågel" då frågade jag mig varför ingen hade illustrerat ljudet.

Två flugor i en smäll eller två käpphästar i glad förening, nämligen programmering och fågelskådning. Nej, förresten var det många fler flugor. Samtidigt kunde jag ju lära mig lite om grafik på min mikrodator. Att döva mitt samvete som ständigt säger att jag inte borde hänga bakom tangentbordet halva natten, det lyckades jag däremot inte med. Även om jag försökte efterlikna en av faunans trevligaste varelser.

Jag började med att skaffa mig kopior av Videografik- kartan, som finns i bruksanvisningen för ABC-80. Sedan ritade jag in konturerna på min fågel, som hör till familjen Sturnidae. Starar kan nämligen härma alla ljud, t.o.m. högtalarens gnissel som lätt kan förväxlas med en fönsterputsares arbetssång. Nästa steg var ifyllandet av samtliga punkter som hade drabbats av min blyertspenna. Nu kunde jag se hur bilden skulle se ut på rutan.

Att snabbt komma igång med knappandet, det är freakens melodi. (Freak uttalas fri:k. Det är engelska, förstås, och beteckningen för datorhobbyister. Betyder egentligen hugskott eller "originell människa".) Proffsens JSP-strukturering behärskar jag inte så jag nöjer mig med att repetera avsnittet om grafik i "ABC om BASIC". Om Du undrar vad "JSP" är så kan jag tala om att det är en programspråks-oberoende konstruktionsmetod och förkortningen för Jackson structured programming.

I ABC-boken står det i alla fall att man först sätter hela skärmen i "grafisk mode". Det gör jag på raderna 110 och 120. Sedan försöker jag rita med en teknik som tillåter ASCIItecknen ett bestämt mönster om skärmen är i grafisk mode.

Snart upptäcker jag att det är ganska mödosamt att omvandla alla prickar till bokstäver och konstaterar att denna metod bäst ägnar sig för stora teckningar där en stor del av skärmen är täckt av "linjer". Nej, då är det bättre med punkter i ett koordinatsystem. Det är klart att vill man ha hela skärmen vitt så skriver man inte 5.616 gånger SETDOT(R,K) om man kan omvandla straffet i böter. På hela skärmen finns det nämligen så många grafiska punkter om man räknar ifrån vertikal- rad 0 där omvandlingen till grafisk mode osynligt ligger. R och K är siffrorna i det koordinatsystem som gör att man kan nå samtliga punkter på bildskärmen. Som Du kan se på bild 1a så finns det 78 horisontella lägen och 72 vertikala. De första två lägena är nämligen reserverade för att kunna sätta skärmen i grafiskt mode.

Nu kan jag väl vara nöjd? Men det är med program som med en trädgård - den blir aldrig färdig. Det är så mycket man kan göra annorlunda och bättre. Programmet tar onödigt mycket minne. Genom att konsekvent använda heltal kan man spara 4 byte på varje siffra. Näbbens rörelse i ljuddelen har dragit ner hastigheten och jag måste skruva upp kråkan igen så att det blir lite fart på kvittret. Sedan kan man väl lägga in klockan i bilden och .....

Nej, nu får det räcka för den här gången. Nästa bild ritar jag med en helt annan teknik. T.ex. med programmet RITA som jag som medlem i ABC-klubben har fått. I detta program kan man med olika tangenter styra markören över bildrutan och på detta sätt rita en bild. Kanske morgondagens konstnärer slänger pensel och staffliet och köper sig ett äpple och en fröjdepinne.

Hjälp, det tar tid det här. Jag försöker snabba upp ritandet med hjälp av ED-kommandot. Jag får förstås ändra radnumret och siffrorna om jag vill få glädje av den editerade raden. Nu skulle man behöver en mänsklig medhjälpare som kan läsa upp siffrorna.

Ibland får man göra RUN för att kolla att man har knappat in rätt siffra. Samtidigt är det en liten belöning för mödan när man ser hur bilden växer fram.

Efter några timmar är jag då färdig med denna del av programmet och kan fortsätta med ljudet. I enlighet med god programmeringssed så skriver jag REM för ljudet så att jag inte kan glömma vad det är för konstiga saker som jag har skrivit. Sedan skriver jag looparna som bestämmer ljudet med OUT 6,5 och pauserna med OUT 6,0. Nu provar jag lite olika hastigheter för att lista ut vilket kvitter som blir bäst. Eftersom allt låter lika bedrövligt så skriver jag dit en siffra och bestämmer mig för att vara nöjd. Jag gör RUN igen. Pippin kommer fram och kvittrar och egentligen borde jag vara nöjd men jag tycker att det saknas något. Pricken över i:et! Vad är detta? Den rör ju inte på sig, den är död den. Det är klart att den ska röra på näbben och flämta och flaxa. Men var någonstans i programmet lägger jag in denna rörelse så att den rör sig med samma hastighet som ljudet. I ljudets loop förstås. Där lägger jag in ett annat läge för näbben och nu tänder sig näbben med SETDOT och CLRDOT i överläge och undre läge. Flämtandet med bröstet får jag på veterinärens inrådan ta bort - det ser sjukligt ut.

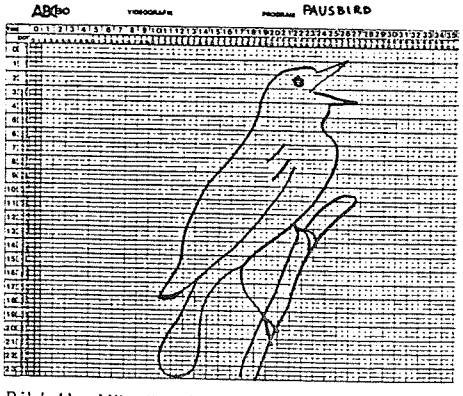


Bild 1b: Här är fågeln inritad.

Exempel på olika sätt att adressera bildskärmen
CURSOR, SETDOT direkt och med READ DATA SETDOT.

```
 10 REM SAVE PAUSBIRD.GRF
 20 REM
 30 REM ************************
 40 REM * PAUSFÅGEL FÖR ABC-80 *
 50 REM * AV FERDINAND MICAN   *
 60 REM * TEL. 0510-623 36     *
 70 REM * REV. 820326-1        *
 80 REM ************************
 90 REM
100 ; INP(3);CHR$(12%)
110 FOR I%=0% TO 23%
120 ; CUR(I%,0%);CHR$(151%);
130 NEXT I%
140 ; CUR(1%,21%);"p,18f"
150 ; CUR(2%,20%);"6"
160 ; CUR(2%,22%);"ö,!"
170 ; CUR(3%,19%);":"
180 SETDOT 9%,47% : SETDOT 9%,48% : SETDOT 9%,4
    9%
185 SETDOT 9%,50% : SETDOT 9%,51%

660 READ R,K : SETDOT R,K
670 DATA 55,24,55,26,55,30,55,38,55,40,55,44
680 DATA 56,26,56,29,56,30,56,38,56,39,56,40,56
    ,44
690 DATA 57,25,57,29,57,39,57,43
700 DATA 58,25,58,29,58,39,58,40,58,43
710 DATA 59,25,59,29,59,39,59,40,59,41,59,42
720 DATA 60,25,60,29,60,39,60,42
730 DATA 61,24,61,29,61,39,61,40,61,42
740 DATA 62,24,62,28,62,29,62,38,62,40,62,41
750 DATA 63,23,63,28,63,29,63,38,63,40,63,41
760 DATA 64,23,64,28,64,29,64,37,64,38,64,40,64

860 REM ********************************
870 REM *   KVITTER OCH RÖRELSER       *
880 FOR K=100*RND TO 100*RND
890 OUT 6,5
900 REM
910 REM ÖVERNÄBBEN-ÖVRE LÄGE TAS BORT
920 CLRDOT 4,49 : CLRDOT 4,50 : CLRDOT 3,51
930 CLRDOT 3,52 : CLRDOT 4,52 : CLRDOT 5,51
940 CLRDOT 6,50 : CLRDOT 7,49 : CLRDOT 7,48
950 CLRDOT 5,50 : CLRDOT 3,53 : CLRDOT 3,51
960 REM
970 REM ÖVERNÄBBEN-NEDRE LÄGE SÄTTS
980 SETDOT 5,49 : SETDOT 5,50
990 SETDOT 5,51 : SETDOT 5,52 : SETDOT 5,53 : S
    ETDOT 5,54 : SETDOT 6,53
 995 SETDOT 6,52 : SETDOT 6,53
1000 SETDOT 6,52 : SETDOT 7,49 : SETDOT 7,51 : S
    ETDOT 7,51 : SETDOT 7,50
1005 SETDOT 8,48
1010 REM
1020 REM NEDRE NÄBBEN-NEDRE LÄGE TAS BORT
1030 CLRDOT 11,52 : CLRDOT 11,53 : CLRDOT 11,54
    : CLRDOT 11,55
1040 REM
1050 REM NEDRE NÄBBEN-ÖVRE LÄGE SÄTTS
1060 SETDOT 9,52 : SETDOT 9,53 : SETDOT 10,54 :
    SETDOT 9,54 : SETDOT 9,55
1070 FOR I=0 TO K : NEXT I
1080 OUT 6,0
1090 FOR I=0 TO K : NEXT I
1100 REM
1110 REM ÖVERNÄBBEN-NEDRE LÄGE TAS BORT
```

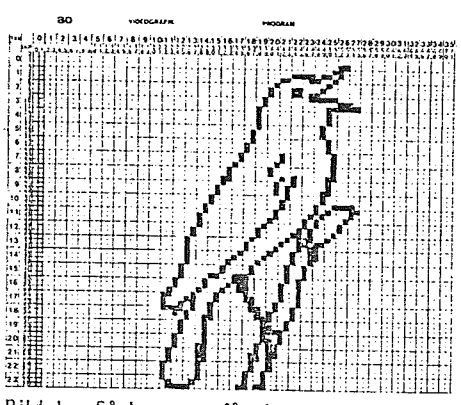


Bild 1c: Så kommer fågeln att se ut.

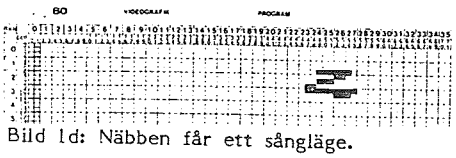


Bild 1d: Näbben får ett sångläge.

Insändare

Hej

Jag har beställt den utmärkta disassembleringen av ABC 80:s programvara. Där har dock smugit sig in ett fel bland 'lablarna'.

Felet finns på sidan 240 och det är log och log 10 som har förväxlats. Sidan 240 skulle ha sett ut såhär:

```
LOG       45:185     LD   BC,60:196
          45:188     JR   45:198
LOG 10    45:190     LD   BC,60:183
          45:193     JR   45:198
```

Genom att stryka över det gamla LOG 10 och fylla i 10 på det nya.

Felet beror troligen på att när man skulle fylla i 'labelarna' så förväxlade man dom, vilket är ganska lätt beroende på att dom ligger omvända i funktionstabellen.

Så här ser funktionstabellen ut:

```
.
.
.
134                 'INT KOD
INT
136                 'LOG 10 KOD
LOG 10
135                 'LOG KOD
LOG
137                 'PI KOD
```

Hoppas att det kan hjälpa någon som vill förstå logaritmberäkningarna.

Hälsningar

2351 Mats Thulin

ÖNSKAS KÖPA

Flexskiveminne till ABC80
Datadisk , Luxor eller likvärdig.

Kurt Perman
Tallstigen 37
915 00 Robertsfors
<medl 1061>
0934-10989 efter kl 16

SÄLJES

ABC80. grundutförande,<11273>, med Facit skrivare säljes för 6.000:-
Tel 0418-15987

Facitskrivaren skriver på tryckkänsligt papper.

Hej !!!

Jag undrar om andra medlemmar som har ABC80 har samma problem som jag har? Problemet är att färgen på kåpan till tangentbordet lossnar. Det är inte bara min ABC80 som färgen har trillat av, jag har sett flera andra.
Det är säkert andra ABC-medlemmar som har samma problem.
Jag tycker att vi kan skriva till Luxor och klaga.

Lars-Ove Jonsson <2499>

SVAR

Efter årsmötet i februari 1981 vid paneldebatten där ledande personer vid Luxor Motala var närvarande kom denna fråga och Bengt Lönnqvist Luxor svarade då att felet låg i plasten kåpan är gjuten av.
Detta har rättats till i senare tangentbordskåpor.
På frågan vad Luxor kunde göra åt problemet med gamla kåpor svarade Bengt Lönnqvist att dessa kunde sändas till Luxor Motala för utbyte mot ny kåpa som skulle tåla handslitaget bättre

Kanske bäst att ringa Luxor och fråga hur och vart att sända kåpan.

Reds Funderingar →

Går man till en radio eller datorbutik för att KÖPA en -i denna fundering - dator och ska betala NY-pris vilket affären antagligen hoppas på - DÅ ska väl datorn vara OANVÄND !

Många i den uppväxande generationen har tydligen inte fått lära sig att inte pilla och peta på det mesta utan att betala för sig. "Prylarna står ju bara där, då gör det väl inget om vi använder dem", är en vanlig kommentar.

Ässå att vara med i någonting, exvis ABC-klubben, nästan så man hör 'kan vi inte få allt gratis utan att betala någonting'.
Till synes förståndiga -vuxna- människor stod på årsmötet och pläderade för väldigt låga medlemsavgifter i klubben.
Vi är ju en ideell förening, men idealiteten får ju inte sträcka sig hur långt som helst.

Avgiften för juniormedlem är ju två - fyra biobiljetter - beror ju på var man bor och hur bra film - och sitter de ändå framför datorn har de ju snart sparat in till klubbavgiften.

Å kommer det insändare å de ska KNAPPAS in ska man inte behöva sitta å ilskna till under inknappningen

claes s.

Falun 820912
Insändare till ABC-bladet

Fantasins nödvändighet - "Hachers"

Hur många missade denna underbara dokumentär om oss unga datoranvändare den åttonde september. Programmet var gjort av Anders Bergman huvudsakligen i Stockholm - Datorhuset, Solna m.fl. platser.
Jag är själv 16 år och går N1 i Falun, har hållit på med datorer sen 1979 någång. Jag tycker programmet var jättebra gjort, det träffade mig verkligen.
När man kommer på någonting mitt i vardagen, och sedan söker efter första bästa lediga stund att testa id'en på datorn. Fungerar det på en gång så blir man nästan förvånad - man glömmer alltid något utfall. Sen kanske man sitter med en likasinnad vid en dator någång och den kommer med sina förslag, man testar och det hela kanske går bättre.
Får man så kontakt med mer erfarna programmerare - får se program som är mer avancerade, ex vis genom ABC-klubben, så vidgas ens vyer enormt. Man börjar upptäcka vad som finns i de innersta delarna av datorn - maskinkod. Därur kommer så småningom de bästa bitarna i programmeringen - assembler m.fl högnivåspråk. Nu är det tyvärr så att de flesta unga programmerare har mycket liten eller obefintlig kontakt med mer avancerade programmerare.

En del av skulden måste nog läggas på ABC-klubben. Vettiga åsikter om kraftigt reducerade priser för ungdomar framkom på årsmötet men tyvärr fick de inget gehör. Det är min förhoppning att åsikterna i denna fråga ändras till nästa årsmöte. Då vi förhoppningsvis beslutar om en ungdomsavgift på ca 25 kronor.

Tyvärr så såg vi vår knappast någon, möjligtvis en ABC80 i programmet. Det tycker i varje fall jag är ytterst dåligt. Programmet var ju helt svenskgjort och då bör man väl försöka få den stora SVENSKA datorn i stället för ca 98% Apple. Tyvärr måste nog Team 100/ABC80:s återförsäljare belastas för detta. Min erfarenhet av dessa är att de alltför ofta är avogt inställda till ungdomar som vill använda datorerna. Undantag finns naturligtvis, men på det stora hela tror jag tyvärr de är förhållandevis sällsynta. Man möts ofta med orden - "Det är för dyrbara grejor för er att hålla på med.Vi har inte tid." Nästan så man hör schas, schas efter sig. Så ser nan sen datorerna stå där dag efter dag och samla damm. Men fantasin är nödvändig både för barn och vuxna. Vi får så hoppas att få med --- fler unga användare i ABC-klubben. Ex.vis då genom kraftigt reducerade ungdomsavgifter och mera vänligt sinnade ABC80 återförsäljare.

Med en stilla förhoppning om en ljusare framtid

Martin Annerbo <2247> Falun

Pausbird kommer på kassett

ABCV24

Köranvisning för ABCV24

ABCV24 som låg på ABC-kassett nr 3 är en specialutgåva av programmet T80PRT från Scandia Metric som ABC-klubben fick i julgåva 1981. På mångas begäran publiceras med benäget tillstånd av Scandia Metric här en bearbetad köranvisning till programmet.

1. Läs in programmet från flexskiva eller kassett precis som vanligt.

2. Programmet presenterar sig med en julhälsning från Scandia Metric. Därefter kommer på skärmen:

Detta program som gör V24-kontakten användbar för seriell ASCII, samt simulerar 80 kolumners bildskärm

Standardvärden (inom parantes) fås genom att trycka på RETURN-tangenten

Du kan välja mellan följande modulkombinationer

```
0.. Utskrift (ej ETX/ACK)   ( 860 bytes)
1.. Utskrift (ETX/ACK)      (1212 bytes)
2.. Utskrift och Inläsning  (1432 bytes)
3.. Utsk + Inl. + Terminal  (4712 bytes)
      ETT ÖGONBLICK!
   Välj 0 - 3              (3)?3
Standardprintertyp       (KB1)?KB1
Max radlängd vid utsk     (80)?80
Controlltecken      (WEFVEPBÅ)?WEFVEPBÅ
1 eller 2 stoppbitar       (1)?1
Enhetens namn           (V24:)?V24:
```

Först frågas efter vilken modulkombination som skall användas (0, 1, 2 el 3).

(Genom att bara trycka RETURN erhålles för modulkombination 3).

3. Programmet frågar efter typ av yttre enhet. Tre olika parametrar skall anges: (H) (N) (B).

Vilket tecken som skall anges för respektive parameter beror på vilka konventioner som tillämpas av den yttre enheten (eller vilken typ av modem som används).

(H) Handskakning/paritet

Yttre enhet	Paritet/paritetsbit MARK	SPACE	UDDA	JÄMN
SKRIVARE/-IN-ENHET (ej EXT/ACK)	A	P	B	C
SKRIVARE/-IN-ENHET Handskakning (med EXT/ACK)	E	D	F	G
ABC-80 som TERMINAL	I	H	J	K

Enbart en av dessa enheter skall anges.

(N) Antal null-karaktärer efter radframmatning (LF) / Half - Full Duplex
(null-karaktärer har ASCII-koden 0).

Yttre enhet	Parameter
	I
SKRIVARE/-IN-ENHET ABC-80 som TERMINAL	I V
Inga null, Full duplex	A
2 null, Halv duplex	B
4 null, Halv duplex	C
6 null, Halv duplex	D
50 null, Halv duplex	Z

Enbart en av dessa tecken skall anges.

(B) Överföringshastighet

Baud-tal	Parametrer
110	0
300	1
600	2
1200	3
2400	4
1200/75	5
75/1200	6

Enbart en av dessa siffror skall anges.

EXEMPEL:
När du ansluter en printer med 1200 Baud anger du t.ex CA3 och vid modem-kommunikation med ABC-klubbens Monitor KB1 (mot DEC-10 KA1).

4. Programmet frågar efter maximal radlängd. Svara med ett tal, som anger den radlängd som du vill ha på printern. Om du satt radlängden till 40 och skriver ut en rad som är 60 tecken lång, så skickar printerrutinen ett extra <CRLF>, dvs Carrige Return + Line Feed när 40 tecken skrivits ut. Du får en utskrift som är skriven med "Wrap around" precis som på ABC80 bildskärm. (Genom att trycka enbart på RETURN erhålles radlängden 80). Genom att ange radlängd 255 så kopplas denna mekanism helt bort och du kan skriva ut hur långa teckensträngar du vill utan att få andra <CRLF> än du själv vill.

5. Programmet frågar efter önskade kontrolltecken. Svara med sju stora bokstäver som svarar mot de kontrolltecken man önskar använda. (Genom att enbart trycka på RETURN erhålles standardvärdena "WEFVEPBÅ"). För kontrolltecknens funktion, se nedan.

6. Programmet frågar efter antalet stoppbitar. Genom att bara trycka RETURN erhålles 1 stoppbit, vilket skall användas utom vid 110 baud.

7. Programmet frågar efter enhetens namn. Svara med en till tre stora bokstäver som skall vara enhetens namn, t ex PR:. (Genom att bara trycka RETURN erhålles namnet V24:). De flesta standardprogram använder enhetsnamnet PR:.

Därefter skrivs "ABC-80" ut och systemet är klart för användning.

OBSERVERA!

När du väl har valt modulkombination 0, 1, 2 eller 3 kan du utnyttja kombinationer med lägre nummer men inte med högre utan att läsa in ABCV24 på nytt. Du kan inte heller ändra max radlängd, enhetsnamn eller antalet stoppbitar.

Däremot kan du lätt byta kommunikationsparametrar. De kan nämligen också definieras vid OPEN, PREPARE eller LIST (av respektive fil). Detta kan vara önskvärt om man vill arbeta med fler än en fil, eller om man vill använda printer efter att ha kommunicerat som terminal.

Kommunikationsparametrarna är desamma, men går man tillväga på detta sätt blir formatet något annorlunda. Efter filnamnet skriver man (H)(N).(B). D v s, om parametrarna är KB1 skriver man: Filkommando "V24:KB.1" ASFILE filnummer.

Anm. ABC-80's kassettrutiner tar ej hänsyn till V24-porten. Detta medför att varje öppning, läsning eller skrivning på kassetten leder till att Null-karaktärer skickas ut på V24-porten. Denna utsändning stoppas först när man exekverar satsen:

OUT 58, INP(58) OR 8

eller när man öppnar V24-filen.

Terminalmodulen (3)

För att initiera ABC-80 som terminal måste man först öppna V24-porten på samma sätt som vid modulkombination (1) och (2) och därefter göra en INPUT-sats till samma fil. Sekvensen blir:

OPEN "V24:KB.1" ASFILE 1 (alternativt enbart V24:) INPUT § 1, A$

När denna sekvens exekveras (det är exekveringen av INPUT-satsen som gör att ABC-80 intieras som terminal), antingen i ett program eller som ett direkt kommando intieras ABC-80 som terminal. Därefter fungerar ABC-80 precis som en vanlig asynkron terminal. Vill man avbryta detta arbetssätt, d v s "koppla" ABC-80 så att den ej längre arbetar som terminal, trycker man ned CTRL-P på tangentbordet. Även datorn kan bryta förbindelsen genom att skicka ett bryttecken (vanligtvis CTRL-B) och därefter godtycklig text som avslutas med en RETURN- instruktion (CR), varvid texten mellan bryttecknet och CR hamnar i A$.

Som BREAK-funktion, för att t ex bryta en pågående programexekvering, använder man CTRL-(E-accent).

Terminalrutinen ger ABC-80 en logisk bildskärm om 24 rader med 80 tecken per rad. Delar av denna kan visas på bildskärmen. Detta styrs av fyra funktioner:

Kommando	ABC-80's bildskärm
CTRL-F	Alla 80 kol för 12 översta raderna
CTRL-V	Alla 80 kol för 12 understa raderna
CTRL-W	De första 40 kol för alla 24 raderna stega 5 steg åt vänster
CTRL-E	De sista 40 kol för alla 24 raderna stega 5 steg åt höger.

Dessa kommandon kan behöva definieras om, ex man måste ha CTRL-W "ledigt" för att kunna skicka CTRL-W till datorn. Man kan då låta ett annat kommando ta över CTRL-W's uppgift vid styrning av bildskärmen så att CTRL-W blir "ledigt".

De kontrolltecken man önskar ändra kan då ändras vid uppstart av programmet i samband med att önskade kontrolltecken efterfrågas (se punkt 5).

Den logiska bildskärmen på 24 rader om 80 tecken per rad som terminalrutninen ger ABC-80 kan, vilket framgår av ovanstående, inte visas samtidigt på bildskärmen. Den lagras därför i RAMminnet. Förutom att använda ovanstående kommandon för visning av den logiska bildskärmen kan man även läsa av den från RAM-minnet med ett program. Beteckningen för detta 80-kolumners minne är SKR:.

Möjligheten att läsa den logiska bildskärmen från RAM-minnet gör att följande program kan användas för att skriva bildminnets innehåll på en printer, ansluten till ABC-bussen.

```
10 OPEN "V24:" ASFILE 1
20 OPEN "SKR:" ASFILE 2
30 OPEN "PR:" ASFILE 3
40 INPUT §1,A$
50 REM MANUELL BRYTNING M H A
   CTRL-P ELLER
60 REM DATORN BRYTER MED STX
70 FOR R=1 TO 24
80 INPUTLINE §2, A$ (R) REM LÄS
   BILDMINNET
90 PRINT §3,A$ (R) REM SKRIV UT PÅ
   PRINTER
100 NEXT R
110 GOTO 40
120 END
```

När det gäller användning av ABC-80 i kommunikation med ABC-monitor eller DEC-10 hänvisas till artiklar om ABC-Monitor i ABC-bladet.

Anvisning för inkoppling av printer och modem har också publicerats tidigare.

ABC-klubben vill passa på tillfället att rikta ett varmt tack till Scandia Metric.

Fortsättning från sidan 9.

Här är Fig Forth

Överföring från kassett till disk

För att lägga upp screens på disketten i det format de skall ligga i användes programmet DOSCREEN som ligger på ABC-kassett nr 4. Programmet överför screens från textfil SCREEN.TXT resp SCREEN2.TXT till Forth-format samt skriver ett directory med filnamnet "Forthscr.een". Läser man det med vanliga LIB så kan man se hur många sektorer dessa screens tar upp. Forthscr.een är ingen vanlig fil, man kan därför inte kopiera den med COPYLIB. För att kopiera till annan diskett måste man använda något program som kopierar från spår till spår i exakt samma form som orginalet.

Så här går det till att från de levererade filerna på kassett göra en systemskiva för FORTH:

1. överför först de aktuella programmen på ABC-kassetten till diskett med hjälp av programmet CASDISK.
2. Formatera en ny diskett och sätt den i drive 0.
3. Sätt skivan med de från kassetten överförda programmen i drive 1 och gör RUN DOSCREEN.
4. Svara på frågan om det redan finns ett screen-directory med tillräcklig plats. Svara N för nej när du gör en helt ny systemskiva.
5. Svara på frågan om det högsta screennummer du vill reservera plats för. RETURN sätter standardvärde (default) till 75.
6. Ange filnamn SCREEN.TXT resp SCREEN2.TXT.
7. Kopiera därefter över FORTH.ABS (eller FORTH32.ABS) samt CMDINT.SYS till den nya systemskivan med hjälp av COPYLIB.
8. Finns det plats kan du lägga in även andra filer t ex LIB, COPYLIB och COPYDISK som kan vara bra att ha på systemskivan.

Om du redan har en systemskiva med tillräklig plats bör du svara J för ja i punkt 4. Frågan i punkt 5 hoppas då över och övriga programfiler kommer att vara oförändrade varför du inte behöver göra kopieringen enligt punkterna 7 och 8. Vill du utöka utrymmet för screens kommer alla gamla screens som ej skrivs över av nya att vara kvar oförändrade men du måste kopiera in övriga programfiler enligt punkt 7 och 8.

För att köra FORTH sättes systemskivan i drive 0 och du skriver BYE. Därefter ger du kommandot FORTH16 eller FORTH32 om du har extra minne och vill använda den versionen.

FORTH är nu **klar att använda!**

Ett MEN bara !!

ANVÄND UPPER CASE !!!
= Lampan skall lysa!!

Dokumentation till programvaran finns i ABC-klubben rapport nr 2: "Fig-FORTH on ABC-80, Implemented by Robert Johnsen", jfr Ref 3.

Rapporten som är skriven på **engelska** omfattar c:a 50 sidor och innehåller bl a instruktionsmanual för att köra FORTH på ABC-80, screen-beskrivning, referenslista samt utdrag ur Fig-FORTH Installation Manual med Glossary (beskrivning av Forth Words) samt en beskrivning av Fig-Forth Editor. Rapporten kostar 60 kr totalt inkl frakt och beställes skriftligt från:

ABC-klubben, Vidängsvägen 1, 161 33 Bromma

och levereras mot postförskott. Det går också bra att betala in 60 kr på ABC-klubbens postgirokonto nr 15 33 36-3. Skriv Rapport nr 2 eller Forth-manual på talongen.

För den som vill lära sig FORTH rekomenderas varmt Brodies bok, se Ref 1. Beställes genom bokhandeln.

En **kassettversion av FORTH** beräknas komma på ABC-kassett nr 7. Det är vår förhoppning att programvaran skall komma att utvecklas och anpassas även till andra system såsom ABC-800 samt 8-tums flexskiveutrustning. ABC-klubben ställer gärna källkoden till förfogande till dem som vill göra sådana utvecklingar.

Gunnar Tidner

RÄTTELSER i FORTH

Programmet **DOSCREEN**

Rad 160: satsen efter kolon (:) skall bort.

Rad 260: nummret 255 utbytes mot 256.

GLÄDJANDE NYHET
för alla matematiker !!!!

Stig Rosenlund skrev ett brev att han skänker ABC-klubben sitt matematikpaket vilket han hade annons på i bladet 1-1982

Dessa program har anlänt, och kommer på kassett som vi hoppas inom en snar framtid.

Och när ni utvecklat kring-program med intressanta applikationer inom matematiken är Stig väldigt intresserad av en återföring över resultaten.

claes s.

SÄTT AUTOPILOT PÅ DIN ABC80!

Användarrapport

Nyfors Data Science erbjuder ett programhjälpmedel som man kallar Superbasic. Bakom firmanamnet finns Magnus Lundberg och Magnus Jacobsson. Jag har haft tillfälle att prova ett par varianter.

En huvudidé bakom Superbasic är att ABC80:s BASIC får kommandolistan utökad med ytterligare ett tjog ord som kan beställas direkt från tangentbordet.

Många av dessa kommandon är kända från andra i marknaden förekommande produkter - AUTO (automatisk radnumrering), DEL (ta bort önskat antal programrader), VAR (visa använda variabler) - men funktionerna är ofta utökade.

Med VAR X visas alla rader där variabeln X förekommer - ovärderligt vid programbearbetning.

Ibland vill man ha tag på något som inte är en variabel, t ex REM, END eller en textsträng. Då finns kommandot FIND som gör jobbet på samma sätt som VAR.

LIB och COPY kan utföras direkt utan att påverka arbetsminnet.

Med den inbyggda och alltid inkopplade programeditorn kan markören flyttas till valfri plats på skärmen. Programrader kan editeras, för långa rader delas upp, felaktigt skrivna rader "räddas".

Med VISA <namn> hämtas BASIC-filer och textfiler från flexskiva och visas på skärmen utan att påverka arbetsminnet. Man kan på det sättet hämta rutiner från andra program och kopiera in dem i det program man har just då i arbetsminnet.

Superbasic innehåller en HOLD-funktion som är mycket användbar: shift+CTRL+bokstaven O 'fryser" programexekveringen i de flesta program. Ett andra tryck på samma tangenter släpper loss igen. Om man i stället ger kommandot CTRL-P sker utskrift på printer av skärminnehållet.

Ibland händer oväntade saker och program låser sig. Enda boten har varit RESET och förlust av programmet. Nu kan man efter "frysning" välja CTRL-B, vilket för det mesta frigör låsningen - och RAM-minnet förblir opåverkat. Den funktionen har man ofta anledning att välsigna!

Om man av misstag gjort NEW och blivit av med sitt program, kan man med OLD återställa pekarna och jobba vidare.

Det finns fler små nyttiga och roliga funktioner. Men här finns tre programideer som är alldeles speciella och mig veterligt inte lanserats förut: STEP, EDIT och EXEC.

Med STEP kan ett BASIC-program exekveras stegvis. Varje programrad visas på skärmen innan den utförs. Man kan rätta till upptäckta fel och fortsätta exekveringen med STEP<radnr>. Ofta är detta praktiskt när man prövar en rutin eller letar efter underliga fel.

EDIT är ett ordbehandlingsprogram, eller åtminstone början till ett. Vi har ju tidigare ABC-ORD och TV, och dessa båda har finesser som inte EDIT har. Men EDIT har två intressnta fördelar. För det första finns programmet alltid tillgängligt under hela den tid Superbasic är inkopplad. För det andra kan man behandla i det närmaste obegränsat långa textfiler.

Med kommandot EDIT$<programnamn> skapas en textfil. På skärmen skriver och editerar man sedan sin text. Räknare för bl.a. rad och kolumn visas på skärmen och hjälper till att hålla koll på storleken.

När man är klar ges kommandot CTRL-Q samt EN (för end) och filen skrivs in på flexskivan.

Om arbetsminnet blir fullt innan dess, kan texten efter hand överföras till skivan, men man får då kvar i minnet en skarvbit, så att man kan hålla reda på var man är.

Om man vill editera en redan befintlig fil fungerar det på samma sätt. Programmet läser in så mycket som får rum i minnet. När man arbetar sig genom texten flyttas efter hand överskottet till skivan igen. Vid avslutningen kan man välja om man vill spara den gamla filen (under extension .BKP) eller bara den nya. Det är alltså bara skivans kapacitet som anger gränsen för hur stora textfiler man kan jobba med.

EDIT är inte färdigutvecklat och kommer nog att förbättras åtskilligt i kommande upplagor. Programmet fungerar bäst på 80-teckensskärm. Den version jag har provat visade upp en del barnsjukdomar och tycks inte vara riktigt pålitligt i min maskin. Klart är att programidén har lovande utvecklingsmöjligheter.

Nu har det bästa sparats till sist: EXEC, som kan fungera som autopilot på ABC80 ! I broschyren står att du "kan arbeta i BATCH med JOBSTRÖMMAR där du har GLOBALA PARAMETRAR". Min första tanke var att detta är något för de stora grabbarna och inte intressant för mig. Efter att ha provat har jag ändrat åsikt. Med EXEC kan man i början av körningen och på en gång ge datorn alla de kommandon, instruktioner och frågesvar som behövs för att hålla den i gång ända tills hela jobbet är avslutat.

Ett exempel: Anta att du har fem textfiler som du vill ha utskrivna med utskriftsprogrammet SKRIV, i vilket du vill välja olika värden på radlängd, marginal osv. I vanliga fall får du sitta vid tangentbordet och vänta tills maskinen är färdig med den första texten, innan du kan ge nödiga instruktioner för den att köra nästa text.

Med EXEC skriver du genast ned alla de instruktioner som behövs för jobbet. Sedan kan du gå hem och låta apparaten jobba. Det är också möjligt, för tex återkommande jobb av samma slag att spara jobströmmen i en specialfil. Denna kan t o m kalla in sig själv med olika värden och upprepa sig så många gånger som behövs.

Jag blev helt förbluffad över vad min lilla ABC80 plötsligt kunde åstadkomma. Även om man har en del fallgropar (bl a måste varje kommando vara helt rätt och alla måste komma i rätt ordning) är jag ganska säker på att det kommer att vara en användbar funktion.

Hur får man nu in denna Superbasic i sin ABC80? Magnus Lundberg erbjuder varje kommando i en särskild modul: köp och betala bara de kommandon du vill ha! Den som nöjer sig med bildskärmseditorn, AUTO, VAR, FIND, DEL, VISA, LIB och några till slipper undan med ca 500 kr och kan få alltsammans på en flexskiva. Superbasicen läses in i RAM-minnet och läggs under taket där den vanligen inte stör. Men den upptar i alla fall något tusental bytes av arbetsminnet.

Skall man ha alla kommandon blir det dyrare. EDIT och EXEC kostar tillsammans 700 kr. De tar också väl mycket plats, även om man väljer att lägga dem i några av dosbuffertarna.

Magnus rekommenderar en dyrare men bättre lösning: Supern läggs i ett par PROM-kapslar på ett expansionskort. Kortet pluggas in på en ledig plats i diskettläsaren. Då finns alltsammans där hela tiden och lämnar det ordinarie arbetsminnet orört!

Jag har provat båda möjligheterna. Den lilla lösningen är utmärkt att börja med när man inte har så gott om pengar. Men min efarenhet var att jag snabbt insåg mitt behov av fler och fler rutiner. Lösningen med PROM-kort är utmärkt, men tyvärr dyrt - kortet kostar också.

Just när detta skrives har jag fått veta att man experimenterat fram ytterligare en metod att placera PROMarna så att man inte behöver något expansionskort. Den har jag inte kunnat pröva.

En sista slutknorr: Om man byter en DOS-kapsel i diskläsaren mot en specialare (kostnad 500 kr) får man tillgång till ytterligare några hjälpmedel som kan vara bra att ha: Dagens datum läses in på en ledig uddaplats i minnet och skrivs sedan automatiskt in på alla nyskapade eller ändrade filer. Dessutom läggs både datum och filstorlek in i biblioteket och visas direkt på skärmen med kommandot BIB (som ersätter LIB).

Det nya DOSet kan kombineras med automatisk programstart och gör även ett par andra saker som man inte direkt märker, men som snabbar upp hanteringen och gör den tillförlitligare.

Sven Wickberg

UTFÖRANDE AV KOMMANDON I BASIC ELLER I MASKINSPRÅKSPROGRAM

Av Mika Helenius
(Översatt från finska av Risto Koivula)
(Källa: Finska ABC-bladet 2/81)

Det händer ofta när man gör ett program, att man önskar ABC-basicen gav möjlighet att utföra kommandon i programmet. Kommandon som RUN, SAVE, LIST, MERGE och NEW skulle i många program vara användbara och ibland direkt behövas. Så gott som alla dessa kommandon kan dock användas med en lämplig CALL-sats.

KOMMANDO	ADRESS HEX	ADRESS DECIM.	FINNS PARAMETER	PARAMETER INNEHÅLL
BYE	017A	378	-	
LIST	0AEF	2799	+	filnamn
LOAD	0BA8	2984	+	filnamn
MERGE	0BA9	2985	+	filnamn
NEW	0A70	2672	-	
RUN	0D4C	3404	+	filnamn
RUN	0D55	3413	-	vanligt
RUN	0D58	3416	-	utan CLEAR
SAVE	0D6C	3526	+	filnamn
UNSAVE	014C	332	+	filnamn

Alla ovanstående kommandon kan anropas via CALL, såväl i Basic som i assembler. Kommandon utan parameter är lättast att använda. Där räcker det med ett maskinspråksanrop, t.ex.:

```
999 Z%=CALL(3413) : REM RUN
```

startar programmet från början, nollställer alla variabler samt stänger alla filer.

Kommando som kräver parameter är litet knepigare. Då fordrar systemet att HL-registerparet innehåller adressen till början på parameterlistan. Parametern måste avslutas med ASCII-tecknet 13 (carriage return). Problemet är att man inte kan sätta HL-registerparets värde vid CALL-anrop.

Nedan följer en enkel Basic-rutin, som genom POKE med början på adress -128 sänder strängen C$ + carriage return.

```
10000 C$=C$+CHR$(13%) : FOR Z%=1%
      TO LEN(C$) : POKE -129%+Z%,
      ASC(RIGHT$(C$,Z%)) : NEXT Z
      % : RETURN
```

Nedan en maskinspråksrutin som laddar HL-registerparet med värdet -128 (parameterns startadress) och därefter hoppar till adressen som finns i DE-registerparet. Eftersom DE-parets innehåll kan bestämmas via ett CALL-anrop, erbjuder dessa rutiner ett sätt att enkelt utföra kommandon.

```
1 POKE -9,213,33,128,255,201
```

Om man antar att rad 1 är körd utförs kommandot

```
LIST 100,300
```

på följande sätt:

```
10 C$="100,300" : GOSUB 10000
15 Z%=CALL(-9,2799) : REM LIST
```

Genom att i stället för talet 2799 skriva adressen för respektive kommando, kan man utföra vilka kommandon som helst, t.ex. filnamn i sambant med RUN-kommandot, eller helt tom.

Om parametern är felaktig, får man under körningen fel, som inte tas om hand av ON ERROR GOTO satser utan programmet stoppas.

Några användbara tips plockade ur den finska ABC-klubbens tidning nr. 1 & 2 -81.

Efter det att ett program stoppats genom ett felmeddelande eller genom CTRL-C, kan man fortsätta programmet utan att förändra variablerna genom kommandot:

```
POKE 65060,0 :GOTO radnummer
```

Kommandot måste ovilkorligen ges på samma rad. Programmet kan inte fortsätta från en subrutin eller från en FOR - NEXT loop. Adressen 65060 indikerar körtillstånd (0) eller kommandotillstånd (1).

```
DEF FNM(X,Y)=-X*(X>Y)-Y*(Y<=X)
DEF FNN(X,Y)=-X*(X<Y)-Y*(Y>=X)
```

Funktionen FNM returnerar största värdet av X och Y, FNN returnerar det minsta. Dessa funktioner är speciellt användbara vid anpassning av HP-2000 program till ABC80 och ersätter då funktionerna MIN och MAX.

Vad gör följande program?? Gissa..

```
1000 POKE -128%,243%,201%
1010 Z%=CALL (-128%)
1020 ; CHR$(PEEK(-528%));
1030 IF INP(56%)<>131% 1020
1040 ; "TRODDE DU.."
1050 GOTO 1030
```

Pröva..

SÄLJES

Radskrivare Scandia-Metrics ITOH
med parallellsnitt o engelsk bruksanvisning.

Sven Wickberg 0750-50456 (kvällstid)
0750-50494 (dagtid)

Stortext

Man har ibland behov av att skriva med större text på datorskärmen än den som är standard. För detta finns ett eller annat program på marknaden, t ex MINDEMO av "the mad programmer" Per Lindberg

Alla har det gemensamt att man skriver bokstäverna med hjälp av en matris, som lagras i datorn med koordinaterna för rad och kolumn.

I MINDEMO innebär detta en formidabel räcka av omfångsrika DATA-satser, som stjäl massor med minne och är mer än lovligt bökiga att knappa in (den som gjort det har också skrivit en kommentar om etta längst ned!).

I ABC-bladet nr2/1981 (sid 22) finns ett uppslag av Niklas Notteman i programmet BOKSTAV, att i stället spara matrisen i form av sju binärtal (i decimal form). Då behöver man bara sju tvåsiffriga tal i ATA-satsen för varje bokstav.

Vitsen är att binärettor ger fylld ruta, medan binärnollor ger tom ruta. På det sättet bestäms rad för rad vilka rutor i matrisen som är fyllda och vilka som är tomma. Matrisen fylls mycket snabbt radvis.

I närskrivna program STORTEXT demonstreras en annan id', som kommer från Gunnar Hasselberg och bearbetats en smula. Den bygger på samma matrisid' som Per Lindberg, men i stället för att ange rad och kolumn som ett tvåsiffrigt tal, och ha komma mellan varje sådant tal, registrerar man talen som tecknet med motsvarande ASCII-värde.

65 skrivs alltså som A, 99 som c osv. Fördelen med detta är att man behåller intrycket på skärmen av att bokstäverna "skrivs" fram, vilket ger känsla av "action", samtidigt som man högst avsevärt kondenrar det behövliga minneutrymmet.Det blir också bekvämare att skriva in programmet, och listan blir mycket mindre omfångsrik.

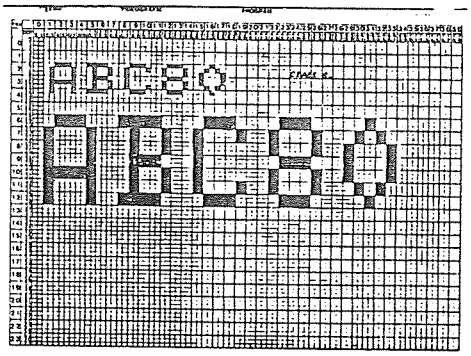
I vårt exempel, som bara visar själva id'n, ges bara tre bokstäver. För att få ASCII-värden inom ett lämpligt område har maten fått radnummer 60-120 och kolumnnumer 1-5.

Vill man vara litet vidlyftigare och i stället har en 9x5 matris (för att få med underslängar) kan man ju ta med raderna 40 och 50.

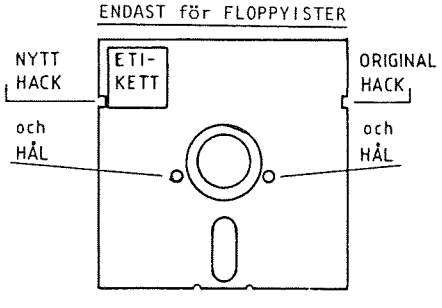
Själva utskriftsslingan behöver också förbättras så att den blir generell och skriver texten i snygga rader, men den övningen överlåtes åt den intresserade. Här visas bara två mycket enkla modeller att ta fram fyllda rutor, med CTRL-> (=ASCII 127) på rad 1200 och med SETDOT på rad 1300.

På rad 1100 har använts R% och K% (heltal) i stället för INT(I/10) osv. I princip bör man ju använda heltal överallt där det går i ett program som detta, eftersom datorn skall 'arbeta själv' så mycket mellan vändorna på tangentbordet, att tidsvinsten blir märkbar.

Sven Wickberg



```
100 REM ...STORTEXT
200 REM Id` av Gunnar Hasselberg
210 REM presentation av Sven Wic
    kberg
215 REM Jobbar med ett rutnät dä
    r
220 REM övre vänstra rutan är 61
    ,
225 REM raderna är tiotalen R%
230 REM kolumnerna är entalen K%
240 REM Talen symboliseras av AS
    CII-
250 REM koderna hos motsvarande
    tecken
260 REM L anger förflyttning i s
    idled
270 REM mellan bokstäver
280 REM Fattas rutin för radförf
    lyttn
290 REM Layouten kan förbättrras
300 N=5 : REM antal tecken att s
    kriva
400 ; CHR$(12)
500 FOR I=1 TO 23 : ; CHR$(151)
    : NEXT I
600 FOR J=1 TO N
700 READ A$
900 FOR I=1 TO LEN(A$)
1000 A=ASC(MID$(A$,I,1))
1100 R%=A/10 : K%=A-10*R%
1200 ; CUR(R%,K%+L)CHR$(127%)
1300 SETDOT R%,(K%+L)+2
1400 NEXT I
1500 L=L+7
1600 NEXT J
1700 ; CUR(19,5)'KLART'
1800 DATA yoeÄQG>?´KU_isåfgh
1900 DATA =>HRÖfpzy?´ÉU^Åisöäz
2000 DATA K´?>GQÄeozäös
2100 DATA ´?>GQÖÅ^isöäzoeUK
2200 DATA ?HQÄepäri_UJ
```



ENDAST FÖR FLOPPYISTER

Vill du få dubbelt utrymme på dina flexskivor? Det finns ett enkelt sätt: utnyttja även baksidorna!

De flexskivor man köper i dag är mestadels likadana på båda sidor. Anledningen till att man endast kan använda ena sidan är det lilla hålet intill det stora centrumhålet (se skiss). När skivan sätts in i skivspelaren måste drivmekanismen ta tag i det stora centrumhålet och läsning/skrivning ske genom det ovala hålet nedtill på skissen. Dessa hål ligger symmetriskt kring mittlinjen (streckad) och fungerar lika bra även om man vänder skivan kring mittlinjen.

Det lilla hålet motsvarar ett ännu mindre hål på själva skivan. Genom dessa hål skall en ljusstråle kunna lysa när skivspelaren är i funktion. Denna del av mekanismen sitter i de flesta enheter nedanför det stora hålet i skivan. Vänder man skivan, kan ljuset inte komma igenom.

Det botas dock lätt om man gör ett motsvarande hål (dvs två - ett på var sida om skivan) i skivfodralet som skissen visar. Detta är inte fullt så enkelt som det låter. Hålen på vardera sidan måste ligga på samma ställe och passa ljusstrålen i apparaten.

Man kan med en vass sax klippa upp en fyrkant på var sida kring det aktuella stället. Det finns risk att man därvid skadar den del av skivan där dataspåren skall finnas. Om skivan är tom gör det inte så mycket om något spår skadas: det tas bort vid formatteringen. Även om man skulle förlora fem spår på varje sida, vinner man 35 spår på baksidan.

Våra tekniskt sinnade läsare kan säkert komma på något smartare sätt att göra hålet. Hör gärna av er!

Medan saxen ändå är framme måste man också klippa ett jack i kanten - den som kommer att vara nedåt när man vänt skivan. Annars kan man ju bara läsa och inte skriva på den nya sidan.

Varsågod, operationen klar och du har dubbelt så mycket (eller nästan) utrymme på skivan. Sedan är det en smaksak om man tycker det är bra eller inte.

Det lär finnas skivspelare med två lampmekanismer, där man kan vända skivan utan ytterligare åthävor,men någon standardvara är det inte. Jag undrar varför.

Sven Wickberg

Har man grammofon med dubbelsidig avläsning behövs inget extra hål. (red).
Jag har hört om och även sett s n y g g a
hålupptagningar. Man gör en noggrann mall, lägger in ett skydd för skivan och använder en hålpuns vilken man snurrar mellan fingrarna tills det blir hål. Lika förfarande på motsatta sidan, men glöm inte att vända mallen rätt. Man skär med en vass kniv hacket i skivan, var noggrann!
Jag har fått låna disketter med hål men inte utan vidare kunnat läsa dem. På FD4 UD måste man sätta in en pappbit på motsatta sidan, det dög inte med tejpen för skrivskydd, det uppfattades tydligen som felinsatt diskett och driven - som startar snurra när man sätter in disketten och stänger luckan - stoppar.
(red).

smartaid® III

* Automatisk uppstart av SMARTAID-3 funktioner och printerrutiner
* Bildskärms"dump", skärmens innehåll skrivs ut på skrivare
* Bildskärmseditorn förbättrad med bl.a. delete- och insert-funktion
* Stoppfunktionen vid programkörning, som möjliggör körning av program-men en bit i taget med eller utan TRACE-funktionen
* Multianpassad till såväl äldre som nyare ABC80 samt även för 80 tecken
* Egen printerrutin gör det möjligt att använda den och ABC80 tillsammans med skrivare utan att behöva ladda in printerrutin i arbetsminnet eller ansluta till en flexskiveenhet.

HELP	skriver ut alla SMARTAID-3's och ABC80's kommandon på skärmen
REN	omnumrerar även delar av programmet.
LIST	listar programmet utan att de första rullar förbi. Med pilarna kan scrollning ske framåt och bakåt. Listning av valda programdelar kan ske till fil t.ex. skrivare.
Ü	I stället för LIST kan du använda en förkortning, bokstaven ü
CHANGE	byter ut en gammal variabel till en ny i hela programmet.
FIND	söker och skriver ut hela de programrader, där sökt begrepp finnes.
OLD	tar fram föregående program efter NEW eller RESET.
PEEK	listar minnesinnehåll i decimal-, hexadecimal- och ASCII-form
LIB	visar ett bibliotek på flexskiva (DIR för 8 tums skivor)
SYS	visar kvarvarande- och program-minnesstorlek, olika pekare, systemtid i tim, min, sek m.m.
RESUME	återupptar programkörningen efter CTRL C eller STOP med bibehållna variabelvärden.

FOR...NEXT FOR..NEXT loop kan användas som direkt kommando (utan program)

Dessutom finns AUTO, DEL VAR och START (se SMARTAID II).

NY DESIGN SMARTAID-3 levereras i en robust låda med svart stålplåt och minimalt djup bakom tangentbordet, mindre än 20 mm.

BESTÄLL DIN SMARTAID-3 IDAG !

För demonstration eller beställning kontakta närmaste återförsäljare eller ring OWOCO

OWOCO AB

Kvarnbergsvägen 25, 141 45 HUDDINGE Tel. 08/774 02 90

LISP 1.5

Torbjörn Alm
Tranholmsvägen 3
178 00 Ekerö
0756/30751

Denna interpreter har utvecklats av Lysator och ställts till ABC-klubbens förfogande tillsammans med några andra program. Den kan endast köras på ABC-80 med checksumma 11273, då den nyttjar flera interna rutiner, som har olika plats i olika versioner av ABC-80. Interpretern innehåller de viktigaste LISP-funktionerna. Den är utformad som ett .ABS program och startas över CMDINT. Den kan endast köras på system med floppy-disk. På ett 32k-system har man när interpretern startar c:a 6000 LISP-celler till sitt förfogande. Då varje LISP-cell är 4 bytes, blir det mindre än 2000 celler på ett 16k system, villket är i minsta laget. På disken finns även 3 hjälp-program, LOAD, SAVE och EDITF. LOAD kan ladda in program från skivan, SAVE kan skriva ut program i återläsbar form, och EDITF är en LISP-editor för att editera LISP-funktioner.

LISP-interpretern använder underline, _, som promt. För att starta och ladda hjälp-funktionerna ges följande kommandon:

```
BYE

<Command interpreter laddas>

LISP

LISP 1.5 (V2.1)

_(OPEN 1 'LOAD)

_(EVAL (READ 1))

_(CLOSE 1)

_(LOAD SAVE EDITF)

_
```

Nu finns c:a 5500 LISP-celler till förfogande. LOAD används som framgår av ovan så, att efter LOAD räknar man upp de filer, som skall in. De förutsätts alla ha extension .LSP .

SAVE används på följande sätt:

```
_(SAVE FILNAMN FUN1 FUN2 FUN3 ... FUNn)
```

Det finns inget krav att alla namn skall in på en rad, högerparentesen avslutar listan. Det finns en liten feature i SAVE. Om man nyttjar långa namn, kan man få för långa rader, dvs. över 120 tecken. Eftersom namn i OBLIST tar 1 LISP-cell per 2 tecken, är det starkt att rekommendera korta namn och att använda samma symboler där det är möjligt.

Eftersom ingen dokumentation fanns tillgänglig, åtog jag mig att försöka utreda vad LISP-interpretatorn kunde göra och framför allt hur editorn skulle användas. Nedanstående beskrivning baserar sig på studier av LISP-koden för EDITF samt ett antal timmar trial-and-error för att utröna vad den igentligen gör.

Detta är vad jag kom fram till, och jag har per telefon fått bekräftat av en av författarna, Anders Isaksson:

För att editera funktionen FUNK skriver man:

```
_(EDITF FUNK)

*
```

EDITF promtar alltid med *

En LISP-funktion är alltid en lista, som i sin tur är uppbyggd av atomer och listor. Var och en av dessa kan editeras, genom att man uppsöker den och låter den bli aktuell nivå. En lista expanderas då den pekas ut och kan editeras. Om listan i sin tur är uppbyggd av listor atomer, kan dessa pekas ut och editeras.

Kommandon:

?	Lista aktuell nivå i sin helhet i kompakt form.
P	Lista aktuell nivå. Atomer skrivs ut, listor markeras med &.
PP	Print Proper, dvs redigerat med indrag, aktuell nivå.
AB	Abortera editeringen och återgå till LISP.
OK	Låt den editerade versionen av funktionen ersätta den gamla.
1-99	Tag fram ATOM/LISTA 1-99 på aktuell nivå, och låt den bli aktuell nivå.
Ü	Gå tillbaka till lägsta nivå.
(n (ABC))	n är 1-99. Byt ut ATOM/LISTA n på aktuell nivå mot (ABC).
Z	Tag fram sista ATOM/LISTA på aktuell nivå och gör den till aktuell nivå.
(N (ABC))	Lägg in (ABC) sist på aktuell nivå.
(: (ABC))	Byt ut allt på aktuell nivå mot (ABC)
0	Gå upp en nivå.
ÜÜ	Gå tillbaka till lägsta nivå.
(n (ABC))	n är 1-99. Byt ut ATOM/LISTA n på aktuell nivå mot (ABC).
Z	Tag fram sista ATOM/LISTA på aktuell nivå och gör den till aktuell nivå.
(N (ABC))	Lägg in (ABC) sist på aktuell nivå.
(: (ABC))	Byt ut allt på aktuell nivå mot (ABC)
0	Gå upp en nivå.
(E (FCN))	Detsamma som (EVAL (FCN)). Kan tex. nyttjas för att mitt i en editering starta en ny, och sedan gå tillbaka till den gamla.
(UP)	Detta är en feature. Det är ett embryo till en funktion, som finns i editorn för Interlisp på DEC-10/20. Den har dock aldrig lagts in. Om man gör UP erhålls ibland utskrifter efteråt med ? och PP, som ser ut som om något hänt, men är en synvilla.

För att lära sig hur LISP-editorn fungerar, är det bästa att lägga in en enkel funktion, och sedan applicera olika editeringskommandon och se vad som blir resultatet.

Jag har kört igenom ett enkelt exempel, som visar hur man gör. Vi börjar med att definiera en enkel funktion:

```
_(DE CADDAR (L) (CADR (CDAR L)))
```

LISP svarar med CADDR.

Därefter skriver vi:

```
_(EDITF CADDAR)
```

EDITF svarar med en *. PP ger utskriften:

```
(EXPR LAMBDA (L)

(CADR (CDAR L)))
```

? ger utskriften:

```
(EXPR LAMBDA (L) (CADR(CDAR L)))
```

P ger utskriften:

```
(EXPR LAMBDA & &)
```

där & markerar att det är en lista på den platsen och ej en atom. Om man nu skriver:

3 P return, erhålls:

```
(L) .
```

Vi har nu pekat ut Atom/lista på pos. 3 som aktuell nivå.

0 4 P return ger utskriften:

(CADR &), och följs det av:

2 P return erhålls:

```
(CDAR L) .
```

Vi är nu inne vid den innersta listan. Ü return gör att vi kommer längst ut igen, och med 0 kan vi gå ut en nivå i taget.

Antag att vi vill byta ut L mot K i hela funktionen, dvs den formella parametern eller LAMBDA argumentet. Vi ger följande kommando-sträng, som kan ges på en rad, varvid EDITF tar in ett kommando i taget och exekverar det.

(3 (K)) 4 2 (2 K) Ü ? return.

EDITF svara med:

```
(EXPR LAMBDA (K)(CADR(CDAR K)))
```

för ABC 80

För varje exekverat kommando erhålls en asterisk på en rad på skärmen. Här följer ytterligare tillämpningar. Om man i detta läge skriver (N KK) ?, erhålls som svar:

(EXPR LAMBDA (K)(CADR(CDAR K))KK)

KK har lagts in sist på aktuell nivå, dvs här högsta nivå.

Vill man nu byta ut KK mot K skriver man: 5 (: K) Ü ? och får ut:

(EXPR LAMBDA (K) (CADR (CDAR K)) K)

Skriver man nu OK sparas den nya versionen av CADDAR, under det att om man skriver AB , avslutas editeringen och eventuell ny definition kastas bort. Allt arbetsutrymme, som nyttjats under editeringen frigörs, och kommer att knytas in i free space vid nästa Garbage Collection (GC).

Tillgängliga funktioner i ABC-80 LISP.

Med ledning av OBLIST har jag sammanställt de funktioner som finns i ABC-80 LISP. Jag har delat upp dem, beroende på typ av funktion.

Aritmetiska och logiska funktioner:

PLUS	MINUS	TIMES	QUOTIENT
ADD1	SUB1	IAND	IOR
IXOR			

Predikat:

NULL	ATOM	LISTP	EQ
ZEROP	NUMBERP	GREATERP	

Input- och Output-funktioner:

READ	PRINT	PRIN1	TERPRI
TEREAD	OPEN	CLOSE	

Elementära listfunktioner:

CAR	CDR	CAAR	CADR
CDAR	CDDR	CONS	LIST
SETQ			

LISP-operatorer för definition av funktioner:

PROG	GO	RETURN	COND	
AND	OR	FUNCTION		
PROGN	QUOTE	LAMBDA	DE	DF

Diverse operatorer:

NIL	T	APVAL	SUBR
FSUBR	EXPR	FEXPR	FUNARG
IT	RPLACA	RPLACD	EVAL
APPLY	ALIST	ASSOC	GETPROP
PUTPROP	PL	REVERSE	MEMB
LAST	PEEK	PEEK2	POKE
POKE2	CALL	INP	OUT
OBLIST			

Definition av funktioner:

Det finns 2 slag av funktioner. Den ena typen är normala funktioner med ett fast antal argument, som evalueras före anropet, och som var och svarar mot en formell parameter i funktionen. Den andra typen är s.k. no-spread funktioner, där argumenten ges i form av en lista med alla argumenten i icke-evaluerad form. De definieras med DE resp. DF.

Exempel: funktionen NCONC, som länkar ihop 2 listor fysiskt, finns ej inbyggd. Den definieras med hjälp av LAST och REPLACD:

(DE NCONC (L1 L2) (COND((NULL L1)L2) (T(RPLACD(LAST L1)L2))))

DE är en funktion, som i sin tur nyttjar RPLACD. När man gör SAVE på funktioner, matas de ut på följande sätt:

```
(PROGN (RPLACD (QUOTE NCONC)
       (QUOTE (EXPR (LAMBDA (L1 L2)
              (COND((NULL L1)
                    L2)
                    (T(RPLACD (LAST L1)
                              L2))))))))
```

För DF-funktioner gäller, att funktionen måste göra EVAL och argumenten skall evalueras. Dessutom har en DF-funktion alltid 2 argument. Det första argumentet är en lista med alla parametrar i icke-evaluerad form och det andra argumentet är ALIST med alla bindningar. Denna parameter måste vara med i definitionen, även om ALIST ej nyttjas. Exempel på DF-funktion:

(DF F (X A)(FB (CAR A) (CADR X)))

Jag testat ett flertal funktioner, dock inte alls alla. Det mesta tycks fungera. Jag har skrivit om (dvs. bantat alla namn) i det genealogi-program, som fanns i BYTE September 1981. Det består av sammanlagt 52 funktioner och kräver c:a 3000 LISP-celler. Genom att ersätta en APPEND med den ovan definierade NCONC, gick det att hantera litet större datamängder utan att spränga stackeh. Programmet fungerar hjälpligt, men klarar givetvis endast en begränsad databas. Jag har lagt upp en fil, BAGGINS. LSP, som är tagen från artikeln i BYTE, och som beskriver FRODO BAGGERs släktträd ur sagan om ringen. Programmet BAGLOD.LSP kan laddas och startas sedan via (BAGLOD). Det läser in och laddar databasen. Sedan kan man ställa frågor via (R namn1 namn2) och får till svar släktskapen. Släktprogrammet heter GENEALOG.LSP .

Som ett exempel på en litet mer komplicerad LISP rutin har jag valt just BAGLOD, som ser ut så här:

```
(DE BAGLOD NIL (PROG (B) (OPEN 1
('BAGGINS))

LOOP (SETQ B (READ 1))

(COND((ATOM B)(PROGN(PRINT B)(TERPRI)
(RETURN('DONE)))))

(PRINT B)(PRINT SUN)(EVAL B)(GO LOOP)))
```

Programmet startas via (BAGLOD) . PROG är en speciell LISP-funktionstyp, som medger hopp till en LABEL, och därigenom mer konvensionell programstruktur. LOOP är en sådan LABEL. (B) avser en lokal variabel. Först öppnas filen BAGGINS.LSP för läsning på enhet 1. Loopen börjar med LOOP. Först läses en atom eller lista in. Den testas, och om det är en atom, returneras DONE. I annat fall trycks funktionen ut på skärmen tillsammans med tidigare släktträd med funktionen SUN. B evalueras, och återhopp sker till LOOP. Man exekverar de fakto en kommandofil. Detta exempel visar på det faktum att i LISP har data och program samma format, vilket ju är unikt.

Filen BAGGINS.LSP börjar med (CUNI), som initierar databasen. Funktionen MA(rriage) avser att koppla ihop 2 personer, funktionen C(hild) definierar barn, och (F NAMN SEX) definierar en person. Samtliga dessa personer är FEXPR eller no-spread funktioner, dvs. argumenten evalueras ej innan en funktion anropas. Detta ger funktionerna möjlighet att hantera samtliga argument i original-format.

Varje gång Garbage Collection genomlöpts, erhålls en utskrift på skärmen med information om kvarvarande utrymme.

Allmänna synpunkter på ABC-80 LISP.

Som framgår av artikeln, är ABC-80 en alltför liten maskin för LISP. På grund av det sätt på vilket LISP bygger upp mellanresultat och arbetar rekursivt till i princip godtyckligt djup, är språket oerhört minneskrävande. Om med programmet GENEALOG söker en relation mellan personer i direkt nedstigande led eller mellan syskon, gör GENEALOG rätt, men så fort det fråga om mer avlägsna relationer, erhålls några GC-utskrifter, följt av STACK OVERFLOW. Jag har även sökt stoppa in ett program för symbolisk derivering, en klassisk LISP-tillämpning, i ABC-80 LISP, men det är helt omöjligt. Över huvud taget får man betrakta ABC-80 LISP som en ren leksak. Den har ett värde i tex. skolundervisningen för att man skall kunna se hur språket fungerar och köra några enkla exempel, men vill man verkligen köra LISP i något produktivt sammanhang, måste man ha tillgång till en DEC 20 eller liknande.
En annan kraftig begränsning är att alla beräkningar görs med 16 bitars precision. Skulle man t.ex. vilja laborera med rationella polynom, en typisk LISP-tillämpning, slår man i taket överallt på en gång, både i fråga om talområdet och listutrymmet.

Underline-Markör

För dig som är händig i elektronik torde detta vara en välkommen nyhet. Du bör vara insatt i hur IC-kretsarna på ABC80:s kretskort är placerade. Koordinatsystemet som finns utmärkt på datorkortet använder jag mig av i kretsschemat över underline-markören. Om du inte känner till detta samt ej heller klarar av att läsa kretsschemat bör du överlåta modifieringen till underline-markören åt en fackman.

Underline-markören för ABC80 får samma prestanda som den i ABC800.

Markörfakta:
Blinkande "underliggande", fast vid skrivning, eller efter ca 5 sek om inte markören flyttas.

Med de tre omkopplarna kan följande funktion på markören lätt ställas in. Vid körning i CP/M kan ibland markör med helt fast sken vara lämplig.

En omkopplare ger antingen "underline-markör" eller "fyrkantig markör". De övriga två omkastarnas funktioner här nedan, gäller för de båda markörtyperna.

En omkopplare ger antingen ständigt blinkande markör eller "blink i 5 sekunder". Den tredje omkopplaren har enbart en uppgift, nämligen att ge möjlighet till att få ett fast sken på markören.

Uppkoppling:

Du kopplar enklast upp de ingående komponenterna på ett förborrat fenolpapperkort med modul-delning. Placera kortet längst fram i tangentbordet, till höger. Till höger på ABC80:s kretskort finns ett hål, detta kan användas till att skruva fast det lilla kretskortet i.

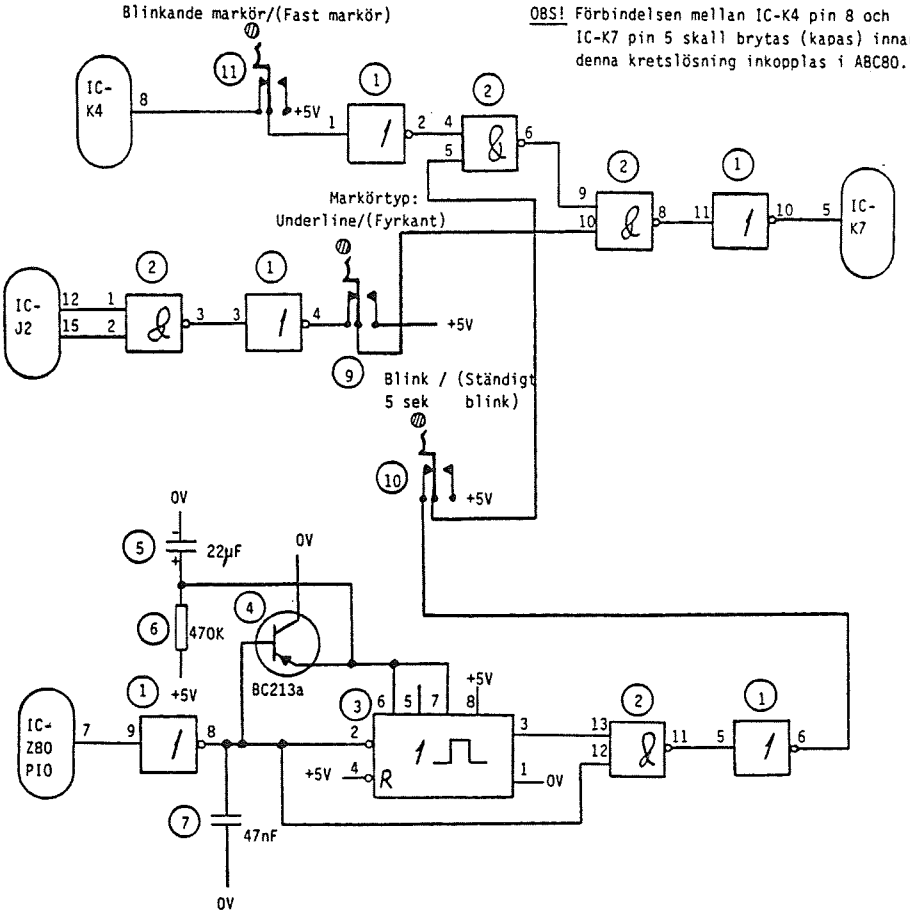
Ledning:

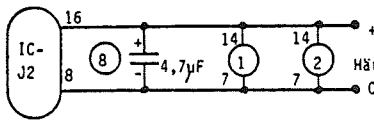
De ovala symbolerna i kretsschemat visar externa anslutningar, dvs. trådar som skall anslutas till ABC80:s stora kretskort. I de ovala symbolerna står koordinaten till aktuell IC-krets samt bennummer på själva anslutningen. Ett undantag från detta är den ovala symbolen med följande text "IC-Z80 PIO".
MED "IC-Z80 PIO" avses istället den krets som kallas så (Z80 PIO), då denna är så stor att några koordinater ej torde ge ett entydigt besked om vilken krets man avser.

Lycka till med arbetet.
Kretsschemat får publiceras i ABC-bladet men i övrigt inte spridas, och ej heller får kretslösningen användas i kommersiellt syfte.

Hälsningar
Nils-Gunnar Westermark
08/45 85 48 eller
MBS-N 0047 73243
Stockholm

KRETSSCHEMA över UNDERLINE-MARKÖR TILL ABC80
av samma typ som i ABC800.
(c) N-G Westermark, Stockholm 1982-07-09.





Härifrån hämtas matningsspänning till i schemat ingående komp.

KOMPONENTFÖRTECKNING:

Pos.	Komponent,	Pos.	Komponent,
1	Mikrokrets 74LS04	6	Resistor 470 KOhm
2	Mikrokrets 74LS00	7	Kondensator 47 nF
3	Mikrokrets NE555	8	Kondensator 4,7 µF
4	Transistor BC213a	9-11	Omkopplare 1-pol växl. låsande
5	Kondensator 22 µF		

Kassett fyra

CASDISK .BAS
Det program som traditionsenligt inleder en ABC-Kassett.

GLIPP .XXX
Spelprogram av hög kvalitet. Består av två filer som trots ovanlig extension är BAC-filer. Programmet kan köras på kassett.

GLIPP .STY
Andra-programmet i GLIPP-paketet, DU MÅSTE starta med RUN GLIPP.XXX ty där laddas maskinkoden före körning av GLIPP.STY.
Viktigt ! Om DU flyttar GLIPP till annan kassett är det nödvändigt att spara dem i samma ordning, samt med motsvarande namn och extension. Flyttning av GLIPP till annan kassett:
Load cas:
FOUND GLIPP .XXX (maskinens svar)

byt kassett tryck ned play och rec
Save GLIPP .XXX
När bandspelaren stannat byt till ABC-kassett igen
Load cas:
FOUND GLIPP .STY (maskinens svar)

byt till din GLIPP-KASSETT tryck ned play och rec
Save GLIPP .STY
(OBS ! DU kan inte köra glipp på kassett om DU har floppyn inkopplad. Stäng av floppyn.)

GAPHALS1.BAS
Simulering av driften i ett stort kärnkraftverk. Det gäller att hålla igång utan för stora utsläpp eller en härdsmälta. Du har ett antal kontrollstavar till ditt förfogande.

HJÄLPARE.REM
Textfil som beskriver nedanstående program. Filen kan läsas med TV-editorn. Har man floppy kan man använda programmet VISA från kassett nr 1.

HJÄLPARE.16K
Detta program som ligger i BAS-format trots extension 16K sparas lämpligen med extension BAS på din diskett eller kassett. Någon risk för sammanblandning med nästa program föreligger knappast då det räcker med ett av programmen för din del, antingen har man 16K eller 32K.

HJÄLPARE.32K
Samma program som ovan men för 32K minne.

HJÄLP16K.ASM Skall döpas om till DELETE2.ASM
(Heter KÄLLKOD1.ASM på monitorn.)
Källkod till programmet DELETE2.ASM av misstag med i stället för källkod till maskinspråksdelen i HJÄLPARE.16K.
Kan assembleras med assembleraren vilken finns på kassett nr 6 På nya kassetter vilka görs efter 1982 11 10 kommer detta att rättas till.

HJÄLP32K.ASM Skall döpas om till RESETFIX.ASM
(Heter KÄLLKOD2.ASM på monitorn.)
Källkod till programmet DELETE2.ASM + RESETFIX.BAS av misstag med i stället för källkod till maskinspråksdelen i HJÄLPARE.32K.
Kan assembleras med assembleraren vilken finns på kassett nr 6 På nya kassetter vilka görs efter 1982 11 10 kommer detta att rättas till.

När Du döper om ett program på disk skriver Du:
NAME"HJÄLP16K.ASM"AS"DELETE2.ASM"
och
NAME"HJÄLP32K.ASM"AS"RESETFIX.ASM"
Har Du kassett måste Du använda TV-editorn, ladda in och och sen spara under nytt namn på samma ställe. Beträffande HJÄLP32K bör du ta bort de första 1580 tecknen vilka tillhör DELETE2.

RESETFIX.BAS
Med RESETFIX kan du klara av situationer när datorn 'låst' sig. Du gör reset och kan ändå få tillbaka ditt program.
Källkoden till RESETFIX finns av misstag med efter 1580 tecken i HJÄLP32K.ASM. Se ovan.

INUTFIL .BAS
Program som läser och skriver textfiler. En kombination av VISA och MAKETEXT. Vid inskrivning av text, görs avslut med * (stjärna eller gångertecknet).

TV .BAS
Inledningsprogram till TV-editorn som här kommer i två nya versioner. På denna kassett finns ingen kassett-version av TV-editorn, även om TVMAIN3 kan utnyttjas i kassett-versionen, som finns på kassett nr 3. Programmet TV laddar in TVMAIN.BAC, och läser dessutom in TVSUBR.ABS som innehåller en del maskinspråksrutiner.

TVMAIN3 .BAC

TVSUBR .ABS

TVLIB .MRG
En rutin som gör det möjligt att få veta hur mycket utrymme som finns kvar på floppyn under pågående editering utan att man därför måste gå ur TV-editorn. Du använder TVLIB för att skapa en speciell version av TVMAIN. Tänk på att denna nya vesion av TVMAIN inte har lika mycket minne tillgängligt för editering som den ursprungliga versionen. Så här går det till att skapa en ny version av TVMAIN med lib-funktion.

LOAD TVMAIN3

MERGE TVLIB.MRG

SAVE TVMAIN

Om du vill ha två versioner av TV-editorn på samma systemskiva så kan du göra så här.
Bestäm först vad du vill att TV-editorn ska ladda in för version i första hand. Döp den versionen till TVMAIN och döp den andra till TV2 exempelvis. När du nu ger kommandot 'RUN TV' så laddas utgångsversionen in, låt oss säga versionen med lib-funktion. Vill du byta version för att få mer minne så ger du kommandot ;h>> och går ur editorn. Nu kan du ge kommandot RUN TV2 . Detta fungerar eftersom maskinspråket ligger kvar.

GRAFPRIN.PRT
Program för utskrift på printer av skärmens innehåll. För att fungera krävs det att du har en printer-rutin i maskinen exempelvis ABCV24 som finns på kassett nr 3.

FILDIS .REM
Textfil som beskriver nedanstående program.

FILDIS .BAS
Disassembler för ABS-filer som utnyttjar filerna OPCODE1.TXT och OPCODE2.TXT vilka båda finns på kassett nr 3. Med FILDIS kan du lära dig mer om ABC80. Vi anser oss dessutom veta att ABS-filer kommer att bli allt mer intressanta för den vanlige amatören. Du kan räkna med intressant läsning i tidningen och spännande program på kommande kassetter.
FILDIS kan endast köras på floppy och de två TXT-filerna som nämts ovan måste också ligga på disken.

ABCTRANS.80K
En utveckling av ABCTRANS som fanns redan på kassett nr 1. Programmet som ska köras tillsammans med ABCV24, vilket finns på kassett nr 3, är ett kommunikationsprogram. Med ABCTRANS kan du föra över filer från monitorn till ABC80, men du måste ha floppy och modem. Du kan naturligtvis skicka filer i andra riktningen. Programmet fungerar lika bra på 40- som 80-kolumners skärm.

DOSCREEN.BAS
Program som används för att tillverka en skiva med FORTH-SCREEN's. Tyvärr finns en bug i DOSCREEN. Det är ingen helt katastrofal bug och det är inte säkert att du märkt den. Du rättar buggen lättast med filen DOSCREEN.MRG från kassett nr 6.

SCREEN .TXT
En textfil som kan omvandlas till FORTH-SCREEN's med DOSCREEN.

FORTH16 .ABS
Något så egendomligt som ett maskinnära högnivåspråk. Det är hypersnabbt, bara obetydligt långsammare än maskinspråk. Att få FORTH i sin ABC80 är ungefär som att få en helt ny dator! Om du trodde att du kunde din ABC80 vid det här laget så är det alltså dags att tänka om. Har du ställt din ABC80 på hyllan, ta ner den!
FORTH16 fungerar lika bra på en 32K maskin och du kan titta på FORTH även om du saknar floppy, vad du behöver är ett CMDINT-program för kassett. Ett sådant program ligger på kassett nr 5, dessutom är det bra att kunna flytta FORTH16.ABS till en annan kassett. Det kan du göra med COPYCAS på kassett nr 6. Slutligen, - FORTH har ingenting med FORTRAN att göra, tack och lov, det är en alldeles egen produkt som står stadigt på egna ben. Läs om FORTH i ABC-rapport nr 2, Eller FIG-FORTH ON ABC80 som den heter.

FVISA .BAS
Program som endast fungerar på floppy. Det visar FORTH-SCREEN's på skärmen. Du har lämpligen en skiva med SCREEN.TXT och FVISA på i driven när du läser FIG-FORTH ON ABC80 och kan då följa med genom att trycka fram olika SCREEN's. FIG-FORTH ON ABC80 kan köpas från klubben och kostar 60 SEK.

Kassett fem

CASDISK .BAS 12
Program som kopierar från kassett till disk. Ligger normalt först på en ABC-Kassett.

FYLLE .BAS 15
Musik och bild med överraskning. Fungerar utan floppy och med 16K minne.

TVSPEL .BAS 29
Liknar de TV-spel som kan köpas i varuhusen. Fungerar på grundutrustning.

DEFENDER.BAS 19
Ett bra spelprogram som inte får dig att bryta samman.

HUSET .BAS 6
Bild 'ritad' med GRAFEDIT som ligger på kassett nr 6.

TONTEXT .BAS 20
Informationsprogram till programmet TONE, som är ett musikprogram för musikproduktion. TONTEXT körs som ett BASIC-program och gör att du bekvämt kan läsa texten, byta sida osv. Det är naturligtvis möjligt att skriva ut texten på printer och läsa den sedan, men det är inte avsikten.

TONE .BAS 10
Program för den slutliga musikproduktionen som sker i två steg.

ROCK .BAS 9
Rock'n roll kan ABC80 producera med hjälp av TONE och ROCK. ROCK används som indata i TONE och musiken spelas in på en kassett som sedan kan spelas upp i ABC80's högtalare eller i vilken stereoanläggning som helst.

TON1 .BAC 4
En kortversion av ROCK dvs musiken är densamma, men programmet är kortare.

CRTOS .INF 15
Textfil med information om programmet CRTOS.BAS. Textfilen komplett nedan.

CRTOS .BAS 11

CRTOS VI.3

Råkar du också ut för det där förargliga plingande ljudet när du trycker på RETURN efter att ha matat in en lång programrad och finner att du är tvungen att mata in hela raden på nytt? En programrad med syntaxfel sparas inte i minnet (utom i bildminnet). Därför hjälper inte ens ED-kommandot
CRTOS-bildskärmseditor kompletterar systemet genom möjlighet ges att ändra i rad på bildskärmen. I editorn styr man cursorn med kontrolltecken. När man trycker på RETURN, sänds hela den rad på bildskärmen där cursorn står. För att få snabbhet, enkel hantering och litet minnesutrymme, har programmet skrivits i maskinkod av:
M HELENIUS & E A REIMAN.

Uppstart av editorn

Programmet laddas från skiva eller kassett. Efter RUN-kommandot kan man med NEW-kommandot rensa ut pokesatserna ur arbetsminnet. Editorn utnyttjar nu endast 429 bytes (kommando-tolken CMDINT 6 på adresserna 62291..62498 samt radbehandlingstolken CRTED på adresserns 62499...62719)
OBS! Om du har flexskiveenhet inkopplad, måste du innan RUN-kommandot ge:

POKE 65064,250

Efter det man gått ur bildskärmseditorn, kan man åter komma in i editorn med kommandot:

;CALL(-3)

Arbetsminnets innehåll förblir oförändrat.

Tangent-ljudsignal

När du är i editorn hörs ett svagt knäpp i högtalaren varje gång du trycker ned en tangent. Detta är en hjälp för den som är van att skriva utan att titta på bildskärmen. Med knäpp-ljudet är det lättare att uppfatta t.ex. nedtryck av två tangenter samtidigt. Ljudsignalen kan stängas av med kommandot:

POKE 32767,0

Genom att ändra minnescellens värde till 125 får man tillbaka ljudsignalen.

Styrning av cursorn

Cursorn kan styras med följande kommandon:

CTRL/W - uppåt
CTRL/A - åt vänster
CTRL/S - åt höger
CTRL/Z - nedåt

Tangenterna är valda på grund av sitt logiska läge.

Övriga kommandon i editorn är:

CTRL/H - vänsterpil
- borttagande av tecken
CTRL/I - högerpil
- tillägg av tecken
CTRL/X - tar bort resten av rad

CTRL/H tar bort tecknet där cursorn står samt lägger till ett blanktecken i slutet av raden. Tecken till höger om cursorn flyttas ett steg till vänster.

CTRL/I lägger till ett blanktecken där cursorn står och flyttar tecknen till höger om cursorn ett steg till höger. Om raden blir för lång försvinner tecknet längst till höger.

CTRL/X tar bort slutet av raden från cursorn och ersätter det med blanktecken Cursorns läge förändras ej.

Grafisk mod

Alla styrtecken utom ovan nämnda samt RETURN CTRL/M kan skrivas i programmet T.ex. i en sträng i en printsats ger CTRL/J radförflyttning. Med CTRL/Q får man grafisk mod (motsvarar CHR$(151)) och med CTRL/B kan man återvända till textmoden.

ANMÄRKNING:
Kom ihåg att ändring på skärmen blir giltig först sedan du tryckt på RETURN. Ändringen kan göras ogiltig genom att innan RETURN flyttar cursorn till annan rad, t.ex. med CTRL/Z. Om du skriver basic-program med editorn, kan dina programrader bli max. 40 tecken långa.

Programmet hämtat ur den finska ABC-tidningen nr 2. Med tillstånd översatt av Risto Koivula.

NOCTRLC .TXT 6
Sparsam information och källkod till NOCTRLC .BAS

NOCTRLC .BAS 5
Detta program flyttar CTRL-C till annan knapp som du själv väljer. Lista programmet och fundera !

CMDINT .CAS 5
Kassettversion av CMDINT.SYS vilken gör det möligt att ladda in ABS-program från kassett. Programmet döps lämpligen om till CMDINT.BAS.

BIGTEXT .BAS 6
Programpaket för stortext på printer uppbyggd med varierande matrisstorlekar.

BIGTXT12.BAS 50
Matrisstorlek 12 x 12 tecken.

BIGTXT69.BAS 17
Matrisstorlek 6 x 9 tecken.

BIGTXT57.BAS 16
Matrisstorlek 5 x 7 tecken.

SD10 .BAS 38
Detta program refererat i ABC-bladet nr 1. 1982 sid 28 i texten Diska Rent.

CASMON .BAS 13
Kassettmonitor som kan ta emot filer som sänds med CASSEND eller ABCTRANS.

SAFT .BAS 22
Simple Ascii File Transmission - som det utläses är ett kommunikationsprotokoll enligt vilket ordbehandlingssystem som skall upphandlas av staten måste kunna kommunicera. Detta innebär att det kommer att vara ett stort intresse för SAFT bland representanter för ordbehandlingsutrustning. SAFT-program kommer att finnas på alla utrustningar som vill ha en chans på den svenska marknaden. SAFT är långsammare än ABCTRANS men ger säkrare överföring.

FILTRANS.REM 7
Textfilen Anvisningar till programmet FILTRANS.BAS komplett nedan.

FILTRANS.BAS 27
Anvisningar till programmet FILTRANS

Programmet FILTRANS är fullt kompatibelt med ABCTRANS och kan alltså ersätta det senare programmet vid körning exempelvis mot ABC-klubbens Monitor.

Samtidigt innehåller FILTRANS en Mini-Monitor som tillåter fil-överföring mellan två ABC-80. Den ena ABC-80 kallar vi A-datorn och den andra ABC-80 B-datorn.

Om B-datorn laddat in T80PRT (eller ABCV24) och sedan FILTRANS och A-datorn T80PRT och sedan ABCTRANS kan A-datorn göra så att B-datorn går över i Monitormode genom att ge kommandot: <CTRL-B>MONITOR<RETURN>

Sedan kan man från A-datorn ge kommandona GETFIL och SENDFIL och på detta sätt föra över filer mellan A- och B-datorn på samma sätt som vid körning mot Klubbmonitorn. Man avslutar från A-datorn med kommandot BYE. Därvid återgår B-stationen till halv duplex. I denna mode kan man skriva korta meddelanden till varandra.

Om även A-datorn har FILTRANS kan man från B-datorn ge kommandot <CTRL-B>MONITOR<CR> och få A-datorn över i Monitor-mode.

FILTRANS är alltså helt symmetriskt i användningen.

Gunnar Tidner 1982-02-18

SCREEN2 .TXT 59
Fortsättning på FORTH SCREENS med numren 34 till 69 samt ändring av 7.

FORTH32 .ABS 32
Programmeringsspråket FORTH för 32K minne i extra kompakt version, som du märker om du jämför med 16K-versionen på kassett nr 4. Skillnaden är faktiskt så tor som 18 sektorer, hur detta kommer ig får du läsa i tidningen.

Kassett sex

KASSETT SEX

CASDISK .BAS 12
Det vanliga kopieringsprogrammet.

PAUSBIRD.BAS 27
Program med grafik och ljud av Ferdinand Mican, se artikel i tidningen, nr 2. 1982.

MORSEÖVN.BAS 40
Program för den som vill lära telegrafi. Programmet utnyttjar ljudgeneratorn och det fordras alltså ingen extra hårdvara.

GRAFEDIT.HLP 18
Textfil som ger information om nedanstående program. Filen läses dessutom in av programmet GRAFEDIT.
Texten GRAFEDIT.HLP finns helt utskriven på annan plats i detta ABC-blad
(red)

GRAFEDIT.BAS 43
Grafedit sparar bilderna i två olika format - .BAS eller som .PIC. Vid .BAS skapas ett basicprogram där bilden är direkt visbar - se HUSET.BAS eller ABC80.BAS. Vid .PIC lagras bilden som textfil där det endast blir en meningsfull bild med GRAFEDIT.

ABC80 .BAS 7
Bild som producerats med hjälp av GRAFEDIT.

EMBLEM .PIC 4
Bild som kunde ha producerats med GRAFEDIT.

EMBLEM har gjorts av och för ett tekniskt gymnasium. Programmet har sedan anpassats till GRAFEDIT och lagrats i det speciella formatet PIC. Filer i PIC-format kan ej köras direkt som program utan måste läsas in i GRAFEDIT.

CASCOPY .REM 7
Beskrivning av programmet CASCOPY som vi tror kommer att revolutionera tillvaron för alla tappra Kassett-Fantomer. Nu kan ni flytta filer från ABC-Kassetten till era egna kassetter på ett enkelt ock smärtfritt sätt. Programmet har den fördelen att det fungerar på alla Check-summor.

CASCOPY .BAS 8
Du startar CASCOPY med RUN CASCOPY. Detta program laddar sen in nästa fil dvs CASCOPY1.

CASCOPY1.BAC 3
Detta program måste ligga i BAC-format på kassetten om det ska fungera.

ASM .BAS 5
Första fil i disk-versionen av assembler-programmet. Filen ASM.BAS laddar in ASM2.

ASM2 .BAS 45
Huvudprogram i disk-versionen läser ASMCON från disk.

ASMCAS .BAS 51
Kassett-version av assemblerprogrammet består av två filer denna och ASMCON. Båda filerna måste ligga på kassett efter varandra med ASMCON sist.

ASMCON . 8
Textfil som innehåller operationskoder och mnemokoder.

LISP .ABS 27
Programmeringsspråket LISP för Check-summa 11273. Se för övrigt artikel i ABC-Bladet.

LISPB .ABS 27
Programmeringsspråket LISP för övriga Check-summor.

SAVE .LSP 10
Hjälpprogram till LISP, LISP-program är textfiler som kan läsas med TV-editorn.

LOAD .LSP 5
Inladdningsprogram från skiva för LISP.

EDITF .LSP 14
Editeringsprogram för LISP som arbetar under LISP-interpretatorn.

FAC .LSP 4

GENEALOG.LSP 65
Se LISP-artikeln i detta ABC-blad (3. 1982)

BAGLOD .LSP 5
Se LISP-artikeln i detta ABC-blad (3. 1982)

BAGGINS .LSP 6
Se LISP-artikeln i detta ABC-blad (3. 1982)

DOSCREEN.MRG 3
Rättelse av DOSCREEN på kassett nr 4. Gör så här!

LOAD DOSREEN (ladda in DOSCREEN.BAS)

MERGE DOSCREEN.MRG

LIST DOSCREEN (spara DOSCREEN som BAS-fil)

```
      Snabbinstruktioner, GRAFEDIT

Kommandon:  L=Line
            F=Frame (Ramritning)
            B=Box   (Fylld rektangel)
            P=Paint (Fyll yta)
            H=Home
            J=Jump  (Till förra punkten);
            M=Mark  (Flytta förra pkt.)
      Ctrl-R=Radera skärmen
     Ctrl-<>=Save
      Ctrl-´=Load

Cursor:     W=Cursor upp
        A & S=Cursor vä & hö
            Z=Cursor ner
    <SPACE>=Ritmod
   <RETURN>=Flyttmod
            I=Inverse
           ->=Set
           <-=Clear
            ?=Instruktioner
```

LISP-Litteratur:

1. **LISP**
   Patrick Henry Winston och Berthold Klaus Paul Horn,
   Addison - Wesley Publishing CO.
   ISBN 0-201-08329-9.
   Prix ca: 200:-
   Rekommenderas av The Mad Programmer. !!

2. **The Little LISPer**
   Daniel P. Friedman,
   Indiana University, Bloomington, Indiana, US.
   ISBN 0-574-19165-8.
   PRIX ??

ABC80 Assembler

Inskrivet av Kalle Lindström

Källa:ABC 80 ASSEMBLER MANUAL
från SCANDIA METRIC AB
SOLNA

ALLMÄN BESKRIVNING

Ett assemblerprogram är ett program, som skrivs i form av förkortade operationskoder - op-koder - där varje op-kod motsvarar en maskininstruktion. Resultatet blir ett källprogram, som sedan skall översättas till lämplig form av ett hjälpprogram - assemblator. Resultatet av översättningen (assembleringen) fås på en fil i form av en eller flera POKE-satser och kallas objektprogram. Ofta kallas såväl källprogram som objektprogram för assemblerprogram, eventuellt med tillägget källkod eller objektkod).

Precis som i BASIC består ett assemblerprogram av ett antal rader med text. Vid assemblerprogrammering tillåts bara en instruktion, d v s bara en sats, per rad. Denna sats kan antingen vara ett direktiv till assemblatorn (en sk pseudoinstruktion) eller en maskininstruktion. Består satsen av en pseudoinstruktion, översätts den inte till maskinkod utan fungerar bara som en order till översättningsprogrammet - assemblatorn. Består satsen av en maskininstruktion, måste den skrivas i en bestämd form med minst ett och upp till fyra ingående element. Dessa element är: Läge, operation, operander samt kommentar. Den aktiva delen i satsen - instruktionsdelen - består av en förkortad operationskod, eller op-kod, samt en eller flera operander. Op-koden är en förkortning av det fullständiga operationsnamnet och beskriver kortast möjliga sätt vad instruktionen innebär. En instruktion kan till exempel se ut som:
LD A,B
Op-koden LD står för Load (ladda) och A respektive B är operander. Instruktionen lyder:
Ladda register A med innehållet i register B.
Vid assembleringen kommer instruktionen att översättas till maskinkod 78H (H står för hexadecimal kod).

Vid assemblerprogrammering av ABC80 tillåter Z80-assemblatorn 74 olika operationer och 25 olika operandtyper, som kan kombineras till 701 olika maskininstruktioner. En fullständig beskrivning av alla dessa instruktioner finner man i "Z80-Assembly Language Programming Manual".

HUR ETT PROGRAM ANVÄNDS - - START, INDATA, UTDATA -

Ett assemblerprogram används vanligen som en subrutin till ett BASIC-program och anropas från BASIC-programmet med instruktionen CALL(U%) där U% är assemblerprogrammets startadress. Är assemblerprogrammet ett självständigt program startas det även i detta fall med instruktionen
CALL(U%).
I båda fallen måste CALL användas som en funktion, dvs CALL medför att en variabel tilldelas ett värde (hämtat från Z80-processorns HL-register) vid återhoppet från assembler-programmet.

Varje program, och alltså även assemblerprogram, behöver i regel någon form av indata att arbeta med. Det finns i princip två möjligheter för ett assemblerprogram att få indata. Den ena metoden är att låta anropsinstruktionen föra med sig ett värde, som då hamnar i Z80-processorns DE-register, genom att skriva anropsinstruktionen som CALL(U%,U1%). I denna instruktion är U% startadressen och U1% är det värde som läggs i DE-registret. Den andra metoden är att assemblerprogrammet hämtar indata från specificerad minnesadress. I detta fall måste önskad indata ha lagrats i respektive minnesadress i ett tidigare skede - antingen av ett BASIC-program, som med instruktionen POKE har lagt ut data till önskad minnesadress, eller manuellt genom att POKE använts som ett kommando för att lägga ut data till önskad minnesadress.

I regel skall programmet också producera någon form av utdata, Analogt med ovanstående kan även detta ske på i princip två sätt. Den ena möjligheten är genom CALL-instruktionen som, i och med att den arbetar som en funktion, för med sig ett värde (hämtat från HL-registret) tillbaka. Den andra möjligheten är att låta assemblerprogrammet lägga ut data till önskad minnesadress, där dessa utdata sedan kan hämtas med PEEK-instruktionen. Då PEEK arbetar som en funktion kan hämtning ske såväl av ett BASIC-program som rent manuellt.

Det är viktigt att ha ovanstående resonemang klart för sig redan vid konstruktionen av assemblerprogrammet, då det i regel gäller att få till stånd ett fungerande samarbete mellan ett BASIC-program och ett assemblerprogram. För en närmare beskrivning av PEEK, POKE och CALL hänvisas till ABC80's bruksanvisning, sid 44-45 samt sid 58-60.

HUR ETT PROGRAM ÄR UPPBYGGT

Ett assemblerprogram består av en följd av satser, som tillsammans formar ett program. För att programmet skall kunna assembleras (översättas) måste vissa direktiv till assemblatorn finnas med. Dessutom är det fördelaktigt att förse källprogrammet med inledande kommentarer, namn på programmet, etc för att man även i framtiden skall kunna veta vad det är för program och vad det är för speciellt med programmet. Vanligen är det ju så att man sparar programmet på en fil, som man kanske inte tittar på så ofta. Att göra om samma program en gång till efter ett halvår bara för att man slarvat med dokumentationen är inte roligt. Efter de inledande kommentarerna följer ett eller flera direktiv till assemblatorn. Ett sådant direktiv är absolut nödvändigt - nämligen information om var programmet skall placeras i minnet. Om man vill att programmets startadress skall vara 65408 ser instruktionen ut så här: ORG 65408. Man måste naturligtvis se till att startadressen refererar till en tillåten ledig plats i minnet (se ABC80's bruksanvisning, sid 50). Därefter följer den aktiva delen av programmet. Givetvis kan pseudoinstruktioner (direktiv till assemblatorn) ingå även i denna del. Den aktiva delen av programmet avslutas lämpligen med op-koden RET, som medför ett returhopp till det anropande BASIC-programmet. (Används assemblerprogrammet självständigt, medför RET att körningen avslutas). Därefter avslutas assemblerprogrammet med pseudoinstruktionen END som talar om att programmet är slut. Assemblatorn ignorerar allt som står efter END.

RADFORMAT

Varje rad i ett assemblerprogram måste skrivas i en bestämd form med upp till fyra ingående element (eller fält). Dessa element är: Läge, op-kod, operand eller operander samt kommentar.

Exempel på en fullständig rad:

Läge	Op-kod	Operander	Kommentar
HIT:	SBC	HL,BC	;Beräkna resten

Läget (HIT) används för att man skall kunna referera till raden. Lägesfältet kan antingen användas för ett rent läge, dvs en hoppadress som kan refereras till, eller för en symbol som då står för ett bestämt numeriskt värde. Även symbolen kan naturligtvis refereras till. Ett läge eller en symbol måste alltid följas av kolon om det inte står längst till vänster på raden. (Se även "Symboler och Lägen" för ytterligare information).

Nästa fält är avsett för operationskoden. Op-koden får INTE skrivas längst ut till vänster på raden utan måste då föregås av minst ett mellanslag. Inleds raden med ett läge måste även i detta fall op-koden avskiljas med minst ett mellanslag.

Därefter följer operandfältet som även detta måste avskiljas från föregående fält med minst ett mellanslag. Operanderna avskiljs sinsemellan med ett kommatecken.

Det sista fältet på raden är kommentarsfältet som alltid måste inledas av ett semikolon. Kommentarsfältet får skrivas var som helst på raden - dock alltid som sista fält. Kommentaren är enbart till för att människan/programmeraren lättare skall förstå vad programmet gör och ignoreras helt av assemblatorn.

Alla dessa delar behöver dock inte finnas med på varje rad i programmet. Kommentarer och lägen kan alltid utelämnas. Dessutom finns det vissa operationskoder som inte har några operander.

Det är även tillåtet att skriva rader som bara innehåller en kommentar och/eller ett läge. Rader med enbart kommentarer kan många gånger vara praktiska för att beskriva ett helt program, eller avsnitt av program.

PSEUDOINSTRUKTIONER

Som tidigare nämnts består ett assemblerprogram inte bara av maskininstruktioner. I programmet ingår även nödvändiga direktiv t''' assemblatorn, sk pseudoinstruktioner. Dessa pseudoinstruktioner översätts inte till maskinkod utan används bara för att styra översättningsprogrammets arbete.

I ABC80 assemblatorn ingår följande pseudoinstruktioner:

ORG nn — Laddar assemblatorns adressräknare (Location counter) med värdet nn. ORG används för att ange programmets startadress. (Observera att startadressen måste vara tillåten - se ABC80's bruksanvisning, sid 50).

EQU nn — Ger symbolen som står i lägesfältet värdet nn. EQU används för att definiera konstanter i programmet.

END — Markerar slutet på programmet. Assemblatorn ignorerar allt som står efter END.

DEFB n — Reserverar en byte i minnet och laddar den med värdet n. Den reserverade byten får den adress som just då anges av adressräknaren.

DEFB 's' — Reserverar en byte i minnet och laddar den med ASCII-representationen av tecknet 's'. (Specialfall av ovanstående). Den reserverade byten får den adress som just då anges av adressräknaren.

DEFW nn — Reserverar två bytes i minnet och laddar dessa med värdet nn. Observera att den minst signifikanta byten lagras först i minnet. Den första reserverade byten får den adress som just då anges av adressräknaren och den andra reserverade byten samma adress + 1.

DEFS nn — Reserverar nn bytes i minnet utan att ge något värde. Den första reserverade byten får den adress som just då anges av adressräknaren, den andra reserverade byten får nästföljande adress, osv.

DEFM 'ss' — Reserverar lika många tecken i minnet som det finns i strängen 'ss' och laddar dessa med ASCII-representationen av tecknen. Om strängen skall innehålla ^postrofer måste dessa dubbeltecknas. Ex: SÄG: 'HEJ' skrivs som 'SÄG: "HEJ"'. Strängen får innehålla max 16 tecken. Den första reserverade byten får den adress som just då anges av adressräknaren, den andra reserverade byten får nästföljande adress, osv.

Pseudoinstruktionerna får ha lägen och kommentarer precis som alla andra satser. EQU-instruktionen har ingen effekt om den inte har ett läge (eller symbol).

SYMBOLER OCH LÄGEN

Den beteckning som skrivs i lägesfältet på en rad kallas för en symbol eller ett läge. I båda fallen representerar beteckningen ett 16-bitars heltal. Om man kallar beteckningen symbol eller för ett läge beror på hur den används. Det 16-bitars heltal som beteckningen representerar kan nämligen vara antingen en adress till något ställe i minnet, och då är det ett läge, eller en numerisk konstant, och då är det en symbol. För själva beteckningen gäller samma restriktioner och begreppen symbol eller läge kan betraktas som synonyma.

En symbol får bestå av ett till sex tecken. Det första tecknet måste vara en bokstav och de övriga får vara bokstäver, siffror eller något av tecknet understreck (_) eller frågetecken (?). Som bokstäver inräknas A-Ö samt É och Ü (se även Stora och små bokstäver). Symbolerna får inte innehålla mellanslag (blanktecken).

Prioritet	Operator
1	-
2	*
2	/
3	+
3	-

Exempel på tillåtna symboler:

```
PROG1
KLART?
DEL_1
```

För att en symbol skall kunna användas måste den definieras. En symbol definieras genom att den står i lägesfältet på en rad. Symbolen måste börja i första positionen på raden om den inte följs av ett kolon (:). En symbol får bara definieras på ett ställe i ett program. Försök att definiera en symbol på flera ställen resulterar i felutskrift. (symbolen får naturligtvis anropas hur många gånger som helst).

I assemblatorn finns en räknare som kallas adressräknare (Location Counter). Den används för att hålla reda på den aktuella adressen i minnet. När ett läge definieras tilldelas läget det värde, som just då anges av adressräknaren. Om en rad ser ut som:

```
PROG1  ORG 65408 ;Början av beräkningen
```

kommer beteckningen PROG1 (alltså läget) att tilldelas värdet 65408, som är adressen till programmets början. Om beteckningen i lägesfältet är en symbol, som står för en konstant som t ex

```
KONST  EQU 73 ;Första konstant
```

kommer beteckningen KONST (alltså symbolen) att tilldelas värdet 73.

Förutom de symboler som definieras i programmet finns det en särskild symbol, sol (o) som alltid har samma värde som det aktuella värdet i adressräknaren. Denna symbol kan användas när man vill referera ett antal bytes framåt eller bakåt i programmet.

Beteckningen i lägesfältet, som kan vara en symbol eller ett läge, representerar ett 16-bitars binärt heltal. Detta heltal kan antingen betraktas som ett positivt heltal mellan 0 och 65535, eller som ett heltal i 2-komplementär form mellan -32768 och 32767. Eftersom assemblatorn inte bryr sig om overflow vid addition och subtraktion spelar det ingen roll hur man ser på talen. Så länge man håller sig inom talområdet -32768 till 65535 kommer resultatet att bli riktigt. (Precis samma regler gäller ju även för heltal i ABC80-BASIC).

Det finns en gräns för hur många symboler eller lägen som får finnas i ett program. Denna gräns beror på hur stort minnet är och på vilken version (kassett eller flexskiva) av assemblatorn som används. Se "Begränsningar - kassettversionen" respektive "begränsningar - flexskiveversionen".

UTTRYCK

I många instruktioner förekommer uttryck som operander. Ett uttryck (expression) kan i det här sammanhanget bestå av en enda term eller av flera termer åtskilda av operatorer. En term kan vara en konstant eller en symbol. Följande operationer är tillåtna:

Betydelse	Exempel
negation	-2
multiplikation	2*VÄRDE
division	TAB/8
addition	A+B
subtraktion	BRUTTO-RABATT

Prioriteringen (dvs rangordningen) mellan operatorerna är precis samma som vid vanlig skolmatematik, dvs först utförs negation, därefter multiplikation och division, och slutligen addition och subraktion. Vid samma prioritet utförs operationerna från vänster till höger. Beräkningsgången kan ändras med parenteser, varvid uttrycket inom parenteser behandlas först, precis som vanligt. Observera dock att ett uttryck som är helt omslutet av parenteser betecknar en minnesadress. Exempelvis betyder 65408 värdet 65408 medan (65408) betyder värdet som finns på adress 65408.

Ett uttryck får inte innehålla mellanslag (blanktecken) eller kommatecken eftersom dessa tecken används för att avskilja fältet på raden.

Uttrycken behandlas precis som heltalsuttryck i ABC80-BASIC. Detta innebär bl a att man inte får overflow vid addition eller subtraktion. (Detta löser de tidigare problemen med negativa och positiva tal). Det innebär ochså att tecknet "/" står för heltalsdivision, dvs eventuella decimaler huggs av. 5/3 blir alltså 1.

Allt detta kan låta komplicerat. Den som är ovan vid räkning med 2-komplementära tal kan dock trösta sig med att man i praktiken inte märker någonting av allt detta. Uttrycken får helt enkelt de värden man väntar sig.

Ett uttryck behöver givetvis inte innehålla några operatorer. De allra flesta uttryck som ingår i ett program består i själva verket av en ensam konstant eller symbol.

KONSTANTER

Som tidigare nämnts får ett uttryck innehålla konstanter. Dessa kan vara av två typer - numeriska konstanter och teckenkonstanter. En numerisk konstant är helt enkelt ett heltal skrivet i någon lämplig bas. En teckenkonstant däremot är ASCII-representationen av ett tecken.

Numeriska konstanter

Assemblatorn kan hantera tal som är skrivna i flera olika talsystem (olika baser). Det är många gånger praktiskt att utnyttja denna möjlighet för att göra programmet lättare att förstå. Det är t ex ofta lämpligt att skriva tal som skall ingå i logiska operationer i binär form. Det är då lättare att se vilka bitar som påverkas.

Ett tal måste alltid inledas med en siffra, om eventuellt kan vara en inledande nolla. Talet får följas av en bokstav, som anger i vilken bas talet är skrivet. Utelämnas bokstaven tolkas talet som decimalt (bas 10).

Följande baser är tillåtna:

Bas	Bokstav
2 (binärt talsystem)	B
8 (oktalt talsystem)	O eller Q
10 (decimalt talsystem)	D eller ingen
16 (hexadecimalt talsystem)	H

Exempel:
Talet 217 kan skrivas på följande sätt:

11011001B, 331O, 331Q, 217D, 217, 0D9H

OBS: Om den inledande nollan i 0D9H utelämnas kommer D9H att tolkas som ett symbolnamn. Detta resulterar antagligen i felutskriften "Odefinierad symbol".

Teckenkonstanter

En teckenkonstant består av ett ASCII-tecken omgivet av apostrofer. Värdet av en teckenkonstant är lika med tecknets ASCII-representation.

Exempel på teckenkonstanter:

Konstant	Värde
'A'	41H
'a'	61H
'1'	31H
'4'	34H

Observera att citationstecken INTE dubbeltecknas om de ingår i en teckenkonstant.

Stora och små bokstäver

Assemblatorn skiljer normalt inte mellan stora och små bokstäver. Det finns tre undantag från denna regel:

1) teckenkonstanter
2) tecken i DEFB-sats
3) tecken i DEFM-sats

Detta innebär att satserna

```
HiT   Ld  A,0fH
och
hiT   LD  A,0fH
```

tolkas helt lika av assemblatorn. Däremot betecknar 'A' värdet 41H medan 'a' betecknar värdet 61H.

ALLMÄNT OM ASSEMBLATORN

Assemblatorns uppgift är att översätta ett program som är skrivet i assembler, dvs i form av op-koder, till maskinkod. Detta kallas assemblering. Det program som skall översättas (assembleras) kallas källprogram. medan det översatta programmet kallas objektprogram.

Innan ett program kan assembleras måste det matas in ock sparas på en fil. Detta görs lättast med hjälp av TV-editorn, men kan även göras enligt en annan metod.

Assemblatorn är en sk tvåpassassemblator. Detta innebär, att assemblatorn läser källkodsprogrammet två gånger. I första passet översätts alla instruktioner, vilka även kontrolleras med avseende på formell riktighet. I det andra passet beräknas alla relativa adresser. Om källkodsprogrammet finns på kassett bör det därför vara lagrat två gånger efter varandra.

Assemblatorn finns i två versioner. En kassettversion och en flexskiveversion. Här kommer närmast kassettversionen att beskrivas. Skillnaderna mellan flexskiveversionen och kassettversionen beskrivs senare i artikeln.

Som utdata ger assemblatorn dels ett objektprogram och dels en utskrift av programmet på bildskärmen. Denna utskrift kallas programlista. I flexskiveversionen skrivs objektkodprogrammet ut direkt på en fil. Detta innebär, att programmen kan vara i stort sett obegränsat stora. I kassettversionen däremot sparas objektkodprogrammet i minnet tills assembleringen är klar. Därefter skrivs det ut på kassetten. Detta begränsar programmets storlek till cirka 300 rader. I kassettversionen fås programlistan på bildskärmen och i flexskiveversionen kan programlistan dessutom fås på fil.

Handhavande, kassettversionen

Kassettversionen av ABC80-assemblatorn består av två delar. Dels själva programmet, ASMCAS och dels en fil, ASMCON, som ska ligga omedelbart after ASMCAS på kassetten.

När ett källprogram, som är lagrat på en fil på kassett, skall assembleras, blir tillvägagångssättet följande:

1. Ladda in assemblatorn till ABC80's minne. Följande metod är lättast att använda:
   Spola tillbaka kassetten med assemblatorn till början av kassetten. Skriv därefter RUN ASMCAS (eller RUN CAS:), varefter först programmet ASMCAS läses in, vilket i sin tur läser in filen ASMCON. När inläsningen är klar, kommer följande att visas på bildskärmen:

```
ABC 80 assembler

Vad skall listas på skärmen?

H = hela programmet

F = endast felaktiga rader

?
```

   Efter att ha svarat H eller F och tryckt på RETURN visas följande:

```
Tryck på RETURN när bandspelaren är
klar för PASS 1
```

2. Lägg i kassetten med källkodsprogrammet så att bandet står vid början av källfilen. Därefter kan man trycka på RETURN, varvid assembleringen börjar.

   I och med att ABC80-assemblatorn är en tvåpassassemblator kommer källkodsprogrammet att läsas två gånger. Är källkodsprogrammet lagrat två gånger behövs inga speciella åtgärder vidtagas mellan passen. Är källprogrammet däremot bara lagrat en gång på kassetten måste kassetten spolas tillbaka till början av källkodsprogrammet innan pass 2 kan påbörjas.

   OBS: Om fel upptäcks under pass 1 kommer pass 2 inte att utföras.

När assembleringen slutförts frågar assemblatorn om man vill att objektkodsprogrammet skall lagras på en fil. Vill man det måste man sätta i den kassett som man vill att objektkodsprogrammet skall sparas på och sätta bandspelaren på inspelning. Efter att man skrivit vad objektfilen ska heta, och tryckt på RETURN, spelas objektkodsprogrammet in. Objektkodsprogrammet är nu lagrat på kassetten i form av en eller flera POKE-satser med början vid den önskade minnesadressen (assembler-programmets startadress).

Vill man köra objektkodsprogrammet måste man ladda in detta i ABC80's minne precis på samma sätt som med ett vanligt BASIC-program - alltså med kommandot LOAD filnamn (alternativt LOAD CAS:). Detta resulterar i att objektprogrammet, som har formen av en eller flera POKE-satser, laddas in i minnet som ett BASIC-program. Kör man detta BASIC-program medför POKE-satserna att maskinkoden laddas in i minnet med början vid assemblerprogrammets startadress. Maskinkodsprogrammet kan sedan anropas med CALL(U%), där U% är programmets startadress. (Se även "Hur ett program används - Start, Indata").

Detta låter kanske lite rörigt, men studera exemplet så klarnar det säkert.

Begränsningar - kassettversionen

För kassettversionen i 16K-minne gäller följande begränsningar:

1. Symboler får vara maximalt 6 tecken långa.
2. Det får finnas högst 50 symboler i ett program.
3. Det får sammanlagt finnas högst 5 st ORG eller DEFS-satser i ett program.
4. Objektprogrammet får bli max 500 bytes långt.
5. Maximal radlängd är 78 tecken.

EXEMPEL PÅ ANVÄNDNING AV ABC80-ASSEMBLATORN (KASSETTVERSIONEN)

I detta exempel skall vi visa ett kort program som får bildskärmen att blinka.

Vi börjar med att skapa ett källkodsprogram med hjälp av programmet TV. Programmet ser ut så här:

```
        ORG   65408
        LD    HL,7C00H
        LD    BC,400H
BR:     SET   7,(HL)
        CPI
        RET   PO
        JR    BR
```

Starta upp TV-editorn samt skriv in ovanstående program och spara det två gånger på bandet som BLINK.ASM (ASM=källkod till assemblerprogram).

Kör sedan RUN ASMCAS och svara H eller F beroende på om du vill se alla rader eller endast de felaktiga. Tryck sedan på RETURN.

När ABC80 svarar med samma fråga en gång till, så, beroende på om du har sparat källkoden en eller två gånger, spolar du tillbaka programmet och trycker på RETURN eller så trycker du enbart på RETURN.

Byt nu kassett och sätt på inspelning samt svara J och tryck på RETURN.

När ASMCAS är klart kan du ladda in objektprogrammet genom att backa tillbaka och skriva LOAD CAS: När du nu tar LIST så ska du se ett par POKE-satser. Kör run och skriv sedan ; CALL(-128). Om skärmen inte blinkar har du gjort fel någonstans.

SKILLNADER MELLAN KASSETTVERSIONEN OCH FLEXSKIVEVERSIONEN

Det finns i huvudsak tre skillnader:

1. I flexskiveversionen kan i stort sett hur stora källfiler som helst användas p g a att där skrivs objektfilen direkt på skivan.
2. Man kan välja om man vill ha programlistan på bildskärmen eller på någon fil (eller printer).
3. Hanteringen av flexskive-versionen är något annorlunda, vilket beskrivs nån annan gång. (red)

Märk att flexskiveversionen består av programmen ASM, ASM2 och ASMCON. Samma begränsningar gäller som för kassettversionen.

FELMEDDELANDEN FRÅN ASSEMBLATORN

2) NAMN MED OTILLÅTNA TECKEN
Indikerar att ett namn (läge eller operand) innehåller otillåtna tecken.

3) OTILLÅTEN OP-KOD
Erhålls om op-koden innehåller otillåtna tecken.

4) OTILLÅTET TAL
Erhålls om ett tal innehåller otillåtna tecken inom specifierad talbas.

5) OTILLÅTEN OPERATOR

6) SYNTAX FEL
Erhålls om ett uttryck har fel format eller paranteser saknas.

7) ASSEMBLER FEL
Instruktionen har ej kunnat utföras. Detta kan bero på att raden är felaktig.

8) OKÄND SYMBOL
En symbol i operandfältet har ej definierats. Erhålls för felstavade eller ej definierade lägesnamn.

9) FELAKTIG KOMBINATION AV OPERANDER
Erhålls om ett registernamn eller villkorskod är felstavad eller felaktigt använd.

10 UTTRYCK EJ INOM INTERVALL
Indikerar att värdet för ett uttryck är för stort eller litet. Erhålls t ex vid overflow för 16-bitars aritmetik, division med 0 eller felaktigt värde för en byte.

11 DUBBLERAD DEKLARATION
Indikerar försök till omdefiniering av ett lägesnamn. Erhålls om en variabel är felstavad eller felaktigt använd vid flera tillfällen.

13 CITATTECKEN OBALANSERADE
Indikerar att en sträng ej är korrekt omgiven av citattecken eller att citattecken inom en sträng ej är korrekt grupperade i par.

För ytterligare information hänvisas till Z80 ASSEMBLY LANGUAGE MANUAL.

BREV

Följande brev fick vi alldeles när förra numret var klart. Det ger en del vinkar lika dem från finland samt endast för floppy-ister. Och dessa tips upprepas härmed. (red)

BREVET

Kalle Lindström
Vasseurs väg 35
182 35 Danderyd

1982 09 14
Gudda,gudda!
Detta lilla brev kommer från en 17-årig kille i Danderyd som är en stor ABC-diggare. Här kommer lite programmeringstips.

1. För att fylla arbetsminnet med varannan cell 0, varannan cell 255 så skriver man Z=CALL(292).
2. Det där med OUT 57,3 skall vi reda ut nu. Det stänger INTE av tangentbordet helt, utan man kan skriva men inte förrän repeatfunktionen börjar. Det blir litet svårare att skriva då förståss. Men man kan få bort det genom att skriva OUT 57,135 (om du klarar det dvs). För att stänga av CTRL-C skriver man enklast OUT 57,255,57,3 och för att stänga av hela tangentbordet skriver man OUT 57,255,57,0. Med OUT "säger" man att nästa byte till den här porten är en mask.
3. Om man i ett program efter frågan "Vill du spela igen" svarar "JA" skall det ju titt som tätt en massa variabler nollställas. Det kommer man undan med Z=CALL(3413). Detta är samma sak som run. Om man efter att ha stannat programmet vill fortsätta med bibehållna variabelvärden, kan man skriva Z=CALL-(3416). Vill man istället testköra ett program man håller på att utveckla, och man får ERR 6 vid run, så kan man skriva Z=CALL(3419). Då får man förståss se till så att man inte exekverar den rad man fick ERR 6 på. Detsamma gäller för NEXT etc.
4. För att få ett eget felmeddelande med stopp i ett program utan exekvering av STOP kan man skriva POKE 65408, 215,X. Vid Z=CALL(65408) ger det felmeddelande X. X - kan vara från 0 - 127. BASICERR.SYS påverkas ej av detta.
5. Du vet väl att när man ligger i terminalmode med ABCV24 kan få en hardcopy av skärmen på skrivare genom att trycka CTRL-Å. Det måste (tyvärr) vara en skrivare med serieinterface.
6. Man kan från ett program komma upp till operativ systemet genom att skriva Z=CALL(378).
7. Vet du att du kan använda båda sidor på en diskett? Gör så här: Klipp ett skrivskyddshål precis mittemot det ursprungliga. Tag sedan en linjal och mät ut hur långt det är från kanterna till indexhålet. Rita ett märke på exakt samma ställe, fast spegelvänt (mittemot alltså). Gör sen ett hål där. Se'n gör man samma procedur med indexhålet på baksidan. Nu är det dags för formatering. Jag har haft det så här i över ett år och har haft 1 (ett) fel hittills. Se bara till så att ni inte klipper in i skivan när ni klipper skrivskyddsjacket.

Med vänliga hälsningar Kalle Lindström

Obs ! Alla adresser är för checksumma 11273

HELP

av TED LIDSTRÖM

Jag köpte för en tid sedan 'HELP' för 400 kr, som numera är ett för mig oumbärligt hjälpmedel vid programmering.
HELP är alltså ett programmeringshjälpmedel för ABC80, med en rad nya kommandon.
HELP har för mig sparat mycket tid. Speciellt då kommandot 'FIND' som söker efter valfri text i programmet, bra att ha då man senare ska in och ändra i program och man letar efter något eller den förargliga dubbelanvändningen av variabler, som kan orsaka stora problem.
Mycket bra är också kommandot 'CAT' (Skivinnehåll) som gör att man kan få fram biblioteket med filstorlekar utan att förstöra något i minnet.

Här följer en uppräkning av de kommandon som finns i help med en kort förklaring:

FIND	Söka text
LIST	Listar 23 SKÄRMRADER och möjlighet för delutskrift på fil
DEV	Enheter i enhetslistan
LINE	Internkoderna i en BASIC-rad
MEM	Utskrift av systempekare och fritt minnesutrymme
LOOK	Listning av minnesinnehåll
VAR	Lista på använda variabler i ett program
CAT	Skivinnehåll med fil-storlekar och kvarvarande utrymme
OLD	Program som försvunnit vid reset kan tas fram igen
BOFA	Möjlighet att snabbt ställa om BOFA
PEW	Som PEEK men två bytes på en gång
POW	Motsvarande POKE men två bytes på en gång
HELP	Utskrift av HELP's kommandon
REN	Möjlighet till DEL-RENUMRERING av program
AUTO	Automatisk radnumrering
CON	Start på valfri rad med bibehållna variabelvärden
ONLY	Sätta/ta bort lässkydd på program

EXIT	Urkoppling HELP
CTRL-SHIFT O	Avbrytning direkt och uthopp till direktmode
CTRL-´	Stoppar exekvering tills annan tangent trycks
CTRL-T	Start av TRACE
CTRL-O	Stopp av TRACE
CTRL-L	Sudda skärm i direktmode
CTRL-Ä	Inkoppling av bildskärmseditor

När man köper HELP får man på skivan med ett antal UTILITY-program:

ROMRAM	Checksumma på ROM och test av RAM
ASCII	Omvandling av exponenttal till ASCII-sträng
PGMCOMP	Komprimerar BASIC-program så att de tar mindre plats i minnet
NOCTRLC	Inhibering av CTRL-C
READFILE	Läsning från fil sectorvis med data representerat decimalt
PRCRT	Styrning av printerutskrift till skärm med valbar hastighet

HELP lagras i RAM-minnet under DOS-buffrarna och kan laddas upp från skivan utan att förstöra programmet som man just har i minnet under förutsättning att utrymme finns ledigt under DOS-buffrarna.

HELP's bildskärms editor medför stora möjligheter att manipulera på skärmen med programraderna. Vad sägs om följande:

Packning av text
Ihopdragning/borttagning av text
Skärmdump på printer
Suddning av skärmen
Scrolla skärm
Skjuta ner hela rader så att utrymme fås
Cursorstyrning:Upp,Ner,Höger,Vänster
Lagring text med ---> (Högerpil)
Borttagning ur buffert med <--- (Vänsterpil)

Utskrift av inmatad text (Bra om man åkt runt på skärmen mycket)
Snabbhopp till nästa tecken eller mellanslag för borttagning m.m.
Snabbradering av buffert
'HOME' utan att sudda skärmen

Jag måste säga att jag är väldigt nöjd med min HELP, så jag rekommenderar den varmt till andra 'HEMMA-pulare' och programmerare.

Min HELP köpte jag av X-DATA i Bålsta, och dom lämnar säkert ytterligare info om du ringer eller skriver till X-DATA HB, Baldersvägen 22, 198 00 BÅLSTA tel. 0171-513 70.

Q-Zentralen - användarrapport

När jag nu haft tillgång till Q-Zentralen i en dryg månad tänkte jag dela med mig av en del erfarenheter jag gjort.

När du skickat in ansvarsförbindelsen och betalt inträdesavgiften får du tillsammans med programkassetten och manualen två s k PPN-nummer (Projektprogrammerarnummer) med HEMLIGA inloggningskoder. Ett PPN är ett slags behörighetskod som bl a reglerar under vilken tid och i vilken utsträckning du kan använda QZ:as dator ODEN. Dessa nummer reglerar också kostnaden för körningarna.

PPN XXX,XXX kan endast användas mellan kl 20.00 - 07.00 alla dagar. På detta PPN kan du köra KOM (se nedan) samt hämta program från programbanken. Kostnaden är ca 20 kr/tim på kvällen för att sjunka ner mot 10 kr/tim på natten.

Om du loggar in på det andra PPN:et får du tillgång till praktiskt taget hela ODEN:s resurser, d v s du kan i princip köra en stordator hemifrån i språk som ALGOL, APL, BASIC, COBOL, FORTRAN, LISP, PASCAL, SIMULA m fl. Här är det kanske på sin plats med ett par varningar:

1. Det kostar. Detta PPN är tillgängligt även dagtid, men kostnaden hamnar då över 100 kr/tim mot 30-40 kr/tim på natten.

2. I vilken grad en nybörjare egentligen kan förstöra något vet jag inte, men genom detta projekt får man ett oerhört kraftfullt instrument i sina händer som man troligtvis inte helt behärskar. Inte minst med tanke på körkostnaderna är det nog klokt att fråga sig fram något, exempelvis i KOM, innan man börjar experimentera.

Utan att logga in kan du läsa en mängd hjälptexter gratis. När ODEN promptar med en punkt, . , svarar du HELP KOM, HELP ABC, HELP <något annat>, eller bara HELP.

Om du vill ha en kurs i hur man använder datorn ODEN kan du logga in på "gratisnumret" PPN YY,YY, det kostar inget.

Vill du logga in svarar du LOGIN på promptern och följer instruktionerna i manualen. Observera att när ODEN första gången frågar efter namn skall du svara med ABC<medlemsnummer> t ex ABC2349.

KOM ----------

KOM är ett telemötessystem på QZ där deltagarna utväxlar brev och meddelanden om de mest skilda saker. Där träffar man folk från FOA, SCB, KTH, QZ m fl andra företag och instutitioner och diskuterar datateknik - alla nivåer, politik, löser gåtor, annonserar prylar och lägenheter m m. Detta sker dock ej huller om buller utan verksamheten är organiserad i s k möten vars namn anger ungefär vad som avhandlas i mötet, t ex ABC-Klubben medlemsforum. Det finns en mängd olika möten och man väljer själv i vilka möten man vill delta. Förutom att läsa och skriva inlägg kan man skicka personliga brev till andra deltagare i KOM. Varje gång man går in i KOM får man meddelande om vad som hänt sen sist, dels hur många brev man fått och dels hur många olästa inlägg man har i respektive möte man är medlem i. Motsvarande händer alltså om du skriver ett inlägg, alla medlemmar i mötet får meddelande om att det finns ett nytt inlägg. Själv hade jag en del frågor i början, bl a om en del KOM-kommandon som har modifierats och inte helt stämmer med manualen. En del fick jag svar på genom att läsa gamla inlägg och en del frågor lade jag in i mötet "ABC-novis Svenne" där jag snabbt fick svar. Tyvärr är det glest mellan inläggen från ABC-medlemmar. Jag tror ändå att det finns ett stort behov hos medlemmarna runt om i landet att "träffas" och diskutera, få hjälp osv. Själv skulle jag t ex gärna diskutera Forth på ABC-80. Klubbens monitor är utmärkt för utbyte av program, men jämfört med KOM ganska opraktisk för diskussioner. Dessutom kanske man inte i onödan skall belasta screening-gruppen som håller ordning på monitorns filer med en massa små filer när KOM gör detta automatiskt.

ABC-Klubbens QZ-projekt är som ni kanske tidigare läst i bladet en försöksverksamhet, för dataamatörer öppen endast för ABC-Klubbens medlemmar. Medlemsskapet i ABC-Klubben ger alltså en unik möjlighet att till starkt reducerad taxa pröva på "riktiga grejor".

Jag hoppas att snart få se fler ABC-medlemmar i KOM, jag är medlem i mötet "Presentation av nya KOM-deltagare" så jag ser direkt när ni dyker upp. Inte för att jag är expert på något vis men kontakta mig gärna om du har svårt att komma igång, antingen genom brev i KOM eller på telefon 08-60 23 98.

Anders Silv´en <2349>

Ovan PPN XXX,XXX och PPN YY,YY erhållés i samband med ansvarsförbindelse / manual. (RED)

Instruktioner till Hjälpare

Följande ConTRoL-kommandon finns:

CTRL-W	Flytta markören uppåt en rad.
CTRL-Z	Flytta markören nedåt en rad.
CTRL-A	Flytta markören en position åt vänster.
CTRL-S	Flytta markören en position åt höger.
CTRL-Q	Flytta markören till övre vänstra hörnet.
CTRL-U	Skriv ut det som finns i inmatningsbufferten på det ställe på skärmen där markören befann sig när <RETURN> sist trycktes.
CTRL-L	Töm skärmen och flytta markören till övre vänstra hörnet. Nollställer även inmatningsbufferten.
CTRL-P	Skjuter fram resten av texten på bildskärmen ett steg.
CTRL-D	Drar tillbaka resten av texten på bildskärmen ett steg.
CTRL-X	Töm inmatningsbufferten och flytta markören till radens början utan att radera på skärmen.
CTRL-E	Paus. Stannar exekvering, läsning på skiva, printerutskrift etc tills någon annan tangent trycks ned.
-->	Kopiera in "överkört" tecken i inmatningsbufferten.
<—	Tag bort ett tecken ur inmatningsbufferten och backa en position utan att radera något på skärmen.
CTRL-SHIFT-O	Bryter allting. Om den jobbade på disken kan det ta cirka 15 sekunder innan disken börjar läsa igen vid nytt anrop.

Följande extrakommandon finns:

DEL n1	Tar bort rad n1.
DEL n1-n2	Tar bort alla rader f o m rad n1 t o m rad n2.
DEL -	Tar bort hela programmet. Samma funktion som NEW.
DEL n1-	Tar bort alla rader f o m rad n1.
DEL -n1	Tar bort alla rader t o m rad n1. I stället för bindestreck kan kommatecken användas.
VAR	Listar alla variabler i ett program. Denna funktion går ej om det finns ett syntaxfel i programmet, exempelvis hopp till rad som inte finns, FOR utan NEXT etc.
RUN nn	Startar exekveringen på rad nn. Kommandot nollställer alla variabler och stänger alla filer.
CON nn	Fortsätter exekvering på rad nn. Nollställer inga variabler och stänger inga filer. Det går ej att fortsätta i subrutiner eller i loopar.
LIB	Visar diskettinnehållet på båda drivarna.
LIB0	Visar diskettinnehållet på drive 0.
LIB1	Visar diskettinnehållet på drive 1.
FIND	Letar upp en instruktion eller en textsträng i programmet. OBS! Kommandot innehåller en bug, så lita inte för mycket på resultatet.
RAD	Visar antalet rader i programmet.
RAD n1	Visar på vilken adress rad n1 ligger lagrad på.
PEW nn	Peek word, dvs samma som ;PEEK(nn)+256*PEEK(nn+1).
POW n1,nn	Poke word, samma som POKE n1,nn,SWAP%(nn).
SYS	Visar BOFA-, EOFA-, HEAP och TAKET-pekarna samt hur många bytes programmet upptar (FILE) och hur många bytes som finns lediga (FREE).
BOFA	Visar värdet på BOFA-pekaren.
BOFA nn	Ställer om värdet på BOFA-pekaren till nn.
TAKET	Visar värdet på TAKET-pekaren.
TAKET nn	Ändrar värdet på TAKET-pekaren till nn.
HELP	Visar alla tillgängliga kommandon.
EXIT	Avslutar HJÄLPARE. Ställer om BOFA-pekaren och I-registret till uppstartsvärden.
OLD	Tar tillbaka ett program och gör det körbart igen efter NEW, SCR, DEL eller CHAIN "". Om du har otur så kan det gå åt helsike på grund av att någon internkod (exempelvis heltal mellan 3328 och 3583) innehåller talet 13, vilket är detsamma som radslut.
AUTO	Ger automatisk radnumrering från rad 10 med intervall 10. Om man redan har matat in rader kommer kommandot att börja från sista radnummer + 10.
AUTO nn	Ger automatisk radnumrering från rad nn med intervall 10.
AUTO n1,n	Ger automatisk radnumrering från rad n1 med intervall n.
LIST	Fungerar som vanliga LIST, förutom att endast de 23 första skärmraderna listas utan tangentnedtryckning, inte som på vanliga LIST där de 23 första programraderna listas och man oftast inte ser början på programmet.

Hoppas att det går bra. Funtionerna CTRL-P, CTRL-D, PEW, POW, AUTO, OLD, SYS, BOFA EOFA och RAD är gjorda av The Computer Phantome alias Kalle Lindström. Resten är gjort av Niclas Wiberg.