ABC BLADET

ABC-KLUBBENS MEDLEMSBLAD FÖR BLANDAD INFORMATION TILL BÅDE NYTTA OCH NÖJE

Nummer 3, 1982

GRATTIS ! ! !

Lasse Karlsson

och

DIAB

till Elektronikindustriföreningens stipendium.

Stipendiet utdelades med motiveringen att DIAB bäst lyckats förena Teknik, Framsynthet, Vitalitet och Ekonomi i sina satsningar under året.

Valberedningen

vill ha in namn på personer villiga att arbeta i och för ABC-klubben.

Vid årsmötet 1982, tillsattes en valberedning för att förbereda personvalen av klubbfunktionärer till årsmötet den tolfte februari 1983.

Valberedningen önskar kontakt med personer, som är beredda att **engagera sig** i ABC-klubben.

Förslagen sänds helst in på en lapp till någon av nedanstående personer i valberedningen. Det går förstås också att ringa.

Jockum Wahlberg, Lexicon
Box 136, 182 12 Danderyd
tel 753 3140

Dieter Leve, T-D-X Smådatorer
Kungsgatan 79, 112 27 Stockholm
tel 521060/521460

Sten Staxler, Infotronics och Teleplan
Murarstigen 17, 181 46 Lidingö
tel arb 08/98 1000, bost 08/767 0742.

Medlemsorgan för
ABC-klubben
Vidängsvägen 1
161 33 Bromma
ISSN 0349-3653

Ansvarig utgivare:
Gunnar Tidner

Redaktör:
Claes Schibler

Monteringshjälp Claes Schibler

ABC-klubbens
postgiro 15 33 36-3
telefon 08-80 15 22
(automatisk telefonsvarare
med aktuell klubbinformation)
telefon 08-80 15 23
(modem med ABC Monitor)

Tryck: Märstatryck AB 1982

Nummer 3, 1982

INNEHÅLL

Medlemsavgifter 1982

Seniorer 125 Skr
Juniorer 70 Skr

Junior räknas man t o m det kalenderår man fyller 18 år. Ange därför personnummer när Du betalar medlemsavgiften.
Medlemskapet är personligt.
Särskild prislista för prenumeration tillhandahålles på begäran.

Medlem blir Du enklast genom att sätta in medlemsavgiften på ABC-klubbens postgirokonto 15 33 36-3.

ABC-klubbens styrelse 1982

Ordförande:	Gunnar Tidner
Vice ordförande:	Kjell-Åke Johansson
Sekreterare:	Ulf Sjöstrand
Kassör:	Marianne Forsman
Redaktör:	Claes Schibler
Ledamot:	Joe Johnsson
Suppleanter:	Allan Larsson
	Björn Sjöborg
	Göran Sundqvist

LEDAREN

Snart har **ABC-klubben fått sin 3000:e medlem.** På mindre än tre år har medlemsantalet i klubben 12-dubblats. Att **ABC-80 är hemdatorn** står nu utom varje tvivel. Och Luxor har sänkt priset för behålla att sin dominerande ställning.

En sådan utveckling är givetvis glädjande men medför helt naturligt vissa problem genom att kraven på klubben stiger i takt härmed. Jag vill inte sticka under stol med att vi särskilt under hösten har haft vissa svårigheter att hinna med att expediera alla beställningar som strömmat in på nytt och gammalt material.

Det är också naturligt att det bland klubbens medlemmar finns stora variationer i kunskapsnivå. Flera av de som hört av sig har nog tyckt att nivån på materialet både i tidningen och på kassetterna ofta ligger en bit över den nivå där dom själva ligger. Från styrelsens sida har det varit en medveten strävan att åstadkomma en avvägd blandning vad gäller kraven på förkunskaper.

Att vi inte har lyckats med detta i samma utsträckning som vi önskat beror på att det mesta av materialet kommer från sådana som har haft sin ABC-80 under längre tid än genomsnittet och därför också ligger lite före i kunskaper.

Vi vill gärna till tidningen ha ett fylligare material som tar upp sådana saker som för den mer erfarne användaren ter sig triviala men som kan vara besvärliga stötestenar för nybörjaren.

Den som är ny i klubben kan ha stor glädje av att studera äldre nummer av ABC-bladet. Vi har givit ut ett särskilt samlingsnummer som innehåller textsidorna (utom annonser) för hela årgång 1980 + 1981. Tillsammans med ABC-kassett nr 1 och 2 kostar det bara 100 kr inkl frakt.

Var inte rädd för att skicka in bidrag till tidningen bara för att du tror att det som publiceras måste ligga på en väldigt hög nivå. Tvärtom vi vill ha material som särskilt inriktar sig på att hjälpa alla nytillkomna.

Vi vill också gärna ha inlägg från folk som berättar om hur de använder sin ABC-80/800.

När det gäller programmen på ABC-kassetterna så är det oftast återkommande önskemålet att få bättre och utförligare dokumentation över vad programmen gör.

Antalet utgivna program är nu så stort att vi kommer att ha väldigt stora svårigheter att få fram dokumentationen om vi inte kan påräkna hjälp med detta från fler av medlemmarna. Man behöver inte själv ha gjort programmet för att kunna hjälpa till med dokumentationen. Det går utmärkt att sätta ihop köranvisning och dokumentation till program som redan ligger på ABC-kassett. Det krävs ofta inte särskilt mycket för att förbättra den knapphändiga dokumentation som hittills finns.

Misströsta inte om du tycker att programmen på kassetterna ibland är mer eller mindre helt obegripliga. Det gäller skärskilt program skrivna i maskinkod. Sådana program kan faktiskt vara både kraftfulla och väldigt nyttiga. Men behöver inte alltid förstå i detalj hur ett program arbetar för att kunna ha nytta av det. För att kunna njuta av en god måltid behöver man inte helt förstå matsmältningsprocessen som den engelske fysikern Heaviside lär ha uttryckt det.

Låt din ABC-80 komma till användning! Tillhör du dem som ställt undan din ABC-80 och inte använder den längre? Sälj den då till någon som har större användning för den. Vi tar utan kostnad emot radannonser i ABC-bladet från medlemmar som vill sälja sin datorutrusning eller köpa begagnat.

Gunnar Tidner

NYTT FRÅN ABC-Sthlm

Stockholmskretsen av ABC-klubben

INLEDANDE VÅRMÖTEN.

Plats: Vidängssalen

Vidängsvägen 1
Alvik

Tid: Onsdagar kl 19.00

MED FÖLJANDE ÄMNESOMRÅDEN:

83 01 19 Kvällens tema, **KASSETTPROGRAMMEN.**

Representanter för programgruppen inom ABC-klubben redovisar några av de viktigare programmen, som kommit ut på kassetterna.

STIG LÖFGREN <872> berättar om olika fel på kassetter och bandspelare. Åtgärder för att förhindra ERR. vid inladdning av program.

83 02 16 Kvällens tema, **DATAKOMMUNIKATION.**

GUNNAR TIDNER <1306> informerar om kommunikationen med Stockholms Datamaskincentral, QZ.

DAG ISSJÖ <611> berättar om modem och modemanvändning.

83 03 16 (preleminärt) **ÅRSMÖTE.**

Diskussion om Stockholms-kretsens kommande verksamhet.

ABC-kvällar, i klubblokalen på Vidängsvägen 1.

Tisdagar kl 18.30 - 22.00 tills vidare.

ABC-Sthlm,s öppethållande av klubblokalen för att ge medlemmarna möjlighet att träffas mer informellt och även stämma träff med varandra. Vidare ges möjligheter till:

* Programbyten mellan medlemmarna.

* Möjligheter till kopiering av program. Tomma kassetter och skivor kan köpas.

* Datalitteratur kan lånas från klubbens bibliotek, som är under uppbyggnad.

* Anslagstavla för medlemmarna, sälj och köpmeddelanden mm.

Kallelse till ABC-dagen med årsmöte i ABC-klubben

Tid: Lördagen den 12 februari 1983
Plats: Brommasalen Kommunalhuset i Alvik, Gustavslundsvägen 168 Bromma, T-banestation ALVIK.

I likhet med tidigare år räknar vi med att på ABC-dagen arrangera en utställning av utrustning och tillbehör till ABC80. Även denna gång hade vi tänkt ge de små företagen i branchen en chans att visa sina produkter och intressanta nyheter. En del leverantörer kommer också att ha försäljning av enklare tillbehör och förbrukningsmaterial. Utställningen är öppen 11.oo-13.oo och 14.3o-17.oo

Program: 11.oo Utställningen öppnar

13.oo Årsmötesförhandlingar

14.3o (c:a) Paus för besök på utställningen eller klubblokalen i samma hus.

15.3o Frågestund. Här har du tillfälle att ställa alla möjliga och omöjliga frågor till en panel av experter på ABC80.

19.oo Gemensam middag.

Förslag till dagordning:

1. Val av mötesordförande.
2. Val av mötessekreterare.
3. Val av två justeringsmän, tillika rösträknare att jämte ordföranden justera årsmötesprotokollet
4. Fråga om mötet är behörigen utlyst.
5. Fastställande av dagordning.
6. Styrelsens redovisningshandlingar.
7. Föredragning och godkännande av revisionsberättelse.
8. Fråga om ansvarsfrihet för styrelsens ledamöter.
9. Fastställande av balansräkning.
10. Beslut med anledning av vinst eller förlust enligt balansräkningen.
11. Fastställande av budget och medlemsavgift.
12. Val av ordförande och vice ordförande samt övriga styrelseledamöter och suppleanter för ett år.
13. Val av två revisorer och en revisorssuppleant för ett år.
14. Val av valberedning om minst två personer.
15. Behandling av ärenden som styrelsen vill förelägga årsmötet.
16. Behandling av motioner som medlemmarna inkommit med senast sex dagar före mötet.
17. Övriga frågor.

Eventuella frågor skall ha inkommit senast fredagen den fjärde (4) februari 1983 till ABC-klubben, Vidängsvägen 1, 161 33 Bromma.

Matematikpaketet


```
**************************
* BRUKSAN1.BAC           *
* BRUKSAN2.BAC           *
* VER 1.0 / 1982-09-28   *
* Gjort o. insänt av 2415 *
* Stig Rosenlund         *
* Västmannagatan 93      *
* 113 43 Stockholm       *
* Tel. 08/331736         *
**************************
```


I 16K-versionen används P16 (gör POKE 65053,230) och ASP16K för inmatning av assemblerrutinerna.
NYPREC16 bestämmer precisionen och VÄRD16 används i stället för VÄRD.
NOLLSÖK och INTEGRAL går bra direkt.
I övriga BASIC-program adderas 16384 till varje adress mellan 32768 och 36626, liksom i bruksanvisningen nedan för 32K-versionen.
I KASFIL, FLEXFIL och BINUT behöver bara rad 10 och 20 ändras.

Matematikpaket ABC80 32 K RAM

Innehåll :

115 Operationskoder

Här ges en tablå över de koder som används vid kalkyler och vid programskrivning i VÄRD med

```
FNU%(R%,I%,K%,U)
FNM%(R%,I%,M%,K%)
FNK%(R%,I%,K%)
FNI%(R%,I%)
FNR%(R%).

U = 0, 1, ... , 65535.

I% = 8, ... , F%.
```

Rutiner R% på adress 36304+3R%

R%	Resultat	
1	R8=...=R(F%)=0	Nollställning R8 tom R(F%)
2	R(U+1)=...=R(F%)=0	Nollställning R(U+1) tom R(F%)
3	S%=SGN(R(I%))	Signum
4	S%=SGN(ABS(R(I%))-1/2)	Signum
5	S%=SGN(R(I%)-U)	Signum
6	S%=SGN(ABS(R(I%))-U)	Signum
7	S%=SGN(R(I%)-R(K%))	Signum
8	S%=SGN(R(M%)-R(K%))	Signum
9	R(I%)=R(M%)+R(K%)	Addition
10	R(I%)=R(M%)-R(K%)	Subtraktion
11	R(I%)=R(M%)R(K%)	Multiplikation
12	R(I%)=R(M%)/R(K%)	Division
13	R(I%)=ABS(R(M%))^R(K%)	Exponentiering
14	R(I%)=min(R(M%),R(K%))	Minimumfunktion
15	R(I%)=max(R(M%),R(K%))	Maximumfunktion
16	R(I%)=U	Inmatning positivt heltal 0-65535
17	R(I%)=0	Nollställning
18	R(I%)=1	Ettställning
19	R(I%)=R(I%)+1	Öka med ett
20	R(I%)=R(I%)-1	Minska med ett
21	R(I%)=R(K%)^U	Exponentiering med heltalsexponent
22	R(I%)=R(K%)	Kopiering av R(K) till R(I)
23	R(I%)=-R(K%)	Kopiering med omvänt tecken
24	R(I%)=-R(I%)	Teckenändring
25	R(I%)=SGN(R(K%))	Signum på variabel
26	R(I%)=SGN(R(I%))	Signum på variabel
27	R(I%)=ABS(R(K%))	Absolutbelopp på variabel
28	R(I%)=ABS(R(I%))	Absolutbelopp på variabel
29	R(I%)=FIX(R(K%))	Heltalsdel av R(K)
30	R(I%)=FIX(R(I%))	Heltalsdel av R(I)
31	R(I%)=INT(R(K%))	Heltalsfunktion
32	R(I%)=INT(R(I%))	Heltalsfunktion
33	R(I%)=R(K%)-FIX(R(K%))	Decimaldel av R(K)
34	R(I%)=R(I%)-FIX(R(I%))	Decimaldel av R(I)
35	R(I%)=1/R(K%)	Invertering
36	R(I%)=1/R(I%)	Invertering
37	R(I%)=R(K%)^2	Kvadrering
38	R(I%)=R(I%)^2	Kvadrering
39	R(I%)=2R(K%)	Dubblering
40	R(I%)=2R(I%)	Dubblering
41	R(I%)=R(K%)/2	Halvering
42	R(I%)=R(I%)/2	Halvering
43	R(I%)=R(K%)2^U	Två upphöjt till U ggr talet
44	R(I%)=R(I%)2^U	Två upphöjt till U ggr talet
45	R(I%)=R(K%)2^-U	Två upphöjt till -U ggr talet
46	R(I%)=R(I%)2^-U	Två upphöjt till -U ggr talet
47	R(I%)=SQR(R(K%))	Kvadratrot
48	R(I%)=SQR(R(I%))	Kvadratrot
49	R(I%)=exp(R(K%))	Exponentialfunktion
50	R(I%)=exp(R(I%))	Exponentialfunktion
51	R(I%)=sin(R(K%))	Sinusfunktion
52	R(I%)=sin(R(I%))	Sinusfunktion
53	R(I%)=cos(R(K%))	Cosinusfunktion
54	R(I%)=cos(R(I%))	Cosinusfunktion
55	R(I%)=arcsin(R(K%))	Arcsinusfunktion
56	R(I%)=arcsin(R(I%))	Arcsinusfunktion
57	R(I%)=arccos(R(K%))	Arccosinusfunktion
58	R(I%)=arccos(R(I%))	Arccosinusfunktion
59	R(I%)=arctan(R(K%))	Arctanfunktion
60	R(I%)=arctan(R(I%))	Arctanfunktion
61	R(I%)=log(ABS(R(K%)))	Naturliga logaritmen
62	R(I%)=log(ABS(R(I%)))	Naturliga logaritmen
63	R(I%)=10log(ABS(R(K%)))	Tio-logaritmen
64	R(I%)=10log(ABS(R(I%)))	Tio-logaritmen
65	R(I%)=FI(R(K%))	Normalfördelning
66	R(I%)=FI(R(I%))	Normalfördelning
67	R(I%)=f1(R(K%))	Egen-definierad funktion
68	R(I%)=f1(R(I%))	Egen-definierad funktion
69	R(I%)=f2(R(K%))	Egen-definierad funktion
70	R(I%)=f2(R(I%))	Egen-definierad funktion
71	R(I%)=f3(R(K%))	Egen-definierad funktion
72	R(I%)=f3(R(I%))	Egen-definierad funktion

202 Inmatning av assemblerrutinerna

POKE 65053,187 och ladda och kör PROG1.

Genom svaret på första frågan reserverar man plats för egna assemblerrutiner. Svara 21 om Du vill prova den enkla maskinspråksrutinen nedan (se 660 Egna assemblerprogram). Svara 0 om Du inte programmerar i assembler. Med svaret på andra frågan väljer Du precision N = antalet byte i mantissan i den binära flyttalsrepresentationen för ett tal. Antal decimalsiffror är 2.4N. Svaret 5 ger 12 decimalsiffror, svaret 21 ger 50 decimalsiffror. Eftersom beräkningstiden ökar med precisionen bör Du välja lågt N medan Du lär Dig systemet. Efter svaret inmatas assemblerprogrammet på adress 32768 t o m 36626. Tid 30 sekunder.

Med svaret på frågan "Var ska BASIC-programmen börja?" bestämmer Du startadressen för VÄRD och de övriga BASIC-programmen i paketet som baseras på assemblerrutinerna. Systemets variabler Ri lagras mellan assemblerrutinerna och BASIC-programmen. Ju högre svar Du ger, desto fler variabler Ri och desto mindre utrymme för BASIC-program inklusive egna tillämpningar får Du. Svaret 50000 täcker de flesta behov. Efter detta svar beräknas log2 och PI. Det tar ca 0.03N^2 sekunder. Vi får R6 = log2 och R7 = PI. Dessa tal ligger till grund för de logaritmiska respektive trigonometriska och cyklometriska funktionerna i paketet.

Sedan kan Du definiera tre egna funktioner (MacLaurinutvecklingar). Se nedan (461 Assemblerprogrammets .....). Därefter är det klart för att köra de övriga programmen. Med programmet NYPREC kan Du besvara alla frågorna igen utan att behöva mata in assemblerrutinerna på nytt.

258 Enstaka kalkyler, in- och ut-matning av variabler

Kör VÄRD. Programmet förblir i den av de tre delarna Du valt tills Du avbryter genom att bara trycka RETURN. Vid Utmatning anger Du I% och får ut R(I%). Vid Inmatning anger Du I% och sedan R(I%), med valfritt antal siffror, med eller utan valfritt placerad decimalpunkt och med eller utan 10exponent enligt BASIC-konvention.
Vid Beräkning anger Du:

R%, I%, M%, K% och U (U = 0, 1,... , 65535)

och får ut R(I%) efter utförande av operation R% enligt operationskodtablån ovan.
Ange t ex R%=21, I%=8, K%=7, U=12345 (M% godtycklig t ex 8), så får Du ut
R8 = 2.066...E6137 = PI^12345.

Vid rutin 2 måste U vara lägst 7 för att inte log2 och PI skall förstöras. I övrigt kollar programmet att Dina angivna värden är tillåtna. CTRL C för stopp får effekt först sedan aktuell maskinspråksrutin avslutats och återhopp skett till BASIC.
Enstaka beräkningar kan även göras som kommando, t ex ger

P=FNU%(21,8,7,12345)

när VÄRD är i minnet samma resultat som ovan.

296 Egna BASIC-program

För egna längre beräkningar skriver Du ett BASIC-program med 600 som första radnummer.
Gör MERGE med VÄRD och ändra siffran 580 till 600 sist på rad 280, så utförs Ditt program vid val av Beräkning. Du kan efter mönster av rad 580 - 590 bestämma värden på R% - U och göra GOSUB 560, så får Du en tillåtlighetskoll. Rutinerna 3 - 8 används vid villkorliga hopp. För kortare programskrivning använder Du de i VÄRD definierade funktionerna.
Vilken som helst av

```
FNU%(R%,I%,K%,U)
FNM%(R%,I%,M%,K%)
FNK%(R%,I%,K%)
FNI%(R%,I%)
FNR%(R%)
```

kan användas för rutin R% (R% = 1, ... , 72). De parametrar som inte är argument bestäms separat.

Exempelvis får Du

R23 = sin(R9) ur S%=FNK%(51%,23%,9%)
eller K%=9% : S%=FNI%(51%,23%)
eller K%=9% : I%=23% : S%= FNR%(51%).

Sista tillgängliga variabeln är R(F%), där
F% = PEEK(34443)+256PEEK(34444).
U = 0, 1, ... , 65535.

Förstör inte R6 och R7. Observera att assemblerrutinerna använder R0 - R5 för mellanresultat.
Rutinerna 12 (division) och 21 (exponentiering med heltalsexponent) använder R0 och R1.
Rutinerna 35, 36, 47 och 48 (invertering och kvadratrot) använder R0 - R2.
Rutinerna 67 - 72 (f1 - f3) använder R0-R4.
Rutinerna 13 och 49 - 66 använder R0 - R5.
Övriga använder endast R0.

Vid Inmatning och Utmatning av tal i VÄRD och SEKVUTIN används R0 - R4.
För inmatning av icke-negativa heltal och för de fyra räknesätten använder Du även

FNÄ%(I%,U) som ger R(I%)=U;
FNS%(I%,M%,K%) som ger R(I%)= R(M%)+R(K%) (minnesregel Sum);
FND%(I%,M%,K%) som ger R(I%)= R(M%)-R(K%) (Difference);
FNP%(I%,M%,K%) som ger R(I%)= R(M%)R(K%) (Product);
FNQ%(I%,M%,K%) som ger R(I%)= R(M%)/R(K%) (Quotient).

Då R% = 3 - 8 i FNU% - FNR% fås signumvärdet som FNV%-funktionens värde. Exempelvis får FNK%(7%,40%,15%) värdet SGN(R40-R15).

Med GOSUB 330 får Du R(I%) = B$
och med GOSUB 450 får Du B$ = R(I%).

Exempelvis ger B$="-123.45e678" : I%=8% : GOSUB 330

att R8 = -1.2345E680.

Detta medger också inmatning i DATA-satser. Blir det trångt om minnesutrymme kan Du förutom alla REM-satser även stryka ett eller fler av blocken Inmatning på rad 300 - 420, Utmatning på rad 430 - 540 och Beräkning på rad 550 - 590 i den mån Du inte använder deras subrutiner. Modifiera då rad 280 - 290. Blocken är oberoende av varandra, bortsett från GOSUB 450 rad 590. BASIC-variablerna A0, N och F% får ej förändras om de tre blocken ska fungera. Parametrarna R% - U förändras inte av GOSUB 330 och GOSUB 450. Se i övrigt programlistan för använda BASIC-variabler. Med NUM$ och VAL omvandlas Ri och BASIC-variabler sinsemellan via B$. Vid omvandling med VAL skall N vara högst 29.

Exempel på programskrivning :

```
600 FOR U=1 TO 10 : I%=U+7% :
    S%=FNÄ%(I%,U)+FNR%(24%)+FNR%(50%) :
    GOSUB 450 : ; B$ : NEXT U
```

ger Ri = exp(-(i-7)) för i = 8,... , 17

och en utmatning av dessa tal på skärmen.

Se vidare VÄRD, INTEGRAL och NOLLSÖK för fler programexempel.

Om floppyn slagits ifrån under en lång beräkning kan den kopplas in igen utan att assemblerprogrammet och Ri-na förstörs. Anteckna b = PEEK(65052) + 256PEEK(65053) = BOFA. Slå till strömmen till floppyn och gör Reset. Gör kommandot POKE 65052,b,SWAP%(b). Gör NEW och POKE 32768,42.

419 Övriga BASIC-program

NOLLSÖK och INTEGRAL skall mergas med VÄRD. De finner nollställen respektive integraler till funktioner. Se bruksanvisning i REM-satser. SEKVUTIN matar ut och in tal sekventiellt. Mer än tre rader siffror på skärmen kan accepteras. Om N är större än 379 (mer än 909 decimalsiffror) kan utmatningen avbrytas innan de första siffrorna försvinner från skärmen. Vill Du ha möjlighet att när som helst avbryta för att sedan kunna återuppta utmatningen skriv till satsen

335 IF INP(56%)=60% GET A$.

Utmatningen avbryts då med < knappen och återupptas med någon annan knapp. I samma programavsnitt kan man också gå in för att programmera en sekventiell printerutskrift av utmatningen. BINUT matar ut och in tal på binär flyttalsform. FLEXFIL och KASFIL lagrar och hämtar variabler Ri på flexskiva respektive kassett. Antag att man lagrar R12 t o m R17 på en fil. Vid hämtning från filen till R25 och R26 blir R26 f d R17 och R25 blir f d R16. Hämtningen sker alltså uppifrån och ner. Andra variabler än R25 och R26 förändras ej. Om endast ett tal hämtas kan N vara mindre än vid lagringen, annars skall N vara detsamma.

461 Assemblerprogrammets och variablernas struktur

Syftet med programpaketet är att möjliggöra beräkningar med valfri precision. Hög precision krävs i många illa-konditionerade numeriska problem. Precisionen är lägst N = 4 (9 decimalsiffror). Övre gränsen för precisionen bestäms dels av tillgängligt minnesutrymme, dels av konvergenssnabbheten hos MacLaurinserien f(u) som ligger till grund för funktionsberäkningarna. Denna serie definieras rekursivt av

$$a_0(u) = 1$$

$$a_k(u) = (b_1k+b_2)ua_{k-1}(u)/((b_3k+b_4)(b_5k+b_6)) \quad \text{då } k \geq 1$$

$$f(u) = \sum_{1}^{\infty} a_k(u)/(b_7k+b_8).$$

Alla funktioner fr o m rutin 49 samt rutin 13 använder denna oändliga serieutveckling. Summeringen avslutas när en term inte längre ger något bidrag till summan. Parametrarna b1 b8 är ickenegativa heltal. De tre funktionerna f1 - f3 definieras genom angivande av parametrarna. För snabb beräkning lagras b1*k+b2 etc som tvåbytetal. Om så många termer krävs att två byte inte räcker fungerar inte rutinen. För rutinerna 13 och 49 - 66 i paketet är b1, b3, b5 och b7 högst 2 och f(u) konvergerar minst så snabbt som en geometrisk serie med kvot 1/2. För dessa fall fungerar f(u) för N upp till 4090 (9816 decimalsiffror). Om högre precision än 3362 decimalsiffror önskas används NYPREC för att bestämma precisionen. Med ledning av givna uppgifter om hur assemblerrutinerna använder R0 - R5 för mellanresultat kan Du ändra kontrollerna rad 90 - 100 efter Dina behov.
Exempel på beräkning med f(u).

Betrakta utvecklingen

$$\arcsin(x) = \sum_{k=0}^{\infty} \frac{(2k-1)!!}{(2k)!!} \frac{x^{2k+1}}{2k+1} =$$

$$= x + \frac{x^3}{6} + \frac{x^3}{2} \sum_{r=1}^{\infty} \frac{2(2r+1)!!}{(2r+2)!!} \cdot \frac{x^{2r}}{2r+3}$$

Man ser att man kan skriva

$$\arcsin(x) = x + \frac{x^3}{6} + \frac{x^3}{2} f(x^2)$$

där

b1 = 2, b2 = 1, b3 = 2, b4 = 2, b5 = 0, b6 = 1, b7 = 2, b8 = 3.

Matematikpaketet använder denna utveckling för ABS( ) <=1/2. För ABS( ) > 1/2 används en omformning för att öka konvergenshastigheten. Utvecklingen ger också PI =6arcsin(1/2). Serien f(u) är inte bara tillräckligt generell för att medge beräkning av elementära och cyklometriska funktioner, utan en mängd andra funktioner kan också beräknas, t ex Besselfunktioner och integraler av funktioner som sin(x)/x. Om u kan skrivas binärt med ett begränsat antal byte (t ex u = 5/16) ökar beräkningstiden för f(u) högst med N^2, annars (t ex u = 0.3) högst med N^3.Vid multiplikation och division söker nämligen programmet igenom variablerna för att avgöra hur många signifikanta byte de har och opererar sedan endast på dessa byte. Vid hög precision betyder detta en avsevärd tidsbesparing. Vid låg precision märks dock genomsökningstiden i förhållande till själva operationstiden. Detta, tillsammans med att det vid fast precision finns möjlighet att förkorta beräkningstiden genom användning av fasta konstanter och polynomapproximationer m m, gör att programpaketet inte är lämpligt för daglig beräkning där snabbhet krävs och precisionen inte behöver vara större än t ex 16 siffror. En ABC800 är bättre. Vid denna precision är programpaketets snabbhet vid funktionsberäkningar ungefär som en programmerbar kalkylator. De fyra räknesätten och kvadratrot går dock snabbt. Särskilda omformningar görs för de elementära och cyklometriska funktionerna för att hålla argumentet för MacLaurinserien litet och beräkningstiden begränsad. Härigenom blir också normalt funktionsvärdena korrekta så när som på högst tre decimalsiffror. Undantag är sin och cos, där antalet inkorrekta siffror ökar med logaritmen på argumentets absolutbelopp (liksom hos alla programspråk och kalkylatorer). Vid normala argument märks detta dock inte. I rutinerna 13 och 21 blir det högst tre felaktiga siffror om tioexponenten i svaret är mellan -100 och 100, men vid stora absolutvärden på exponenten kan det bli upp till fem inkorrekta siffror.

Assemblerrutinerna lagras i minnet på adress 32768 t o m s, där s = 36626 plus antal byte för Ditt eget assemblerprogram. Talet Ri upptar adress Ai-N-2 t o m Ai, där Ai=s+6+(i+1)(N+3).
Med c(j,r) = bit nr r av byte nr j är

$$Ri = (1-2c_{A_i,0})(0.c_{A_i-3,7}\ c_{A_i-3,6}\cdots \cdots c_{A_i-N-2,0})2^{(1-2c_{A_i,1})(c_{A_i-1,7}\cdots c_{A_i-2,0})}$$

på oegentligt blandad binär-decimal form. Bitarna 2 t o m 7 i byte Ai används inte och förändras slumpmässigt. Enda villkoret i denna representation är att c(Ai-3,7) = 0 om och endast om Ri = 0. Gränserna för talens absolutbelopp är alltså 2^-65536 och 2^65535, motsvarande tioexponenterna -19728 och 19728. Om resultatet av en beräkning över- eller underskrider en gräns fås i stället ett tal nära gränsen (t ex 2^65533) utan att felmeddelande ges. Vid otillåtna operationer fås följande resultat.

För y/0 får man 1 då y = 0 och ca SGN(y)2^65535 annars.

För SQR(y) fås -SQR(-y) då y<0.

För arcsin och arccos används värdet vid 1 då y>1 och värdet vid -1 då y<-1.

För R% = 13 beräknas exp(R(K%)log(ABS(R(M%))).

För R% = 61 t o m 64 fås ungefär R(I%) = -2^65535 då argumentet är 0.

Detta bestämmer operationen 13 då R(M%) = 0.

Då ABS(y) >= 2^65534 blir kvadratroten ur y fel.

Precisionen N är lagrad på adress 36523 och 36524. A0 är lagrad på adress 36538 och 36539.
Om Du vill disassemblera rutinerna, ersätt rad 100 i PROG1 med 100 END före körning och disassemblera direkt efter PROG1. Använd sedan NYPREC för precisionsbestämningen. Assemblerprogrammet använder på adress 35896 och 35897 BOFA i nollställningsrutinerna 1 och 2. I övrigt är det helt oberoende av ABC80s operativsystem och BASIC-tolk och kan överflyttas till vilken Z80-dator som helst.

660 Egna assemblerprogram

Startadress för egna assemblerprogram är 36627. Du kan lägga programmen som hexkod i DATAsatser i programmet EGETPROG. Häri inkluderas mellanresultatareor, som kan reserveras med NOP-instruktioner. Använd HEXADR som hjälp att översätta decimala adresser till hexadecimala. På adress 36307 - 36522 finns endast JP- och CALL-instruktioner till andra adresser, i avsikt att förenkla handhavandet av rutinerna i BASIC-programmering. Av rad 130 och 140 i VÄRD framgår hur beräkningar kan utföras i maskinkod. Om man t ex vill utföra

R14 = R35 - R17

skriver man assemblerprogrammet

```
Mnem. instr.    Hexkod

LD DE,17        111100
CALL 35345      CD118A
LD DE,35        112300
CALL 35355      CD1B8A
LD DE,14        110E00
CALL 35365      CD258A
JP 36334        C3EE8D
```

De två första instruktionerna lägger A17 på adress 36525 och 36526. Nästa två lägger A35 på adress 36527 och 36528. Nästa två lägger A14 på adress 36529 och 36530. Av dessa pekare behöver man bara sätta dem för de variabler som berörs av en operation. Pekaren A(I%) (A14 i exemplet) förändras inte av rutinerna 1 - 72, så LD DE,i och CALL 35365 behöver inte göras på nytt för samma i. Ackumulatorn A och byte 32828 svarar mot S%+1 i rutin 3 - 8. Genom att använda byte 36628-36630, 36631 - 36633 osv för CALL- och JP-instruktioner till egna rutiner längre fram kan Du åberopa dessa med FNU% - FNR%, där R% = 108, 109 osv.

Vill Du relokera maskinkoden till något annat läge än 32768 - 36626, gör så här.

Mata in först 32K-versionen och sedan 16K-versionen. Ersätt rad 100 i PROG1 med 100 END. Kör inte NYPREC16. Jämför versionerna byte för byte. Varje skiljaktig byte ändras tillsammans med föregående byte, så att den adress de bildar ökas med skillnaden mellan det nya startläget och 32768. Antag att Du vill flytta koden till 35768 - 39626.

Kör BASIC-programmet

```
10 FOR I=32768 TO 36626
20 X=PEEK(I-1)+256*PEEK(I)+3000
30 IF PEEK(I) < PEEK(I + 16384)
   THEN POKE I-1,X,SWAP%(X)
40 NEXT I
50 FOR I=36626 TO 32768 STEP -1
60 POKE I+3000,PEEK(I)
70 NEXT I
```

Addera sedan 3000 till varje adress mellan 32768 och 36626 i NYPREC, VÄRD m fl.

BREV

Kalix 1982-10-17

Hej!

Tack för ABC-kassett 4, den var verkligen välkommen. Jag har redan börjat "smaka på" FORTH, det verkar intressant. På kassetten fanns ju också ytterligare en version av TV-editorn. När jag hade provat den nya versionen tänkte jag att jag kunde väl försöka att lägga in den funktion jag länge saknat, nämligen manuell kontroll vid replace. Ofta vill man ju inte byta ut alla "+" mot "-" t.ex. Med alla nya kommandon så minskar ju textbuffertens storlek katastrofalt, därför gjorde jag filen "TVMCHECK.MRG" som kan "mergas" till TVMAIN3 och då läggs manual check- funktionen in samtidigt som buffertstorleken ÖKAR med 89 bytes till 18037. Detta åstadkommes genom att ta bort en del "onödiga" rader i TVMAIN3 samt att göra om REM-satserna (inte själva texten, bara **** och liknande). Ytterligare utrymme kan sparas, men det är väl knappast tillrådligt att ta bort några REM-satser. En sådan hopslagen version, TVMAIN4 finns på denna diskett, provkör den gärna, den laddas direkt av den version av TV som jag också kopierat över. Observera att TVMCHECK.MRG inte kan skrivas ut av BASIC:en eftersom den innehåller en del "tomrader", om ni vill titta på den, använd editorn!! Manual-check funktionen fungerar så att vid kommandot ;R frågas "Manual check?", om man svarar något av JjYy så kommer editorn att leta upp första (eventuella) bytesobjekt och fråga Ok?, om positivt svar på denna fråga kommer byte att ske, annars söker editorn genast upp nästa objekt.

Jag bifogar också en kopia av framsidan på PPC Calculator Journal (PPC är en världsomspännande klubb för ägare av HP:s och TI:s programmerbara räknedosor). Kunde inte ABC-bladets framsida få en liknande utformning (åtminstone till hälften) så att man snabbt kunde gå tillbaka i sitt arkiv och hitta en viss artikel. I övrigt tycker jag att "bladet" är bra.

På den här disketten ligger också programmet "KONSTIGT", kör det och förundras; kanske kan någon av er förklara vad som sker och vad som orsakar effekten.

Happy Programming!
Börje Josefsson
Älvdalsvägen 20
952 00 Kalix

PROGRAMBITEN

Sverige har även index på framsidan
((red)cs)

ABC-klubben

Hej !

Kassettbandspelare till ABC 800 är det

ABC-klubben

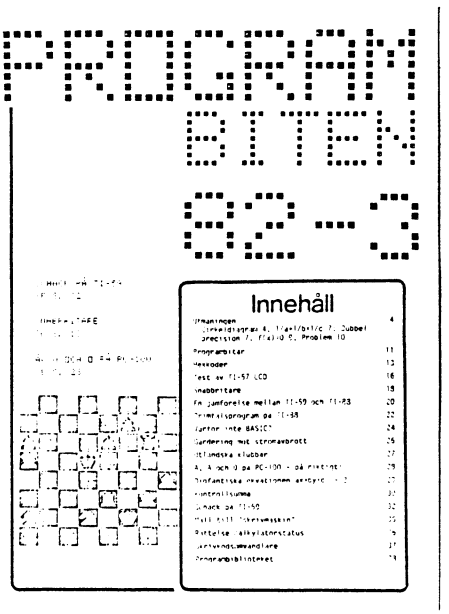
Hej !

Kassettbandspelare till ABC 800 är det kanske inte så många som använder. Tittar man i bruksanvisningen till 800an finner man att kassettbandspelaren ABC820 ser misstänkt lik ut bandspelaren till ABC 80. Det enda som ytligt sett verkar skilja de båda bandspelarna är anslutningen med sladd. Eftersom det på senaste prislistan från Luxor av någon anledning skiljer i pris på flera hundra kronor mellan de båda apparaterna så ställer man sig frågan om det går att använda 80ans bandspelare till 800an.
Svaret på den frågan är ja.
Det är hur enkelt som helst. Det enda man behöver göra är att skaffa sig en vanlig 5-polig DIN bandspelarsladd och en liten telefonplugg. Båda dessa detaljer följer med när man köper bandspelare för 80an. Sedan tar man och öppnar den ena av DIN-kontakterna och löder bort de två ledare som i 800ans bruksanvisning sid 21 har nummer 4 och 5 (dvs vänsterinner och högerinner på gammaldags fotbollsspråk) och som har hand om motorstyrningen. Dessa båda ledare löder man sedan fast i telefonpluggen. Och så är allt klart.

Hälsningar

Per Ahlin (1911)

Enligt uppgift är ytterlådan för 800-kassettspelaren samma som för 80:an
men
innehållet är helt nytt, elektronik bl.a.
(red)



KOLL A KAS 8

Program för
Matematikpaketet
INFO.BAS
MASK.BAS
RPN.BAC

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.
Ü 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.
(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 stacken. 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 villket 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.

RRIIINNGGGGG !!!!!!!!! RRRRIIIIIINNNNNGGGGGGG!!

HALLÅ !

Träffas Svenne ?

SVEENNEEE!

de `e telefon te dej ! ! !

Den här gången är det jag som ringer Svenne för att uppfylla det löfte jag gav förra gången vi talades vid. Jag ska alltså fortsätta med maskinspråk och den här gången tala om Z-80 processorn.

K-Å:
Hej nu ska jag fortsätta med maskinspråk. Är du redo?

Svenne:
Jag tror det, jag har tittat i Z-80 programming manual. Jag har hittat en förteckning över mnemonicks, som verkar möjlig att begripa, i varje fall om jag får lite hjälp.

K-Å:
Vi börjar med registren. Det finns 208 bits programmerbart minne i Z-80. Märk väl att jag sa bits inte bytes. Dessa 208 bits minne är organiserat i register om 8 eller 16 bitar. Vissa 16-bits register består av två 8 bitars register. Jag börjar med att presentera fyra register;
A, BC, DE och HL.
Dessa fyra register återfinner du i mängder av mnemonics.

A är accumulatorregister, ett 8 bits register som används vid aritmetiska och logiska operationer. De tre andra registren är 16-bits register uppbyggda av vardera två 8-bits register. Man kan alltså arbeta med BC som ett 16-bits register, men också med B eller C som 8-bits register. För de tre registren BC, DE och HL gäller att bokstaven till höger betecknar den lägre byten, en byte består av 8 bitar. Detta betyder att det finns minst två sätt att lagra exempelvis talet 65 i HL registret. Man kan använda instruktionen LD HL,NN eller om HL har utgångsvärdet 0 LD L,N. I båda fallen kommer det låga registret L att innehålla 65, vilket kommer att bestämma värdet på hela HL när H är 0. På motsvarande sätt kan man lagra det decimala värdet 512 i HL med instruktionen LD H,2 förutsatt att L är 0.

Svenne:
Finns det många LD instruktioner?

K-Å J:
Det kan man lungt svara ja på. Det finns så många att du måste lära dig att det finns ett visst system i dem. Man brukar inte säga LD utan load-instruktioner. Det finns till att börja med 8 och 16 bits load-instruktioner.

Svenne:
Ja jag förstår LD HL,NN är en 16 bits instruktion medan LD H,2 är en 8 bits.

K-Å J:
Ja, men dessutom skiljer man mellan load-instruktioner som kopierar innehållet i ett register till ett annat exempelvis:

LD H,L

som kopierar innehållet i L till H

(OBS att innehållet i L ej förändras).

Andra typer av LD-instruktioner kopierar innehållet i ett register till en plats i det yttre minnet eller vise versa.

Exempel

LD HL,(NN) kopiera innehållet i minnescell med adress NN till HL

LD A,(HL) kopiera innehållet i den minnescell som har adress lika med innehållet i HL till accumulatorregistret!

Här har du dessutom exempel på två olika sätt att adressera minnet; direkt '(NN)' och indirekt '(HL)'. En LD-instruktion innebär alltså en kopiering av ett innehåll från höger till vänster. Man brukar tala om source och destination försvenskat; källa och destination. Det generella formatet för en LD-instruktion blir då:

LD destination,källa

Svenne:
Kan både källan och destinationen vara antingen ett register eller en plats i minnet?

K-Å J:
Ja du kan tänka dig fyra kombinationer:

LD register,register
LD register, plats i minnet
LD plats i minnet, register
LD plats i minnet, plats i minnet

Svenne:
Finns det fler register än de du nämt?

K-Å J:
Ja det gör det. Det finns totalt 208 bitar programmerbart minne i Z-80 processorn. De register som jag hittills talat om innehåller totalt 56 bitar. Andra register är IX, IY, PC,SP och R. De register som betecknas med två bokstäver är 16-bitars register och R är alltså ett 8-bitars register.

Svenne:
Men det där blir ju bara 72 bitar till, hur hänger det ihop?

K-Å J:
Det hänger inte ihop, jag försöker presentera det här i lagom stora tuggor. Tills vidare räcker det med de register jag nämt. Jag ska berätta vad de gör, men jag tänker hoppa över IX och IY tills vidare.

Svenne:
Är de registren mindre viktiga?

K-Å J:
On the contrary! Men du kan skriva en massa rutiner utan dem. Jag kan kort nämna att de är speciellt lämpade för adressering vid uppbyggnad av tabeller.

Svenne:
Det låter avancerat jag står nog över ett tag.

K-Å J:
OK! Då kör vi. SP och PC registren är 16-bitars register medan R är ett 8-bitars register. SP är en förkortning av stack pointer stackpekare på svenska och PC är en förkortning av program counter, vilket på svenska kan översättas med programräknare. Stackpekaren håller reda på adressen till den externa stacken, stacken är en plats i minnet för lagring av registerinnehåll adresser m m enligt principen sist in först ut. Programräknaren håller reda på adressen till nästa maskininstruktion. Du kommer förmodligen inte behöva använda instruktioner där SP eller PC ingår förrän du lärt dig mer, det är däremot bra att känna till mer om dem för det hjälper dig att förstå hur Z-80 fungerar.

Svenne:
Det börjar bli rätt mycket nu, men ta R-registret också.

K-Å J:
R-registret är ganska speciellt. Det används bara i två stycken instruktioner. En vanlig användning tycks vara som slumptalsgenerator. Du ska få ett exempel på en rutin som ger slumptal mellan 0 och 255.

```
ORG -128
LD A,R
LD H,0
LD L,A
RET
```

Detta källkodprogram kan du assemblera med assemblerprogrammet på kassett nr 6. Det kan vara en bra övning att börja med.

Svenne:
OK! Det låter bra, det ska jag göra. Men kan du inte säga vad jag ska poka in för att köra rutinen nu med det samma.

K-Å J:
Det kan jag göra men glöm inte att assemblera för det!

10 POKE -128%,237%,95%,38%,0%,111%,201%

Svenne:
Finns det bara load-instruktioner?

K-Å J:
Förvisso icke. Det finns en rad andra instruktionstyper. Det finns hopp och anrop, logiska och aritmetiska m m. Du ska få en liten matris över de 64 första instruktionerna dvs de som har operationskod 0-63.

Svenne:
Hur många instruktioner finns det egentligen?

K-Å J:
Det finns 701 stycken.

Svenne:
Tur att jag inte visste det tidigare, då hade jag aldrig gett mig på det här.

Operationskoderna 0-63 decimalt

NOP	EXX AF	DJNZ e-2	JR e-2	JRNZ e-2	JRZ e-2	JRNC e-2	JRC e-2
LD BC,NN	ADD HL,BC	LD DE,NN	ADD HL,DE	LD HL,NN	ADD HL,HL	LD SP,NN	ADD HL,SP
LD (BC),A	LD A,(BC)	LD (DE),A	LD A,(DE)	LD (NN),HL	LD HL,(NN)	LD (NN),A	LD A,(NN)
INC BC	DEC BC	INC DE	DEC DE	INC HL	DEC HL	INC SP	DEC SP
INC B	INC C	INC D	DEC E	INC H	INC L	INC (HL)	INC A
DEC B	DEC C	DEC D	DEC E	DEC H	DEC L	DEC (HL)	DEC A
LD B,N	LD C,N	LD D,N	LD E,N	LD H,N	LD L,N	LD (HL),N	LD A,N
RLCA	RRCA	RLA	RRA	DAA	CPL	SCF	CCF

8-Bits load-instruktioner
operationskoderna 64-127 decimalt

LD B,B	LD C,B	LD D,B	LD E,B	LD H,B	LD L,B	LD (HL),B	LD A,B
LD B,C	LD C,C	LD D,C	LD E,C	LD H,C	LD L,C	LD (HL),C	LD A,C
LD B,D	LD C,D	LD D,D	LD E,D	LD H,D	LD L,D	LD (HL),D	LD A,D
LD B,E	LD C,E	LD D,E	LD E,E	LD H,E	LD L,E	LD (HL),E	LD A,E
LD B,H	LD C,H	LD D,H	LD E,H	LD H,H	LD L,H	LD (HL),H	LD A,H
LD B,L	LD C,L	LD D,L	LD E,L	LD H,L	LD L,L	LD (HL),L	LD A,L
LD B,(HL)	LD C,(HL)	LD D,(HL)	LD E,(HL)	LD H,(HL)	LD L,(HL)	HALT	LD A,(HL)
LD B,A	LD C,A	LD D,A	LD E,A	LD H,A	LD L,A	LD (HL),A	LD A,A

K-Å J:
Jag kan trösta dig med att det finns grupper av instruktioner av samma typ där variationen består i att man permuterar register som källa och destination.

Svenne:
Kan det också visas i en matris?

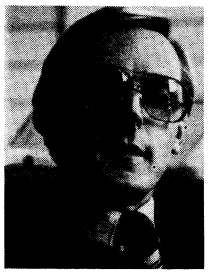
K-Å J:
Rätt så. Jag skickar några matrisuppställningar över mnemoniks till maskininstruktioner du tittar på dom och så hör vi av varandra. Blir det bra så?

Svenne:
Sure!

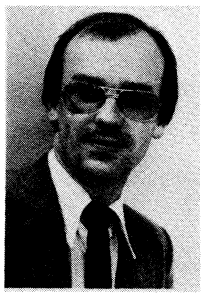
Kjell-Åke Johansson

Luxor

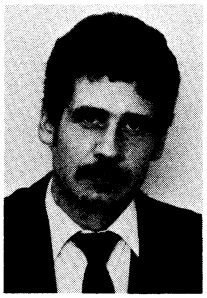
NU SKA VI BLI STARKA I HELA SKANDINAVIEN



- Det anser Luxors Datorers nye marknadschef Staffan Nordstrand.



Ny säljledare är Christer Vollmer



Ny säljkonsulent Robert Rajfors

SÄLJES

Microline 80
Endast demonstrationskörd
3500:- inkl.moms
0304-63190 Efter kl. 18

KÖPES

ABC80 med bandspelare köpes kontant.

Klas Westman

0764-62644 Efter kl. 19

KÖPES

ABC80 även utan bildskärm.

08- 755 77 77

fråga efter Willard.

KÖPES

ABC80 samt Flexskiveenhet.

Märk brevet med 'CS-GT'

(för att komma rätt).

ABC-klubben

Vidängsvägen 1

161 33 Bromma

Att Reparera Skivor Med Programmet "Skivläs"

Av Lars Erik Sörensen

1. Skriv: RUN SD10 <RET>

2. Ange vilken drive du vill läsa: 0 eller 1

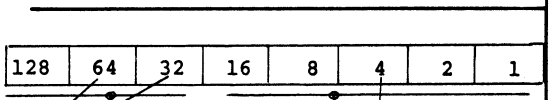
3. Ange vilken rekord du vill läsa: 0 till 1231 <RET>

Antag att du har fått fel på en skiva. Det kan vara fel i biblioteket, felaktigt rekordformat, checksummafel vid läsning, felaktigt tecken inläst eller diverse andra felaktigheter. Låt oss till en början antaga att det blivit fel i biblioteket.

Ange 8 på punkt 3. Finns inte det sökta programnamnet där, stega vidare med <CTRL S> till rekord 9 eller till dess du hittar programnamnet. Finns det inte på rekord's 8 - 15 så se på rekord's 16 - 23 som skall vara identiska med 8 - 15. Finns programnamnet i båda LIB-filerna skall du titta efter att det står på samma LIB-plats. Flytta markören med <CTRL F> eller --> eller <CTRL N>, enligt kommandotabellen, till fjärde byten före programnamnet. Kontrollera med <CTRL U> att de fyra första byten är lika i båda LIB'en.

Tryck därefter <CTRL B> för att få reda på var programmets HEADER ligger. Tryck <CTRL E> och ange detta rekordnummer samt <RET>. Finns Headern intakt tittar du i början med <CTRL U>. Byte 1 visar programnummret, som har samband med programmets plats i LIB'et. Bytes 2 och 3 är sektorns ordningsnummer där Headern börjar på 0. Byte 4 skall alltid vara 255. Sedan börjar pekarna till programdelen eller delarna. Var pekare består av 2 byte. Finns inte fler pekare avslutas med minst 2 bytes 255D. Första byten pekar på det spår den aktuella sektorn ligger. Andra byten talar om var på spåret första sektorn ligger och hur många sektorer delen innehåller. OBS!-ervera att var del inte kan innehålla mer än 32 sektorer inklusive eventuell Header.

Byte 2 i pekaren ser ut så här:




```
0  sektorer in på spåret =   0
1  sektorer in på spåret =  32
2  sektorer in på spåret =  64
3  sektorer in på spåret =  96
4  sektorer in på spåret = 128
5  sektorer in på spåret = 160
6  sektorer in på spåret = 192
7  sektorer in på spåret = 224
```

Tabell 1

Det kan t.ex stå 109 100. 109 är spår 109 och 100 = 96 + 4

vilket betyder: 3 sektorer in på spår 109 och 5 sektorer långt. (0 räknas som 1:a sektor). 109 * 8 + 3 = 875 uttytt:spårnr * (sektorer/spår) + sektorer in på spåret = rekordnummer.

Saknas dessa bytes eller du har anledning att misstänka att de är fel, kan du trycka på <CTRL P>. SD10 söker då efter alla programdelar med samma ascii-kod med början på rekord 23 och slut på rekord 1208. Headern utmärkes med * . Skriv nu alla rekordnummer under varandra på ett papper. Tror du inte att det finns fler sektorer i programmet stoppar du sökningen med (CTRL <) .I OBS! att SD10 nu är inställd på sist sökta sektor oberoende på vilken sektor som visas. När du skrivit ned de sökta sektornummren kan du trycka <CTRL E> och ange sektorns på skärmen (Headerns) rekordnummer. Denna aktiveras då åter. De rekordnummer som kommer i följd avdelar du i grupper med ett streck. Genom att dividera gruppens första tal med 8 får du spåret första sektorn i gruppen ligger på. Börja med Headern. Skriv upp talet, ty det är första ledet i pekaren i Headern. Om spårnummer * 8 minus rekordnummer ger en skillnad, talar denna om hur långt in på spåret sektorn ligger. Genom att gå in i TABELL 1 hittar du bitvikten för sektorplaceringen i pekarens andra byte. Genom att räkna antalet samhörande rekordnummer får du veta hur många sektorer gruppen innehåller. Lägg detta tal minus 1 till talet för sektorplaceringen och du har fått andra byten.

Det finns ett par fällor. Gamla programrester kan ha samma programnummer. Se för säkerhets skull efter om radnummer stämmer. En senare tillfogad sektor kan komma direkt efter en äldre grupp av rekords, den har i så fall ett eget pekarpar. En grupp av rekords kan också komma före programmets Header och sist, E = ASCII 64, så tomma sektorer med E inskrivet räknas med om programnummer är 64.

Har du gjort rätt nu ska allt fungera. Är man noga, går man in i bitkartan, rekord 6, och kollar att de sektorer programmet har belagt är spärrade. I annat fall tror ABC80, att det är fritt fram att skriva där igen. Är man inte noga skrivskyddar man eller kopierar över till ny skiva. Man kan också gå en medelväg och fylla de aktuella spåren i bitkartan med 255D för att vara på den säkra sidan.

Hur du lagar bitkartan kan du läsa om i artikeln "Diska Rent", ABC-bladet 1. 1982.

```
++++++++++  S D 1 0  ++++++++++

****** K O M M A N D O N ******

Allt är Ctrl-tecken
Du styr cursorn med pilarna

V - Ger denna lista (Visa)
Q - Slut (Quite)  JÄMFÖR C
W - 256 bytes på printer (Write)
E - Byt rekord-nummer (Exchange)
R - Cursor upp (40 tecken/steg)
T - Cursor home (Top)
Y - Repeat-tangent
U - Visar 10 ascii-tecken
I - V-PIL "FRAMSTEG-tangenten
O - Flytta rekord
P - Letar upp programsnuttar
Å -
^ -
A - Rekord -1
S - Rekord +1
D -
F - Cursor ner (40 tecken/steg)
G - Byt drive (0 eller 1)
H - V-PIL "BACKSTEG-tangenten
K - Räknar ut byte 1 & 2 i Lib-
    titel
Z - Skriv rekord
X - Skriva in Ctrl-tecken <31
    och >127
C - NÖDSTOPP  JÄMFÖR Q
B - Auto header-sökning
N - 5 Steg framåt
J - 5 Steg bakåt
```

OBSERVERA !

Programmet SD10 vilket finns på kassett fem är gjort för 77 spårs flexskivor vilket kanske förundrar en del medlemmar och kan förorsaka bryderi när programmet vill läsa längre än som finns sektorer på den egna utrustningen. Man har här:
// * 8 * 2 = 1232 sektorer i dubbel täthet. Således 0 till 1231.

Många andra medlemmar har något av nedan sektorantal:

Vid enkel täthet:
40 spår * 8 sektorer = 320 sektorer.
Vid dubbel täthet:
40 spår * 8 sektorer * 2 spår/varv = 640 sektorer.
Vid dubbel täthet och dubbelsidig:
40 spår * 8 sektorer * 2 spår/varv * 2 sidor = 1280 sektorer.

OBS ! Det blir som datorn läser det:
319, 639 och 1279 sektorer.

Kolla följande rader:

```
 290 A%=VAL(A$) : IF A%<0% OR A%>
     1231% 220
 610 IF K%=19% IF A%<1231% A%=A%+
     1% : GOTO 320 : REM S
 900 A%=VAL(A$) : IF A%<0% OR A%>
     1231% 870
2530 IF B2%=1231% ; 'Ok!' : GOTO
     400
```

och ändra dem till ert eget sektorantal.

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.

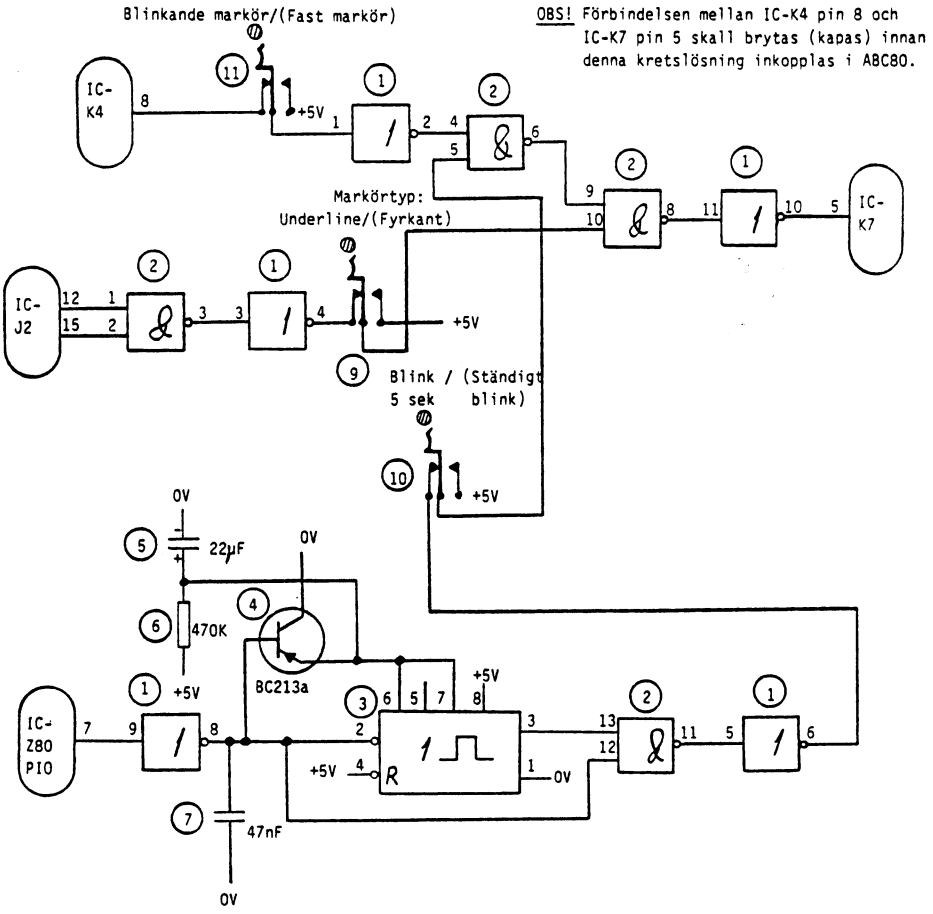
Ledning:

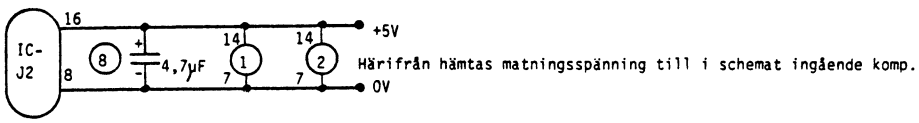
De ovala symbolerna i kretsschemat visar externa anslutningar, dvs. trådar som skall anslutas till ABC80:s stora kretskort. I de ovala symbolerna står koordinaten till aktuell IC-krets samt bennummer på själva anslutningen. Ett undantag från detta är den ovala symbolen med följande text "IC-Z80 PIO".
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.





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 adresserna 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 Monitormode.

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 stor som 18 sektorer, hur detta kommer sig 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 Checksumma 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 ??

GRAFEDIT

Instruktioner till GRAFEDIT Ver 3.21

Av Benny Löfgren

Cursorn:

Det finns två slags markörer, en som ser ut så här:  och en annan som ser ut så här: ■ Båda två flimrar lätt. Korset visas när man är i flyttmod, punkten när man är i ritmod. När programmet startas upp är man i flyttmod. Man flyttar cursorn med följande tangenter:

Uppåt: W
Vänster & höger: A S
Nedåt: Z

Håller man tangenten nertryckt flyttar sig cursorn fortare och fortare. Man kan även, om man håller CTRL-tangenten nertryckt samtidigt, få snabbförflyttning till kanten på skärmen.

Om man i stället flyttar cursorn med:
Å kommer den att flyttas till den
Ö Ä närmaste punkt som har motsatt
- färg som den som cursorn står på.

Det finns alltid två markörer på skärmen. Den ena är den tidigare beskrivna. Den andra, som är en flimrande punkt, visar var markören sist befann sig. Den flyttar sig till nuvarande positionen när man utför något av kommandona Frame, Box, Line och Mark (Jag kommer till dom sedan).

Den översta raden på skärmen visar båda markörernas position. Först visas Y- och X-koordinaterna visas koordinaterna för den andra markören, som Y' och X'. Sist visas vilken av moderna Set, Clear eller Inverse som programmet befinner sig i. (Flyttmod och Ritmod visas ju av cursorns utseende!) Den här raden försvinner om någon av markörerna befinner sig där, men kommer tillbaka så fort det blir tomt. OBS! Raden förstör inget där den finns, allt som är ritat där kommer tillbaks när man flyttar upp markören.

Kommandon:

Line: (L) Linjedragning. En linje dras mellan den förra punkten och huvudmarkören. Den förra punkten flyttas till huvud markörens position.

Frame: (F) Ramritning. En ram ritas med ett hörn i huvudmarkörens position och ett i den förra punktens.

Box: (B) Fylld rektangel. I övrigt som Frame.

Paint: (P) Fyll yta. Ett avgränsat område fylls.
OBS att man kan vara tvungen att fylla flera gånger för att få helt täckt.

OBS! Dessa kommandon är beroende av i vilken ritmod programmet befinner sig.

Set=rita med vitt.
Clear=rita med svart (radera).
Inverse=vitt blir svart och svart blir vitt.
(Fungerar inte med Paint, Inverse omvandlas till Set.)

Kommandon, forts:

Home: (H) Markören hoppar till mitten på skärmen.

Jump: (J) Markören hoppar till förra punkten.

Mark: (M) Den förra punkten flyttas till huvudmarkörens plats.

Save: (Ctrl-<>) Spara bilden. Filnamn frågas. Om ingen extension anges blir den '.PIC'.

Load: (Ctrl-E) Ladda in bild. OBS att den gamla bilden finns kvar (MERGE). Se i övrigt Save.

Save och Load kan avbrytas genom att i stället för filnamn ge ett '<'-tecken. Om Save avbryts med texten 'Filen finns!' betyder det att det redan finns en fil med det namnet. Om man ändå vill spara är det bara att trycka RETURN.

Kommandon, forts:

Set: (->) Allting ritas hädanefter med vitt.

Clear: (<-) Allting raderas hädanefter.

Inverse: (I) Allting inverteras efter det här kommandot.

Ritmod: <SPACE> Cursorn blir en liten punkt. När man nu flyttar markören sätter den ut punkter i enlighet med ovanstående.

Flyttmod: <RETURN> Cursorn blir åter ett kors.

Radera: (Ctrl-R) Rensa skärmen, flytta cursorn till mitten av skärmen.

För snabbinstruktioner tryck '!'.

```
  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
```

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 like 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, som 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
        END
```

4 viktiga

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 floppyicter. 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 nú. 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 ----------

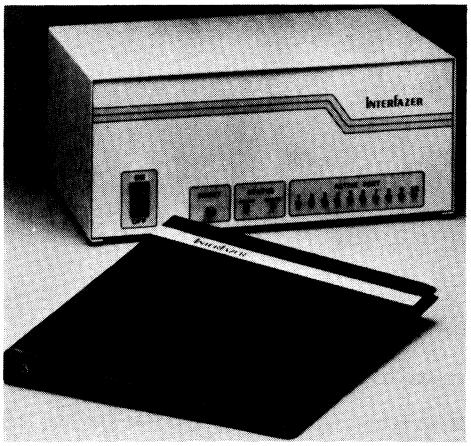
KOM är ett telemötessystem på QZ där deltagarna utväxlar brev och meddelanden om de mest skilda saker. Där träffar man folk från FOA, SCB, KTH, QZ m fl andra företag och instutitioner och diskuterar datateknik - alla nivåer, politik, löser gåtor, annonserar prylar och lägenheter m m. Detta sker dock 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.

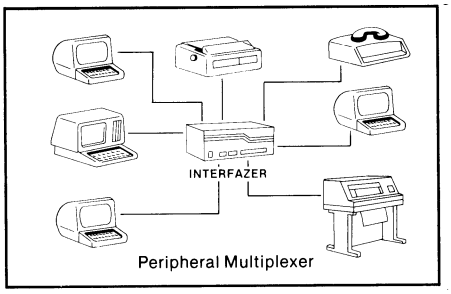
ABC-Klubbens QZ-projekt är som ni kanske tidigare läst i bladet en försöksverksamhet, för dataamatörer öppen endast för ABC-Klubbens medlemmar. Medlemsskapet i ABC-Klubben ger alltså en unik möjlighet att till starkt reducerad taxa pröva på "riktiga grejor".

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 erhålles i samband med ansvarsförbindelse / manual. (RED)





MICROFAZER EN NY SNABB PRINTER-BUFFER

Microfazer kan ta emot data från datorn med en hastighet av upp till 4000 tecken per sekund, lagra detta i minnet för att sedan sända det vidare till skrivare. Denna buffring av data medför att datorns höga hastighet kan bibehållas trots skrivarens låga mekaniska hastighet. Detta gör, tillsammans med enkel installation och handhavande, att Microfazer är användbar i alla datorsystem.

Valfritt antal kopior av senast utskriven information fås genom att trycka in 'Copy' knappen erforderligt antal gånger.

Microfazer är en kompakt låda med måtten 7 * 18 * 3 cm i vilken kretskort med alla nödvändiga kontrollkretsar och upp till 8 plug-in 64K RAM minneskapslar. Detta möjliggör att användaren själv kan utöka minnet från 8K (en kapsel) till 64K (8 kapslar) beroende på tillämpning.

Sedan Microfazer är installerad så sköter den sig själv. Det behövs inga startrutiner eller andra drivprogram utan allt sker automatiskt via datorn.

Microfazer kan pluggas in direkt på de flesta på marknaden förekommande skrivare. Tag bort kabeln som går till skrivaren, montera Microfazer på skrivare och anslut kabeln till Microfazer.

Minnesstorlekar:
Parallell / Parallell 8K - 512K

Seriell / Parallell, Seriell / Seriell,
Parallell / Seriell 8K - 64K.

Samtliga modeller levereras med ett års garanti.

Pris från 1495:- exkl moms
Marknadsförs av Dataimport
Box 256
131 02 Nacka

Telefon 08-716 14 35

InterFazer - Fleranvändarsystem för upp till åtta arbetsplatser

InterFazer är en intelligent kontrollenhet för upp till åtta arbetsplatser (datorer, terminaler, ordbehandlingssystem) som fördelar deras utmatning på en eller två utenheter (skrivare, etc). Inmatning och utmatning kan vara i valfri kombination seriell och/eller parallell. Den seriella överföringshastigheten kan väljas från 75 - 19.2K. Buffertminnet är utbyggbart i 16K block till 128K RAM.

InterFazer kontrolleras av en 8085 mikroprocessor som tillsammans med ett prioritetssystem övervakar de inkommande signalerna från de 10 olika in/utgångarna. InterFazer kan via antingen seriell (RS232C) eller parallell (centronics kompatibel) överföring ta emot data från de flesta på marknaden förekommande datorer (bl.a. ABC80, ABC800, Apple II,Apple III, mfl.), terminaler eller ordbehandlingssystem och buffra data till de flesta skrivare.

InterFazer kan bl.a. användas som:

1. skrivarkontrollenhet i ett fleranvändarsystem
2. gränssnitt för olika överföringssätt
3. expansionsenhet för datorns in- och utmatning
4. kringutrustningsmultiplexer
5. omvandlare av olika överföringshastigheter
6. en buffer till övrig kringutrustning

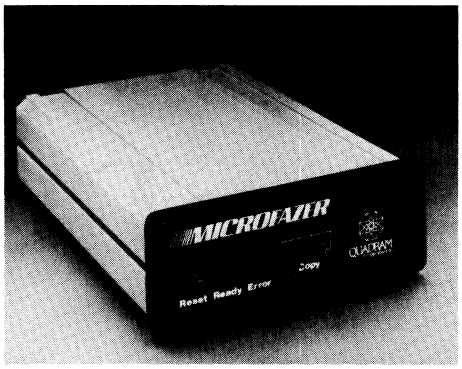
På frontpanelen finns strömbrytare, 'Reset' samt lysdioder som indikerar status, linjeaktivitet och felkoder.

InterFazer är konstruerad för att minska datorns väntetid genom att öka dess operativa produktivitet utan att samtidigt öka den operativa komplexiteten. InterFazer kräver inget särskilt program som liknande enheter ofta gör. Det är dock möjligt att programmera den för särskilda tillämpningar, men dess inbyggda prioritetssystem fungerar utan yttre inblandning för att maximalt utnyttja Er kringutrustning. Dess stora lagringskapacitet (upp till 128K eller ca. 64 sidor text) betyder att Er dyrbara datortid ej behöver upptas av utskrifter eller liknande uppgifter.

Kompletta system från 7775:- exkl moms.

Ett års garanti.

Marknadsförs av Dataimport
Box 256
131 02 Nacka



Liber

Liber har gett ut en programkatalog för ABC80/ABC800

Liber Distribution
Order Utbildning,
162 89 Stockholm

08-739 91 00

De har även givit ut "Data-Litteratur från Liber"

Liber Grafiska AB
Marknadsavdelningen - Förlag
08-739 90 00
162 89 Stockholm

BASLAGER

Ett nytt lagerprogram för de tre mikrodatorerna LUXOR ABC80, ABC800 och FACIT DTC har kommit från
MAGINIUS ELEKTRONIK i PÅARP.
BASLAGER är uppbyggt på PDATAS generella databas PENTAPUS, som också ligger till grund för det populära programmet BASREGISTER.

Några viktiga punkter om BASLAGER:

* Lätt att lära och använda.
* Alla rutiner är snabba och effektiva.
* Hög datasäkerhet.
* 1100 artiklar per skiva (40 spår dubbel densitet).
* Snygg och överskådlig skärmpresentation.r

* 20 uppgifter per artikel.
* Inmatningskontroll och formering.
* Beräkning av vägt nettopris.
* Packning av numeriska data.
* Automatisk insortering.
* Skyddade fält.
* Reservering och kassering.
* Periodstatistik.
* Snabbsökning.
* 8 överskådliga rapportlistor.
* Olika selekteringsmöjligheter vid listning.

Snabbkopiering av dataskivan.
* Snabbformatering av ny skiva.
* Dataskiva för övning medföljer.

Ytterligare information om BASLAGER kan erhållas från:
MAGINUS ELEKTRONIK
Lundsgatan 13
260 33 PÅARP

Telefon 042-22 73 20

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. tab !

CTRL-S Flytta markören en position åt höger. tab !

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-Push Skjuter fram resten av texten på bildskärmen ett steg.

CTRL-Draw 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.

ABC KLUBBEN

Publikationer m.m.

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

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

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

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

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

ABC-klubben rabatterar för nya medlemmar vid köp av båda dessa samlingsnummer och tar då 210 Skr. Av praktiska skäl kan inte någon uppdelning göras av dessa i mindre delar. Om Du är intresserad av någon av ABC-klubbens publikationer får Du dem enklast genom att sätta in beloppet på ABC-klubbens postgirokonto 15 33 36 - 3 och ange vad Du vill ha samt **en tydlig och läslig leveransadress.**

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

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

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

ABC-klubben
Vidängsvägen 1
161 33 Bromma

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

Postgiro 15 33 36 - 3