A B C-bladet Samlingsnummer
1981 : 1 sidan 8

INKOPPLING AV MODEM OCH PRINTER

Den som vill slippa att koppla om sladdar varje gång man växlar mellan att ha modem eller printer ansluten till V24-kontakten på ABC-80 kan ha nytta av att göra liten kopplingsbox enligt vidstående schema. Med denna box kan man ha både printer och modem permanent anslutna. Beroende på hur modemet är kopplat internt behöver inte alla ledningar anslutas. Följande ledningar är dock **nödvändiga** att ansluta för att modemkommunikationen skall fungera: **TXD, RXD, CTS, DCD och jord.** Ledning 105 är ofta kopplat i läge **TILL** internt i modemet, i så fall behöver **RTS** ej anslutas. Likaså är ibland även 108 satt **TILL,** då behöver ej **DTR** anslutas. I detta fall kan man även utesluta switch 1. (Om man vill använda modemet i läge för automatsvar bör **DTR** vara ansluten med en sådan koppling att man kan styra tillslag av spänningen på 108 med reläet för bandspelarens motorstyrning.)

För inkoppling av printern räcker det ofta med att ansluta tre ledningar: printerns **RXD,** normalt stift 3, jord stift 7 samt stift 20 som är **DTR.** På Diabloprintern finns en option där man kan ersätta DTR med en speciell Printer Ready Signal. Det går till så att man flyttar den svarta tråden från pin 3 till pin 2 i den 12-poliga kontakten på HPRO4-kortet. Man får då samma signal som Anadexprintern ger: utsändning av tecken från ABC-80 stoppar när bufferten är nästan full och slipper på det viset overflow när man skriver ut med 1200 baud. Med denna inkoppling kan printern användas vid utskrift i lokal mode, switch 3 skall då stå i läge **TXD** och switch 2 i läge **CTS.** Om under utskrift switch 2 bryter stoppar utskriften så fort bufferten tömts, när man slår till switchen igen fortsätter Diablo att skriva nästa tecken även om det råkar vara mitt ett ord. På Annadex tappar man slutet av raden om uppehållet överstiger 10 sek. När man använder denna inkoppling skall man inte använda handskakning med ETX - ACK.

När man kör terminal skall switch 2 stå i neutralläge, med switch 3 väljer man läge **RXD** om man vill skriva ut inkommande signal vid t ex körning mot DEC-10 eller annan stordator med full duplex. I läge TXD får man utskrift av det som från ABC-80 sändes till den utrustning man kommunicerar med. Genom att ställa switch 3 i neutralläge stoppar man onödig utskrift på printern och får den enbart på skärmen, praktiskt om man vill spara papper.

NÄR JAG FICK MIN PROMPROGRAMMERARE

Av Magnus Lundberg

Veckan före jul fick jag hem PROM-PROGRAMMERAREN från METRIC. Det är ett kretskort i europaformat som ansluts till ABC-bussen. På kortet finns 2 st "brännplatser" där man kan programmera 2716 eller 2708 (även 2758). Det finns också 2 "körplatser" med byglingsbara adresser, så det går att prova direkt om ett program fungerar. Nu var min avsikt inte att i första hand skriva ABC-program i PROM utan jag ville se om man kunde tillverka PROM för att placera i automatiska slipmaskiner och andra NS-maskiner. Det var fråga om att lagra 6k byte text (bl. a. ledtexter på manöverpanelen).

PROM-programmering går (i alla fall på ABC-80) till så att ett stycke av RAM-minnet kopieras över till PROMmet. Detta sköts via någon form av kommunikation mellan datorn och PROM-PROG-kortet. Vid leveransen ingick en skiva med hjälpprogram för ändamålet.

Mitt första praktiska problem var hur jag skulle få in mina 6 kilo text i RAM-minnet. Till saken hör också att det inte var renodlad text, vissa stycken består av ej skrivbara ASCII-tecken. Men jag har ju Scandia Metric's nya assembler.

Eftersom editorn bara klarar 40 teckens radlängd använder jag fortfarande den gamla editorn, som är modifierad så att den passar mig bättre. Efter en stunds skrivande hade jag något som liknade ett assemblerprogram. Med nästan bara DEFM och DEFB etc. Jag editerade en liten bit i taget och sedan knöt jag med ett speciellt program ihop alla småsnuttar till en slutlig fil som går att assemblera.

Vid närmare betraktande av assemblatorns output upptäckte jag att de längsta raderna inte har blivit fullständigt kodade i ITH-filen. De var stympade efter 32 bytes. I Zilogs bok står att DEFM kan ta texter upp till 64 bytes. I ABC-80s version av assemblern har man tydligen halverat detta. Nå, jag skrev om alltihop och nu såg det bra ut. Det blev en rejäl ITH-fil av alltihop.

Jag stack mig lite på byglingspinnarna när jag skulle stoppa i mitt nya PROM i sockeln. Kortet verkar ganska ömtåligt. Det är alldeles oskyddat där det ligger på skrivbordet.

Sedan flyttade jag upp BOFA'n och beredde plats där jag tänkte "mellanlagra" min text. Jag har ju fullt utbyggt RAM-minne så det var inte alls trångt, men jag föredrog att ha det verksamma programmet "upptill" och texten "nedtill".

Laddningen av ITH-filen gick fint med det bifogade hjälpprogrammet. Jag såg själv till att hela det lediga minnet först fylldes med FF (hex) för att inget oönskat skräp skulle finnas med på de ställen jag INTE tänkte bränna i PROMmet. Bitvis har jag luckor i min text (DEFS i assembler).

Nästa hjälpprogram heter PROG. Här kan jag dels

a) kontrollera att PROM är tomt
b) jämföra PROM med minne
c) kopiera PROM till minne
d) kopiera minne till PROM

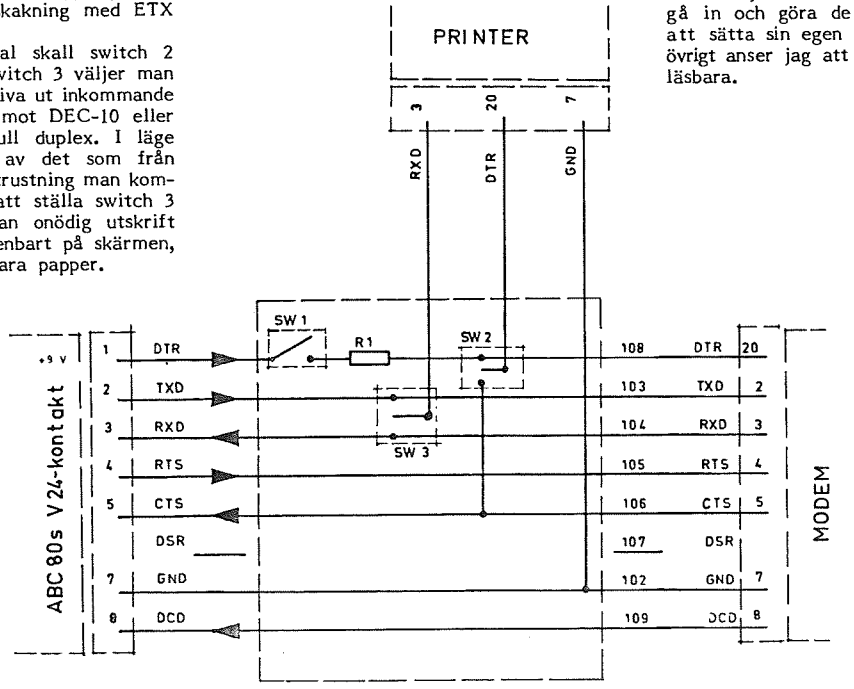
Det tog 1 min 40 sek att fylla en 2716 (2k bytes).

Kanske är jag ovanligt fumlig eller ovan, men jag tyckte det var trångt och lite besvärligt att ta bort PROMmet och sätta dit nästa. Men visst gick det.

Strax hade jag mina tre PROMmar klara. De har senare visat sig vara helt riktiga och användbara i slipmaskinen. Eftersom jag hoppas att få fler uppdrag att bränna PROMmar ville jag förstås hitta på ett mer effektivt sätt att lagra min kod än i ITH-format. Detta föranledde mig att skriva ihop ett eget fungerande filhanteringssystem, med bitkarta och directory. Ett eget bibliotek helt enkelt. Det går nämligen att ändra en pekare till DOSBUF 0 så att man får in sektorn direkt där man vill ha den i RAM-minnet efter endast ett enda call till DOS'et.

Denna teknik innebär en fantastiskt snabb laddning av koden samt en skivminnessnål lagringsmetod. Koden ligger i oförvanskat skick på skivan och fyller hela sektorerna (256 bytes). Andra vägen, att tömma RAM-minne till skivan, går till på liknande sätt och är lika snabbt.

Var och en har naturligtvis sin egen åsikt om hur ett program skall fungera och för en gångs skull levereras hjälpprogrammen till PROM-PROG av Scandia Metric i helt oskyddat skick. Man kan alltså själv gå in och göra de ändringar man vill för att sätta sin egen prägel på systemet. För övrigt anser jag att alla program skall vara läsbara.



YTTERLIGARE ETT BREV FRÅN TOMAS DYFVERMAN

Ledaren i ABC-bladet 4 antydde en fråga om inte bladet kanske skulle övergå till att drivas konventionellt med avlönad personal etc. Varför inte? Den stora medlemskaran kan knappast vänta sig att ett litet gäng entusiaster skall vara uthålliga nog att gratis ge service även när den första upphetsningen lagt sig.

Oavsett ovanstående, tidningen är värd mycket mer för varje ABC-80-ägare än vad den kostar. Speciellt lösnummerpriset är löjligt lågt. Jag föreslår att man höjer lösnummerpriset till 25 kr, så att en årgång alltså kostar något mer än ett medlemskap. Detta är ett vanligt förhållande utomlands när specialistsällskap ger ut en tidning. Även då man kan prenumerera är priset högre än för det medlemskap där prenumeration ingår. Ett utmärkt grepp, anser jag. Då har man också råd att betala de provisioner som handlare och prenumerationsförmedlare skall ha.

ABC-bladet innehåller relativt sett mer avancerade saker än enklare matnyttigt stoff för oss på Basic-nivå (som bör vara ganska många). Detta avspeglar förmodligen ingen policy från redaktionen, utan beror väl helt enkelt på att det främst är "djupt databitna" som sänder in bidrag... Inget fel i det. Jag skulle inte vilja ha bort något av det som tryckts hittills.

Jag hoppas bara att alla de avancerade PEEKar som med POKErface utdelas i spalterna inte avskräcker folk med "enkla" erfarenheter och vardagspraktika ting att sända in bidrag.

Här ett eget tips som är så enkelt att flera assemblerare riskerar att skratta ihjäl sig ...

Program som inte skall köras

Checklistor brukar användas, privat och i företag, när en mängd olika åtgärder skall vidtas och det är lätt att glömma bort något. Speciellt när det kan få dystra säkerhetsmässiga, funktionella, ekonomiska eller andra konsekvenser om något glöms bort. Det kan gälla uppstartningen av en maskin, lastningen av en servicebil eller vad som helst.

Jamen, skriv en lista på vanligt sätt då, för hand eller på maskin. Det fungerar bra, första gången. Det är bara det att förhållanden ändras, nya moment tillkommer, andra försvinner, oklara saker skall omformuleras etc. Det orkar man normalt inte med att hålla aktuellt och omskrivet någon längre tid. Därför kan det vara en vits med att skriva in sådana här listor i datorn så att de blir lätta att uppdatera. Och varje gång en kopia av listan behövs (för att kunna pricka av och markera fel etc), så skriver datorn billigt och snabbt ut ett nytt, aktuellt, original. Och det är här mitt tips kommer in. Håll i er, assemblerare! Man behöver nämligen inte göra något program för detta, utan man låter programmet vara checklistan. Det enda man får "offra" för denna enkelhet är att man måste acceptera tre onödiga tecken i form av REM på varje rad. Och det kan man väl, om det bara gäller ett internt papper.

Att man får radnumren i vänsterkanten är ju bara en fördel, då har man något att referera till vid diskussioner per telefon etc, ändringsförslag, inplacering av nya moment etc.

Man programmerar och skriver in data i samma nedslag, så att säga. Här är början på ett exempel:

```
10 REM CHECKLISTA LASTNING SERVICE-
   BIL
20 REM ***Bakre lastutrymmet***
30 REM Oscilloscope
40 REM nätsladd      till detta
50 REM probe till detta
60 REM brux
70 REM hopfällbart rullbord
80 REM skarvsladd 10 m jordad

etc etc
```

När man sen vill ha det på papper, skriver man bara LIST PR:, och så får man programmet=checklistan utskriven. Utan behov av eller kunskaper om printsatser, sorteringsprogram, filöppning ...

Man drar ju helt enkelt nytta av ABC-80's förnämliga system för automatisk insortering av rader, listkommandon, radredigering etc.

Så man kan ha stor nytta av ett program som man aldrig ger RUN-kommando, dvs ett program som aldrig körs!

BROMSAR DU DIN ABC-80?

Hör du till dem som tycker att det är höjden av finess att skriva X%=PEEK(-128) istället för X%=PEEK(65408%)? Satserna uträttar samma sak, men den första kommer obönhörligen att bromsa exekveringen till halvfart.

Ta en titt i tabellen nedan. Där ser du åtta sätt att uttrycka samma sats i ABC-80 Basic och hur mycket ett litet procentecken betyder. Alla vet förmodligen, att ABC-80 hinner med fler heltal än flyttal på en kvart - men hur stor är skillnaden? Jämför rad 10 med rad 30, eller rent av rad 10 med rad 80.

MINUS TAR PLATS

Ibland kan det vara lättare att komma ihåg minnesadresser i form av negativa tal, t ex -128 istället för 65408. Men det som är bra för ditt minne uppskattas föga av datorn - hastigheten minskar och programmet tar större plats. Det ser du på raderna 10 och 20 m fl.

Det var Basic, det. Har du riktigt ont om plats och riktigt bråttom ska du självfallet skriva programmet i assembler. Ovanstående sats kan skrivas LD A,(65408), som fyller ackumulatorn med innehållet i adressen på 5 mikrosekunder. Och det programmet tar bara tre bytes.

Rad	Sats	Tids-åtgång	Plats-behov
10	X%=PEEK(65408%)	460us	16 bytes
20	X%=PEEK(-128%)	520us	17 bytes
30	X=PEEK(65408%)	660us	17 bytes
40	X=PEEK(-128%)	720us	18 bytes
50	X%=PEEK(-128)	970us	21 bytes
60	X%=PEEK(65408)	1000us	20 bytes
70	X=PEEK(-128)	1170us	22 bytes
80	X=PEEK(65408)	1200us	21 bytes

ANTAL LEDIGA SEKTORER

Här kommer en subrutin som ger dig antal lediga sektorer på en flexskiva. Bra om du t ex vill testa om en fil får plats INNAN du försöker skriva ut den. Rutinen räkanar "0"-ställda bitar i bitkartan och fungerar för både enkel och dubbel packningstäthet (samt dubbelsidig/enkelsidig lagring).

Vad jag egentligen ville visa är hur jag tycker en subrutin bör vara kommenterad.

Observera att in-, ut-, och temporära variabler är beskrivna. Subrutinen har vidare ett namn! Anropet består av en GOSUB med namnet som kommentar. Radnumret som refereras är subrutinens namnrad (kommentarrad!). Man bör bevara en så här välkommenterad lista av programmet. Med ett PACKA-program tar man sedan bort kommentarerna och fixar så att radnumren pekar på "riktiga" rader.

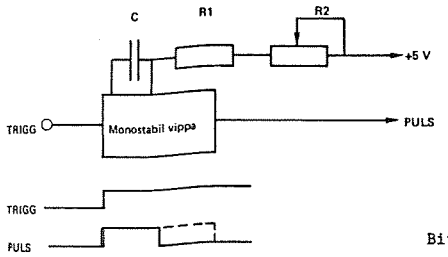
```
100 D%=0% : GOSUB 600 : REM *Sektkvar*
110 PRINT "Skivan har"K%" lediga sektore
    r"
120 STOP
600 REM *Sektkvar*
610 REM ------------------------------
620 REM Ger antalet lediga sektorer på
630 REM skiva i angiven drivenhet.
640 REM
650 REM In.
660 REM - D%   Divenhet (0/1).
670 REM Ut.
680 REM - K%   Antal lediga sektorer.
690 REM - E9%  Felkod. -1 vid fel, annar
    s 0.
700 REM Temp.
710 REM - K1%, K2%
720 REM ------------------------------
730 E9%=0% : POKE -767%,D% : Z%=CALL(246
    78%,192%)
740 IF PEEK(-747%) THEN E9%=-1% : GOTO 7
    90
750 K%=0% : FOR K1%=62720% TO 62879%
760 FOR K2%=0% TO 7%
770 IF (PEEK(K1%) AND 2%^I%)=0% THEN K%=
    K%+1%
780 NEXT K2% : NEXT K1%
790 REM *Sektkvar-end*
800 RETURN
```

JOYSTICK

Då vi nu publicerar två versioner av ett program, där det ena ska köras med en joystick (fröjdpinne?), samtidigt som önskemål om mera hårdvara i tidningen har blivit aktuella, publicerar vi här, med benäget tillstånd av copyrightsinnehavarna, delar av boken Avancerad programering på ABC-80.

Joystick är en slags manöverspak som finns till de flesta TV-spel. Den kan röras åt alla håll, och kan t ex användas för att rita bilder på skärmen...

...Det vi är intresserade av är alltså att överföra läget hos en spak till ett numeriskt värde (pulslängd). Det finns ett digitalt byggelement som lämpar sig utmärkt för detta ändamål - den monostabila vippan, som har den trevliga egenskapen att när vi ändrar ingången från "nolla" till "etta" (triggar kretsen), så kommer vi på utgången att få en puls, vars längd enbart beror på den kondensator (C) och det motstånd (R) vi har anslutit till kretsen.



Närmare bestämt lyder förhållandet så här:

PULSLÄNGD(sek)=0.45xR(Ohm)xC(Farad)

Ett exempel:

R=100kiloohm
C=10 mikrofarad

ger pulslängden =0.45 sekunder.

Men än så länge får vi alltid samma pulslängd, och det är ju inte mycket till A/D-omvandlare. Vi byter således ut R mot ett variabelt motstånd, en potentiometer. (Om vi väljer en termistor - temperaturkänsligt motstånd - som det variabla motståndet så har vi fått en termometer! Det enda som behövs är lite kalibrering, så har vi ett instrument som kan mäta temperaturer på begäran från datorn. Vi har fått ett mätsystem!)

Genom att välja en lämplig kondensator och lämplig resistans på det variabla motståndet kan vi få pulslängden att variera inom de gränser vi vill ha den...

Hjärtat i joysticken är två sådana A/D-omvandlare (en för X- och en för Y-koordinaten), och två potentiometrar som styrs av spaken. Den ena potentiometern vrids när vi rör spaken i sidled, den andra när vi rör spaken fram och tillbaka.

När vi beordrar en mätning med MÄT-X (dvs när vi triggar monovippa 1) kommer vi att få en puls vars längd beror på X-potentiometern, dvs spakens läge i X-led. Så länge pulsen pågår kommer signalen på denna tråd att vara en "etta". Genom att mäta denna tid, och sedan göra omproceduren för Y-potentiometern (MÄT-Y etc) får vi två numeriska värden som entydigt bestämmer spakens läge.

De två tryckknapparna S1 och S2 är kopplade till varsin bit i V24-porten. När en knapp trycks in sätts motsvarande bit till "etta".

Det enda som återstår nu är att utföra signalanpassningen. V24-porten lämnar ifrån sig (och vill ta emot) -12V som "etta" och +12V som "nolla", medan TTL-kretsen arbetar med +5V resp 0V.

Programmässig anslutning

Joysticken kan anslutas till ABC-80's V24-kontakt (I/O-port nummer 58) enligt följande:

I/O-port 58

Bit 7 ...						... Bit 0
---	---	---	MÄT Y	MÄT X	S1	S2	PULS
Användes ej			ut	ut	in	in	in

Från/till ABC80

MÄT-X och MÄT-Y är de signaler som beordrar mätning av X, resp Y-koordinaten. S1 och S2 är insignaler från tryckknapparna. PULS slutligen är den bit där vi får in pulsen från joysticken. Denna bit är "etta" så länge som pulsen pågår, och "nolla" annars.

För att beordra mätning av X-koordinaten (Y-koordinaten) måste vi skicka ut en "nolla" följt av en "etta" på MÄT-X (MÄT-Y), som triggar monovippan.

```
1010 OUT 58,0 : OUT 58,0 : REM MÄT-X
```

Sedan ska programmet mäta hur lång pulsen är (dvs hur länge det dröjer innan bit 0 blir "nolla" igen). Det kan göras så här:

```
1015 X%=0%
1017 X%=X%+1%
1020 IF (INP(58%) AND 1%)=1% THEN
1017
```

Men, för att få loopen så snabb som möjligt (vi vill ju inte vänta mer än nödvändigt på ett mätvärde) tillåter vi oss att använda en lite knepigare metod:

```
(1015 behövs ej)
1017 FOR X%=0% TO 32767%
1020 IF (INP(58%) AND 1%)=1% THEN NEXT X%
```

Så länge bit 0 är "etta" kommer NEXT X% att utföras - och X% räknas alltså upp - men när bit 0 blir "nolla" igen (pulsen är slut) fortsätter programmet med nästa rad. X% innehåller då ett tal som motsvarar stickans läge i X-led.

FOR-loopen går betydligt snabbare än GOTO-varianten, så vi behöver inte vänta lika länge för att få samma upplösning (givetvis måste pulslängden från joysticken justeras också).

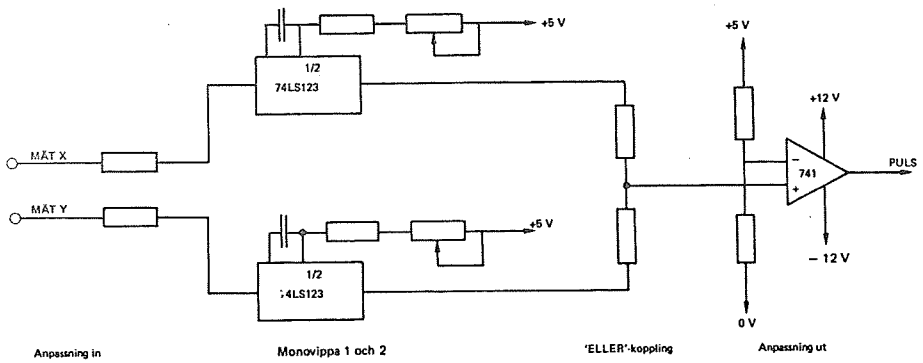
De värden vi får behöver förmodligen skalas för att passa vårt ändamål. Om vi ska använda värdena för att placera markören t ex, ska X-värdet variera mellan 0 och 23 och Y-värdet mellan 0 och 39. Men om vi ska rita i grafisk mod blir motsvarande värden 0 - 70 re p 2 - 79. Denna skalning går till så, att va tar reda på det största och det minsta X-värdet, kalla dem X0% resp X9%. Detta kan vi göra en gång för alla. Dessa värden använder vi sedan i vårt program. Om vi vill ha X-värden mellan 0 och N1 skriver vi:

```
10 X0%=13% : X9%=144% : REM uppmätta
värden
20 N1=23 : REM för textmod. Notera flyttal

30 X1=N1/(X9%-X0%) : REM notera flyttal

.
.
1050 X%=(X%-X0%)*X1 : REM skala X
```

Sedan gör vi samma sak med Y-värdet.

Ett litet program som läser av joysticken och sätter en markör på motsvarande punkt på skärmen kan då se ut så här:

```
 10 REM *hpprog-seq*
 20 REM Initiera
 30 X0%=13% : X9%=144% : REM (våra uppmä
    tta värden)
 40 N1=23 : REM För textmod. Notera flyt
    tal.
 50 X1=N1/(X9%-X0%) : REM Notera flyttal
 60 Y0%=13% : Y9%=144% : X1%=0% : Y1%=0%
 70 N2=39 : REM Textmod
 80 Y1=N2/(Y9%-Y0%)
 90 ; CHR$(12%);
100 REM *markör-ite*
110 GOSUB 230 : REM *mät*
120 IF R%=1% THEN 130
130 ; CUR(X1%,Y1%);' '; : REM Sudda gaml
    a markören
140 ; CUR(X%,Y%),'/'; : REM Sätt nya mar
    kören
150 X1%=X% : Y1%=Y% : REM Kom ihåg koord
    inaterna
180 GOTO 100
190 REM *markör-end*
200 REM *hpprog-end*
210 END
220 REM ------------------------------
230 REM *mät*
240 REM Subrutin som läser in X% och Y%
    från
250 REM JOYSTICKen.
260 OUT 58%,0% : OUT 58%,8% : REM Mät X
270 FOR X%=0% TO 32767%
280 IF (INP(58%) AND 1%)=1% THEN NEXT X%
290 OUT 58%,0% : OUT 58%,16% : REM Mät Y
300 FOR Y%=0% TO 32767%
310 IF (INP(58%) AND 1%)=1% THEN NEXT Y%
320 X%=(X%-X0%)/X1 : REM Justera X
330 Y%=(Y%-Y0%)/Y1 : REM Justera Y
340 REM *mät-end*
350 RETURN
```



Vi kan också använda oss av tryckknapparna för att t ex rita på skärmen. Vi lägger till några rader i programmet:

```
 5 IF R%=1% THEN 110
.
.
160 IF (INP(58%) AND 2%)=2% THEN R%=1%
: REM S1
170 IF (INP(58%) AND 4%)=4% THEN R%=0%
  REM S2
```

Om vi trycker på S1 kommer härefter markören inte att suddas när vi flyttar den, utan den ritar en linje. När vi trycker på S2 kommer markören i stället att sudda alla positioner den passerar.

När vi rör spaken fort märker vi att markören flyttar sig "hoppigt". Detta beror på att vi inte hinner mäta tillräckligt ofta - räkneloopen i Basic tar ganska lång tid (lång i datorsammanhang: flera tiotals millisekunder). Detta kan vi klara upp genom att skriva räkneloopen i maskinkod i stället för i Basic. Maskinkodsprogrammet går mycket fortare än Basic-programmet, så va kan använda kortare pulslängder och ändå få samma upplösning, fast på mycket kortare tid. (Assemblerdrivrutinen för joystick beskrivs i kap 9 i denna bok.)

SLALOM

I Luxor's monter på Tekniska mässan i höstas träffade vi en grabb som stod och lekte med ett slalomprogram i en ABC80. Min grabb fick en utskrift med sig hem och knappade in den under kvällen. Det han ”ck var själva algoritmen som de stora abbarna säger, dvs funktionen i programmet.

Vi satte till lite inmatning och poäng och döpte det till SLALOM. I samma veva fick vi veta att ett program kallat HOSPITAL vunnit hederspris i Luxor's skoltävling under hösten, och att detta fungerade på ungefär samma sätt. Ingen har väl satt sig in i det i detalj, men nog förefaller det att vara samma program. Vem som skrev in programmet på mässan vet vi inte, men vinnarna med HOSPITAL, Lars Oestreicher, Anders Andersson och Tomas Svensson har sagt OK till att vi publicerar programmet. Sak samma har Luxor gjort . Nog om detta.

För att få lite mera fart på det gjorde vi om det till ett JOY-program, där man med joy-stickan väljer fart och banlängder och sedan styr åt höger eller vänster och kör fortare eller saktare. Det blev en ganska häftig version. Vi publicerar bägge här. Den tangentbordsstyrda kanske inte är så fullkomnad, men fungerar i alla fall.

För att kunna köra Joy-versionen måste du (förutom att ha en joy från Comporian) först ladda in joy-stickans drivrutin. Den ligger på samma skiva eller kassett som spelen som följer med joyen. Sedan är det 'bare å åk'.

Mina grabbar håller på att köra igenom Luxor's spelprogram (Spelpak 1, 2 och 3). Ettan och tvåan tycks mest vara skåpmat sett ur deras synvinkel (de har två proppfulla Micropolis-disketter), men trean innehåller tydligen en hel del verkligt bra saker. En närmare redogörelse följer, men redan nu kan sägas att du inte köper grisen i säcken om du gillar spel. Olle Mauritzon

```
 1 REM SAVE SLALOM.JOY
 2 REM Prototyp av en pigg skolkille på
   besök i Luxors stånd på Tekniska Mäss
   an
 3 REM Hans utskrift kodad av Håkan Maur
   itzon
 4 REM Joy-version o regler av Olle Maur
   itzon
 5 GOSUB 67
 6 GOSUB 50
 7 P%=0% : K%=15% : B%=1% : T1%=15% : GO
   SUB 64
 8 FOR I%=1% TO L%
 9 R%=RND*3-1 : T%=T%+1% : IF T%=10% T%=
   0% : B%=0%-B%
10 K%=K%+R%+B% : IF K%<0% K%=0%
11 IF K%>29% K%=29%
12 ; CUR(23%,0%)STRING$(K%,127%);
13 ; CUR(23%,K%+10%);STRING$(40%-K%-10%,
   127%);
14 GOSUB 44
15 F1%=((0%-Y%+79%)*4%)+F%
16 IF X%>35% AND X%<45% GOTO 21
17 IF X%<10% T1%=T1%-2% : GOTO 21
18 IF X%<40% T1%=T1%-1% : GOTO 21
19 IF X%<70% T1%=T1%+1% : GOTO 21
20 T1%=T1%+2%
21 IF T1%>39% T1%=39%
22 IF T1%<0% T1%=0%
23 IF DOT(36%,T1%*2%) OUT 6,11% : P%=P%+
   1%
24 ; CUR(12%,T1%)"W"
25 FOR U%=0% TO F1% : NEXT U% : OUT 6%,0
   %
26 NEXT I%
27 GOSUB 65
28 ; CUR(13%,0%)SPACE$(11%*40%);
29 ; CUR(23%,0%)' Tid:'T' sekunder';
30 IF P%=0% ; CUR(13%,0%)' PERFEKT ÅK!
   ' : GOTO 35
31 ; CUR(13%,0)P%' av'I%-1%' meter utanf
   ör banan'
32 IF P%>I%/5 ; : ; ' Åtskilliga åskådar
   e ligger kullkörda'
33 IF P%>I%/5 ; ' utefter din väg genom
   terrängen.' : GOTO 35
34 IF P%>I%/10 ; : ; " Rätt mycket snö u
   nder skjortan va'"
35 P1=L%/16 : REM Toppfart = 16.7 m/sek
36 P2=T-P1 : REM P2 = 0-?. Poäng under b
   ästa tid
37 P3=(1000%-L%)/100% : REM P3 = 0-9.5.
   Poäng under max banlängd
38 P4=100*P%/L% : REM P4 = Procent utanf
   ör banan. Minuspoäng för lössnökörnin
   g och turistmassaker
39 P%=100%-P2-P3-P4 : IF P%<7% P%=2% : R
   EM Inte göra småbarn ledsna
40 ; : ; ' Du fick'P%' poäng av 100 för
   åket.'
41 ; : ; ' Nytt lopp "J". Ny tävling "N"
   '
42 GOSUB 44 : IF N1% 6 ELSE IF J1% ; CHR
   $(12%) : GOTO 7
43 GOTO 42
44 X%=CALL(65408%,1%)
45 Y%=SWAP%(X%) AND 255%
46 X%=X% AND 255%
47 J1%=(INP(58%) AND 4%)=0%
48 N1%=(INP(58%) AND 2%)=0%
49 RETURN
50 ; CHR$(12)
51 ; CUR(4%,0%)'Välj hastighet med "J"'
52 ; CUR(8%,0%)'Välj banlängd med "N"'
53 GOSUB 44
54 X%=X%/16%+1%
55 F%=(5%-X%)*120%
56 ; CUR(4%,22%)X%' '
57 IF J1% GOTO 59
58 GOTO 53
59 GOSUB 44
60 L%=(X%/4%+1%)*50%
61 ; CUR(8%,22%)L%' '
62 IF N1% RETURN
63 GOTO 59
64 POKE 65008%,255%,255% : RETURN
65 Z%=PEEK(65008%) : S%=PEEK(65009%) XOR
   255% : IF Z%<>PEEK(65008%) 65
66 Z%=((Z%-1%) AND 255%) XOR 255% : Z%=S
   %*51.2+Z%/5 : T=Z%/10 : RETURN
67 POKE 65408%,123%,230%,3%,40%,57%,95%,
   33%
68 POKE 65415%,240%,253%,126%,190%,40%,2
   53%,22%
69 POKE 65422%,8%,205%,155%,255%,77%,22%
   ,16%
70 POKE 65429%,205%,155%,255%,97%,251%,2
   01%,175%
71 POKE 65436%,211%,58%,122%,211%,58%,6%
   ,10%
72 POKE 65443%,16%,254%,33%,233%,255%,35
   %,219%
73 POKE 65450%,58%,163%,40%,250%,203%,36
   %,48%
74 POKE 65457%,4%,46%,0%,24%,7%,62%,79%
75 POKE 65464%,189%,48%,2%,46%,79%,201%,
   215%
76 POKE 65471%,141% : RETURN
```

```
 1 REM SAVE SLALOM
 2 REM A (stor bokstav) kör åt vänster
 3 REM ' (under *) kör åt höger
 4 Z%=CALL(630%)
 5 K%=15% : B%=1% : T1%=15%
 6 FOR I%=1% TO 200% : REM I% (200%) = B
   anans längd
 7 R%=RND*3-1 : T%=T%+1% : IF T%=10% T%=
   0% : B%=0%-B%
 8 K%=K%+R%+B% : IF K%<0% K%=0%
 9 IF K%>29% K%=29%
10 ; CUR(23%,0%)STRING$(K%,127%);
11 ; CUR(23%,K%+10%);STRING$(39%-K%-10%,
   127%)
12 A%=INP(20%) : IF A%=128%+65% T1%=T1%-
   1% : IF T1%<0% T1%=0%
13 IF A%=128%+ASC("'") T1%=T1%+1% : IF T
   1%>39% T1%=39%
14 IF DOT(36%,T1%*2%) OUT 6,11% : P%=P%+
   1%
15 ; CUR(12%,T1%)"W"
16 FOR U%=1% TO 500% : NEXT U% : OUT 6,0
17 NEXT I%
18 REM U% (500%) = farten. Lägre siffra
   = högre fart, högre siffra = lägre fa
   rt
19 ; CUR(13%,0%)SPACE$(11%*40%);
20 ; CUR(15%,0)P%' Portar av'I%-1%' miss
   ade'
```

SM 6 CYZ

Översänder
gar av ett lo
En lämplig

Efter att ha
program i
även i ABC
version sor
vid felslagn
från det av
senterade
förstås SM0D

Automat
för QSL ha
minnesutry
som har t
Dock kan
möjlighet. (
För att
jag tagit m
samt tid för
Vidare finns
läge även n
radera någo
Sortering
dvs efter
andra ord
vilket kan
lista hela n
funktionen.
och sedan
återgång til
Genom a
är det lätt
En del GE
kan verka
för att ta
ningen. De
men funkad
På rad 6(

Som sag
på att gö
inte allt ma
vid en enkel
sådant int
Det finns j
65060,0:GOT

SM6CYZ
Gösta Bengt

```
 1 REM LOG
 2 REM Ide
   h
 3 REM och
 4 REM sam
 5 REM Gös
10 DIM D$=
   %,V$=12
   %=35% :
20 N%=PEEK
   EK(6505
   -710%
30 G%=INT(
40 IF G%<1
50 DIM A$(
60 X$="/"
   RST BAN
   )
70 REM -
80 REM - F
90 REM -
100 ; CHR$(
110 ; CUR(1
120 ; CUR(3
   29%)"DA
130 ; : ; T
```

Marknadsföring

Vad gäller Luxors planer för marknadsföringen av den nya datorn och kommande nya produkter är frågan fortfarande öppen.(*) Förhandlingar pågår dels med den tidigare samarbetspartnern Scandia Metric och dels med andra tänkbara samarbetspartners. Som stomme i distributionen tänker dock Luxor använda sitt s k Team 100 - en grupp av radiohandlare med tillräckliga kvalifikationer för att handha och sälja datorprodukter. Men när det gäller t ex försäljning till skolor och andra liknande institutioner avser Luxor att bl a knyta kontakt med förlag och andra organisationer som redan arbetar med dessa målgrupper.

Produktionen

Vad gäller produktionsstart är denna satt till senvåren. Order kan först tas under maj månad och leveransstart är preliminärt satt till efter industrisemestern. Något exakt pris på enbart datordelen finns inte idag, men priset torde uppskattningsvis hamna i trakten av 13 000 kronor. Till hösten planneras också presentation av en ny skivminnesenhet med totalt 1M byte lagringskapacitet, samt utrustning för avbrottsfri kraft.

* Det har nu meddelats att Luxor har skrivit på ett kontrakt med Facit, som genom sina egna kanaler ska sälja ABC800 i både Sverige och utlandet.

RENGÖR KASSETTMINNET!

Dålig inläsning från kassettminnet kan ofta bero på att tonhuvudet blivit belagt med material från tonbandet, speciellt om man använder band av lägre kvalitet. Själva in/avspelningsspalten, som bara är några 100-dels mm bred, ger dålig verkan på ett band som genom beläggning på tonhuvudet passerar på flera 10-delars avstånd. Effekten av ett smutsig huvud visar sig i dålig diskantåtergivning och över huvudtaget svag signalstyrka, beroende på att bandet inte går tillräckligt nära spalten. Så kallad rengöringstape är inte till någon större nytta.

Nedanstående förfaringssätt gör betydligt bättre effekt. Plocka ut kassetten, lämna luckan öppen och tryck in PLAY-knappen. Då blir två tonhuvuden synliga. Det högra är in- och avspellningshuvudet, det vänstra raderhuvudet. Ta en sk Topz - en pinne med bommulstussar i ändarna (finns på apoteket), doppa den i T-sprit och gnugga rent tonhuvudet på den sida som bandet löper mot. Upprepa tills en ny Topz förblir ren. Genom att starta bandspelaren kan också kapstanrullen - det lilla gummihjulet som du ser snurra till höger om tonhuvudet lätt rengöras. Vad du än gör så försök aldrig att smörja mekanismen på något sätt - resultatet blir bara trubbel.

Skulle det vara så att en rengöring av mekanismen skulle vara ABSOLUT NÖDVÄNDIG måste bandspelaren plockas isär. Vill man göra detta så låssas de fem skruvarna med gul plastbricka på undersidan, men DRA FÖRST UT NÄTSLADDEN!

Sedan kan den beiga plåtlådan lossas försiktigt. Lossa kassettluckan genom att dra ut den vinkelböjda gångjärnssprinten. I kassettfacket finns fyra skruvar som ska lossas, varefter du försiktigt kan lyfta av den bruna överdelen. Lägg märke till vart sladden från RECORD-indikatorn ska anslutas. Nu kan du försiktigt rengöra de mekaniska detaljerna i drivsystemet, byta räkneverksrem mm. Kom bara ihåg att detaljerna ska tvättas med en Topz doppad i T-sprit.

Alla lager på de roterande delarna är självsmörjande, altså INGEN OLJA. Tittar du noga efter kan du se att vissa glidytor på hävarmarna insmorts med fett. Dessa punkter kan försiktigt smörjas med en liten, liten gnutta vaselin. Onödigt mycket vaselin binder damm.

När vi nu i alla fall har överdelen demonterad kan vi också justera tonhuvudets azimut, dvs spaltens vinkel mot bandets rörelseriktning. Den ska vara 90 grader. Redan ett litet fel på några grader ger en försämrad återgivning, mest märkbar i diskanten. Du kan försiktigt lägga en kassett på mekanismen och starta bandspelaren med PLAY. Lyssna på det brus som kommer från ex vis ett oinspelat band. Det går bra att köra bandspelaren via en stereoanläggning om du tycker att ljudet blir svagt i ABC80's högtalare. Azimuten justeras med de två skruvarna som håller fast tonhuvudet. Var observant på att tonhuvudet sitter monterat på en fjäderbelastad plåt som inte får påverkas vid avlyssningen.

Gör ett prov med ett inspelat dataprogram och kontrollera att inställningen stämmer också där. Det kan nämligen vara så att inspelningen gjorts med felinställt tonhuvud med resultat att avspelningen inte blir fullgod. I så fall måste man söka en kompromiss så att de gamla inspelningarna kan läsas. Naturligtvis sker hopmonteringen i omvänd ordning mot isärtagningen. Glöm inte sladden till RECORD-indikatorn. Har du nu gjort hopmonteringen rätt ska det inte bli några skruvar över.

TV-Lindström

PROBLEM MED BANDSPELAREN?

Min bandspelare har gått relativt bra vid inkörningar och inspelningar. Men när jag köpte en printer och hade den inkopplad samtidigt som jag körde in ett program från bandspelaren blev det ERR 35 -CHECKSUMMA FEL VID LÄSNING hela tiden.

Jag undrade förstås vad det berodde på, så jag testade med att koppla ur interfacet. Då funkade det bra igen. Jag trodde ju då att det var fel på interfacet - vem kunde tro annat? Så fortsatte det ett bra tag, tills jag fick höra att det kunde bero på en kondensator i bildskärmen. Det är kondensatorn C101 som sitter på ett anpassningskort i bildskärmen.

Jag ska nu gå igenom, steg för steg, hur man kommer åt den och tar bort den. Börja med att ta bort bakstycket på bildskärmen genom att lossa 4 skruvar. Om du ser in i skärmen bakifrån så ser du på vänster sida i övre hörnet ett kort med bl a kraftenheten. På nedre delen av detsamma sitter en långsmal vit kontakt som du först ska lossa genom att försiktigt dra ut den. Man kanske får hjälpa till med en skruvmejsel om det går trögt. Dra sedan ut det stora kortet som sitter i ett spår i nedre delen av skärmen. Du ska samtidigt dra ut det lilla kortet på höger sida för att sladdarna går dit och räcker inte till annars.

På det större kortet sitter ett mindre kort. Det sitter bakom den högra kylplåten och bredvid en vit kontakt. Dra ur detta kort, rakt upp. På detta kort ser Du att kondensatorn C101 sitter.

Nu är det bara att löda bort den. Värm tennet på undersidan och dra försiktigt bort kondensatorn från översidan med en tång. Värm ej tennet för länge, då kan foliet på kortet lossna. När du tagit bort den är det bara att sätta ihop allting igen. Kontrollera först att inget tenn har flutit över till några andra folieledare.

När jag hade gjort denna manöver fungerade inkörningen perfekt - utan det förargliga ERR 35.

Bjarne Borg

Intresserfördelning m.m. bland ABC-klubbens medlemmar.

Vi har gjort en bearbetning av de förkryssningar på inbetalningskorten som gjorts av alla medlemmar 1980, dvs alla som fanns med i matrikeln. Av 1432 st har 1220 (85%) markerat intresse.

För dessa 1220 st är den procentuella fördelningen:

Administrativa rutiner	49,7
Datakommunikation	40,3
Föreningsliv, div tillämpn	22,5
Matematik och statistik	42,7
Programmeringsteknik	77,8
Spel, grafik och musik	49,9
Styr o Mättekn elektronik	57,3
Textbehandling	39,1
Hjälpa till	11,0
Har tillgång till:	
Flexskiva	45,5
Modem 300 baud	13,5

Det betyder att närmare hälften av klubbens medlemmar har tillgång till flexskiveutrustning, i många fall sannolikt på arbetsplatsen eller skolan, även om man inte disponerar en egen utrustning. Det skall bli intressant att se hur siffrorna förändrar sig från år till år.

A B C-bladet Samlingsnummer
1981 : 2 sidan 6

Bakgrunds/förgrunds-system på ABC80

Vid ett flertal tillfällen har vi stött på önskemål om att med ABC80 kunna utföra fler uppgifter samtidigt. Man har vid dessa tillfällen skissat på någon form av 'time-sharing-system'. De uppgifter man har velat att ABC80 skall utföra har ofta varit någon form av I/O-hantering samtidigt som man har velat exekvera administrativa program typ bokföring, fakturering, löner etc. Den I/O-hantering som varit aktuell har t.ex. varit larmövervakning, styrning och mätning.

För att tillmötesgå de önskemål som funnits har vi under hösten och vintern 1980-81 gjort ett system som tillåter att man har en automatisk I/O-hantering samtidigt som man kan köra Basicprogram om vanligt på sin ABC80. Systemet har alltså två nivåer där program kan exekveras. Den nivå där I/O-hantering utförs kallas BAKGRUND och Basic-nivån kallas FÖRGRUND.

Vi har i det pilotprojekt vi här skall beskriva tagit fasta på larmövervakning och styrning. I projektet ingår även inhämtning av mätvärden. Dessa hanteras ej på annat sätt än att de inläses och läggs upp i en tabell. Ett programpaket i Basic ingår också. Detta skall administrera larmutskrifter, filhantering och ändringar av textutskrifter och systemtabeller. I Basicprogrammet ingår också gränsvärdesövervakning av de mätvärden som bakgrunden hämtar in till ABC80.

Systemet är hårdvarumässigt en standard ABC80 konfiguration så när som ett speciellt bakgrunds/förgrunds-interface som pluggas på bussen.

Programvaran för systemet är skriven i Assembler och ligger prommad i 8K byte ledigt samt på IEC platsen. Systemet utnyttjar RAM-minne beläget under DOS-offert 0. Hur stor RAM-area som behövs beror på aktuell applikation. I vårt fall tas ca. två kbyte i anspråk. När systemet startas upp hämtas systemvariablerna från PROM-minnet och läggs i RAM för att man enkelt skall kunna ändra i systemet under drift.

Bakgrunden exekveras i intervall som ges av en realtidsklocka som ger interupt via kassettingången. För att förgrunden ej skall förefalla 'ryckig' för den som betraktar bildskärmen sprids arbetsbelastningen jämt under tidsinter- vallet i olika faser.

För att kunna kommunicera med bakgrunden finns möjlighet att lägga upp kommandon som har samma status som de vanliga Basickommandona typ RUN, LIST, SAVE etc. En kommunikationskanal med Basicmiljön finns också implementerad så att samarbetet mellan förgrund och bakgrund skall löpa enkelt.

Programvaran servar också med statusinformation typ ÅTERSTART EFTER SPÄNNINGSBORTFALL, SYSTEMET OMSTARTAT samt information om hur de enskilda faserna lyckats eller misslyckats.

I systemet ingår också ett modifierat DOS som möjliggör autostart av programsystemet och även autostart av förgrunden från skiva.

I bakgrunds/förgrunds-interfacet ingår kretsar för att säkerställa omstart om programsystemet skulle slås ut av störningar eller andra oförutsedda händelser.

PILOTPROJEKT.

Det pilotprojekt som vi drivit de senaste månaderna är ett larmövervakningssystem som skall hantera 112 ingångar, 16 utgångar och inläsning av 16 mätvärden. Via ingångarna för systemet besked om drift- och larm-indikeringar. Beroende på hur dessa signaler förhåller sig till varandra skall larmtexter skrivas ut och larmsignaler utlösas.

Till detta har vi utvecklat ett interpretatativt styrspråk som vi valt att kalla ILS (Interpretativ Logisk Styrning). Detta exekveras i bakgrunden. I ILS kan man utföra logiska operationer med in- och utgångar. ILS-språket innehåller också villkorliga hopp, subrutiner, indexerad adressering av in- och ut-gångar, tidsfördröjningar och förreglingar. Vidare finns ett antal olika sätt att tilldela en ingång eller utgång olika värden eller status. Dessa manipulationer kan göras både villkorsstyrda och ovillkorliga. Även möjlighet att programmera så kallade do-loopar finns. ILS kan programmeras on-line i ett installerat system. Detta gör det lätt att modifiera systemets funktioner.

I systemet finns program som läser in mätvärden och ingångar samt ställer ut utgångar. Andra program tar hand om tidsfördröjningar och annan hantering av ingångar och larmer. Vidare finns rutiner som hanterar almanacka och klocka. För att undvika att Basic-programmet låser sig eller blir hängande, t.ex. i en inputsats, finns rutiner för timeout både i absolut tid samt vid för låg aktivitet vid tangentbordet.

Bakgrunden sköter all I/O-hantering och konverterar ingångar efter den ingångsdefinition som finns lagrad i en tabell. I andra tabeller ligger uppgifter om fördröjningstider för till- och frånslag. Bakgrunden sätter även flaggor när ingångarna ändrar värde. Dessa flaggor kan Basicen testa på för att avgöra om larmutskrift skall göras eller ej.

Till ingångarna hör en definition som beskriver hur systemet skall uppfatta den fysiska ingången, hur den skall tolkas och presenteras. Här anges om ingången skall inverteras, vilken prioritet den skall ha, om den skall indikeras som larm, om den skall tidfördröjas och i så fall om det skall ske vid till- eller frånslag eller vid både till- och frånslag. Här anges också om ingången överhuvudtaget skall läsas in eller om den skall vara fiktiv. Fiktiva ingångar kan ILS hantera som hjälpvariabler. Även fiktiva ingångar kan generera larmutskrifter. På detta sätt kan man få ILS att generera larmutskrifter i förgrunden.

Man kan beordra bakgrunden att lägga upp all väsentlig systeminformation i Basicvariabler så att den blir enkelt hanterbar från Basic. Bakgrunden tillhandahåller även editeringsmöjligheter i textsträngar.

Förgrunden behöver således endast bekymra sig om att administrera larmutskrifter och ge support med operatörskommunikation. De kommandon som tillhör bakgrunden kan enkelt släppas vidare från förgrunden till bakgrunden i klartext i en sträng.

Förgrunden kan hantera all kringutrustning som ej kräver egen interuptservice eller störs av att bli avbruten under dessa relativt sett långa intervall.

BAKGRUNDS/FÖRGRUNDS-SYSTEM FÖR ANDRA ÄNDAMÅL.

Den teknik som här redogjorts för kan naturligtvis även användas för att bygga andra typer av system.

En tänkbar tillämpning är olika typer av mätsystem där mätdata kan samlas in av bakgrunden i en mängd och i ett intervall som bestämms av den aktuella mätsituationen. Utvärderande Basicprogram kan köras samtidigt i förgrunden. Hur ett sådant mätsystem skulle se ut kan ju var och en fundera över.

Ett annat område där tekniken är tillämpbar är automatisering av olika typer. Bakgrunden kan här räkna, styra, sortera, indikera osv osv medan förgrunden används för operatörskommunikation och för presentation av den process som systemet hanterar.

Generellt kan man säga att tekniken är användbar i alla sammanhang där man vill ha ett kontinuerligt arbetande system och samtidigt vill kunna göra saker som normalt skulle bryta kontinuiteten.

VIRTUELLA SUBRUTINER.

Ett problem när man vill hantera stora datamängder i primärminnet är att relativt lite minnesutrymme återstår för programmen. Ett sätt att lösa detta är att använda CHAIN-instruktionen i Basic. Detta förfarande har en stor nackdel i det att man förlorar sina variabler som man med stor möda skaffat sig. Ett vanligt förfarande är då att lagra dessa variabler på disketten för att i nästa program läsa in dem igen. Detta är ett tidsödande moment där dessutom ofta en operatör är inblandad som tycker att systemet är trögt och svampigt. Genom att poka undan variablerna kan dessa peekas fram igen. Denna metod är snabbare men betydligt omständligare.

Vi har löst problemet med ett programsystem vi tagit fram under hösten 1980 och som vi kallar 'VIRTUELLA SUBRUTINER'. Enkelt uttryckt kan man säga att tekniken medger att man lägger upp sina programsegment på skivan och kastar ut de man inte för närvarande behöver ur minnet och ropar in ett nytt från skivan. Detta kan göras utan att variabler går förlorade. Detta programsystem beskrivs mer utförligt i en annan artikel i detta nummer av ABC-bladet.

Virtuella subrutiner gör det möjligt att göra servicerutiner som kan anropas utan att man behöver stoppa programmet. Dessa servicerutiner har dessutom tillgång till alla de variabler som systemet hanterar.

Vissa funktioner som inte behöver användas så ofta läggs som en separat rutin och behöver därmed ej belasta ABC80's arbetsminne.

I larmsystemet har vi kallat den typen av funktioner för 'PROGRAM-KOMMANDON' eftersom de utgörs av ett program som anropas från systemet. Exempel på funktioner som utförts som programkommandon är kopieringsrutiner för disketter, ändring av systemdefinitioner, dumpning och hämtning av systeminformation på skivan.

Genom att kombinera bakgrund/förgrundstekniken med virtuella subrutiner kan man åstadkomma väldigt kraftfulla system till ABC-80. Den här något ytliga presentation ger bara en antydan om vad som kan åstadkommas med ett färdigt hård/mjuk-varusystem typ ABC80.

INOWE HB
08 / 81 21 60
Civ. ing. ERIC NORSTRÖM

A B C-bladet Samlingsnummer
1981 : 2 sidan 7

Virtuella subrutiner på ABC80

Vid konstruktion av mer komplexa program konstateras ganska snart att arbetsminnet i ABC 80 inte är tillräckligt. Den vanliga metoden är då uppdelning av programmet i fristående avsnitt och hopp mellan programavsnitten med CHAIN. Tyvärr måste då variablerna sparas antingen i en "POKE area" eller i en mellanlagringsfil på skiva eller kassett.

Nackdelen med reservation av arbetsminne för mellanlagring av data vid "CHAIN" är att programavsnitt för lagring och återhämtning av data tar arbetsminne och programmeringstid.

Nackdelen med att mellanlagra variabler och data i en tillfällig fil på skiva är att det tar tid för lagring "PRINT §L%", programbyte "CHAIN" och inläsning av data efter programbytet "INPUTLINE §L%".

Jag har kommit i kontakt med en tredje variant av programuppdelning, vilken visat sig fungera på ett trevligt sätt och den medför snabbare exekvering av programmen. Metoden kallas av konstruktörerna (INOWE HB) för **VIRTUELLA SUBRUTINER.**

Metoden innebär att programmet byggs upp med en **PROGRAMROT** från vilken uthopp till övriga programdelar (SUBRUTINER) sker. I PROGRAMROTEN definieras och öppnas (OPEN) de filer som skall användas senare i programet. GLOBALA variabler DIM:as och/eller om de ej erfordrar DIM tilldelas de ett värde. Variabler som endast används lokalt behöver inte nämnas i programroten.

Det intressanta med metoden är att man kan använda samma radnummer i subrutiner som i huvudprogrammet eller i andra subrutiner. Detta innebär att man kan lösa problem för ett program och sedan om man så önskar kan avsnittet utan ändringar användas i ett annat programm. Detta förutsätter emellertid att man håller reda på användningen av sina variabler.

Jag vill här åberopa Bengt Olwig:s artikel i ABC bladet nr 1 1981 där han uppmanar landets programerare att arbeta mot enhetliga och kompatibla programvarusnitt.

Om subrutiner och huvudprogram i olika programpaket använde variabler för samma ändamål - utopisk tanke - så skulle en subrutin kunna länkas in från ett programpaket till ett annat. Detta skulle medföra minskad programeringstid vid konstruerandet av nya tillämpningar.

Jag har hittills använt VIRTUELLA SUBRUTINER i två fungerande tillämpningar.

A Ett programpaket för tidsredovisning och faktureringsunderlag för min konsultverksamhet. Programmet blev efter omläggning till virtuellt minne mycket snabbare och gick dessutom att bygga ut med många fler funktioner. Möjligheten att hoppa ut i subprogram för beräkning av dagnummer till datum mm minskar programstorleken i huvudprogram och subrutiner. Bibehållandet av variablerna, vilket f ö är valfritt (se utdrag ur bruksanvisning nedan), gör programmet snabbt.

B Kommunikationsprogrampaket med monitorprogrammet som programrot och övriga funktioner som LÄSA, SKRIVA, HÄMTA, LÄMNA och tvåvägskomunikation OPERATÖR som virtuella subrutiner.

Programmen är utvecklade i samarbete med Magnus Lundberg, "NYFIKEN".

Fördelen med virtuell teknik är den mycket snabbare återgången till Monitorn efter avslutat delprogram samt möjligheten att överföra information mellan programdelarna på ett smidigt sätt.

Monitorprogramet skiljer sig på en avgörande punkt från de program som används av ABC-klubben och ELFA. Som grund har använts Magnus MENY-program vilket presenterats i ABC-blad nr 3 1980. Det är mycket enklare att välja vad man vill göra när hela menyn är presenterad och man slipper hålla valmöjligheterna i huvudet.

Nedan följer ett utdrag ur bruksanvisningen för VIRTUELLA SUBRUTINER. "...Detta programpaket gör det möjligt att "kedja" program med bibehållen variabelarea. Dessutom kan retur till det anropande programmet göras med inhopp omedelbart efter det ställe kedjningen gjordes ifrån. I det anropade programmet, subprogrammet, bibehålls de variabler som det anropande programmet, huvudprogrammet, använder. Dessutom kan lokala variabler användas som sedan "glöms bort" vid retur.

Huvudprogrammet påverkas ej av vad som sker i subprogrammet. Sålunda påverkas ej RESTORE eller ONERRORGOTO satser. Däremot kan senaste ERRCODE i subprogrammet avläsas i huvudprogrammet efter retur. De variabler som är gemensamma för programmen kommer att bibehålla sina värden både vid anrop och retur, ERRCODE däremot nollställs vid anrop men behålles vid retur. På detta sätt kan man från huvudprogrammet avgöra huruvida subprogrammet lyckades eller ej.

En annan finess är att öppnade filer behålles vid anrop och retur. En fil kan öppnas i huvudprogrammet för att sedan stängas i huvudprogrammet eller i ett subprogram. (OBS!! Alla filer måste öppnas i huvudprogrammet - programroten.) Alla programdelar kan läsa eller skriva på filen som vanligt.

Som framgår av ovanstående öppnar sig nya perspektiv för programmering på ABC80. Här följer några exempel.

1. Stora programvolymer kan enkelt organiseras och hanteras.

2. Program som för vissa beräkningar kräver stort minnesutrymme för variabellagring kan organiseras så att minnet utnyttjas optimalt vid varje moment. Lokala variabler "glöms" ju vid retur.

3. Smarta subrutiner kan skrivas och dokumenteras en gång för alla. I de programsystem där de behövs läggs de bara ned på skivan som en separat modul.

4. En gemensam datastruktur kan läsas in i huvudprogrammet och sedan bearbetas av flera olika subprogram, som sins emellan är oberoende av varandra, utan att man behöver använda mellanlagring av data på filer.

I och med att huvudprogrammet och subprogrammen kan betraktas som självständiga enheter vad programstrukturen beträffar kan denna teknik jämföras med "procedurer" i Algol.

Ett subprogram kan i sin tur anropa ett annat subprogram. Detta subprogram kommer då att betrakta det anropande subprogrammet, och dess huvudprogram, som ett huvudprogram. Variabler är gemensamma från och med den programnivå där de första gången används. Detta ger en hierarkisk variabelstruktur där subprogrammen hanteras som om de var subrutiner i huvudprogrammet.

Jämför detta med overlay-teknik i större datorsystem.

När ett program skall hämta ett subprogram från skivan sker detta på följande sätt. Subprogrammets namn laddas i strängen Q0$ och ett "CALL" görs till adress 64003. Programraden kan se ut så här.

```
120 Q0$="PGM12" : Z%=CALL(64003%) : REM Sortera A$()
```

Nu kommer programmet PGM12.BAC eller PGM12.BAS att läsas in och exekveras. Om PGM12 avslutas med ett vanligt RETURN i Basic kommer retur att ske till REM-satsen på rad 120 i det anropande programmet. Här kommer exekveringen att fortsätta som om ingenting hänt. Kan programsystemet ej hitta strängen Q0$ ges felkod 100.

Om man återigen vill anropa ett subprogram och ej har läst in något annat subprogram eller introducerat nya variabler sedan sist, kan man göra ett "CALL" till adress 64000. I detta fall behöver ej filnamnet finnas i Q0$. Genom detta förfarande spar man den tid som det tar att hämta in subprogrammet från skivan vid varje tillfälle. Detta är användbart vid uttryck av denna typ.

```
150 FOR I%=0% TO 99% : Z%= CALL(64000%) : NEXT I%
```

När subprogrammet anropas direkt i minnet på detta sätt, kan anropet göras så att de lokala variablerna i subprogrammet behålles från anrop till anrop. Det sker på så sätt att man tar med sig talet 1 i anropet. Raden blir då så här.

```
150 FOR I%=0% TO 99% : Z%= CALL(64000%, 1%) : NEXT I%
```

Om något subprogram ej kan lokaliseras i minnet går programsystemet vilse. Man torde dock bli varse att något inte står rätt till.

Här följer ett par subrutiner för att hålla reda på om ett subprogram finns i minnet eller skall hämtas från skivan.

```
9000 Z%=(Q0$()Q1$)*3% : Q1$=Q0$ :
Z%=CALL(64000%-Z%) : RETURN
```

Strängen Q1$ användes här för att lagra senast använda namn och får således ej användas för andra ändamål. Naturligtvis kan vilken sträng som helst användas för att hålla reda på senaste subprogramnamn.

Anm
Det är lämpligare att ange subprogrammets namn till Q1$ i subprogrammet i samband med avslutningen av subprogrammet.

```
Q1$='PROG12' : RETURN
```

Om man alltid vill behålla de lokala variablerna vid anrop i minnet kan subrutinen skrivas så här.

```
9000 Z%=(Q0$()Q1$)*3% : Q1$=Q0$ : Z%=
CALL(64000%-Z%,1%) : RETURN
```

Vid inläsning från skivan ignoreras ettan i anropet.
Med dessa subrutiner måste filnamnet finnas i Q0$ och rad 120 kan då se ut så här.

```
20 Q0$="PGM12" : GOSUB 9000 : REM
Sortera A$()
```

1) INIT.BAC Är en initieringsrutin för att hantera virtuella subrutiner.

2) SKIVPREP.BAC Preparerar den skiva som ska innehålla programsystemet.

När en skiva skall prepareras för att använda VIRTUELLA SUBRUTINER körs programmet SKIVPREP. Detta program förväntar sig sin egen skiva i drive 0 och en formatterad skiva i drive 1. Skivan i drive 1 kommer att göras iordning så att filen INIT.BAC finns där. Det är inget som sedan hindrar att filen INIT.BAC kompletteras med ett CHAIN samt döps om till START.BAC eller något annat lämpligt namn. Den rad som filen innehåller från början måste genomlöpas för att programmet skall fungera.
Observera att filnamnen på systemskivan ej får ändras. Slutsatsen av detta blir alltså att det enda program som skall köras på systemskivan för att få programsystemet att gå är just SKIVPREP.BAC.

SNABBFAKTA.
1) Subprogrammen skall avslutas med ett vanligt basic-RETURN istället för END för retur till huvudprogram.

2) Variabler är gemensamma från och med den programnivå där de första gången används.

3) DATA- och ONERRORGOTO-satser i huvudprogrammet påverkas ej av anrop.

4) ERRCODE nollställs vid anrop och behålles vid retur.

5) Filer kan öppnas i huvudprogram och blir då åtkomliga även för subprogrammen. Subprogrammen kan även stänga filen. Observera dock att filer ej får öppnas i ett subprogram för att användas i huvudprogrammet.

6) Programsystemet ockuperar en DOS-buffert. Antalet filer som samtidigt kan vara öppna reduceras alltså med en.

7) Initieringsprogrammet INIT.BAC behöver ej finnas i programmet utan kan kompletteras med en CHAIN till huvudprogrammet och tar därmed ej upp minnesutrymme i ABC80.

8) CHAIN får ej användas i subprogrammen om man vill kunna göra retur till huvudprogrammet. Härvid förstörs nämligen interna pekare i ABC80.

9) Programsystemet fungerar även med "skyddade" program.

10) Programmet fungerar på Basicintepretatorer med checksummor 11273, 9913 eller 10042.

11) Programsystemet fungerar på singel och dubbel densitet.

12) Strängar, vektorer och matriser måste dimensioneras i en DIM-sats eller tilldelas ett värde innan subprogrammanrop får göras. Variablerna kommer annars att kunna anta felaktiga värden och i värsta fall kommer programmet att gå vilse. Nya filnummer får ej introduceras i subprogrammen.

Mina erfarenheter av vitruella subrutiner visar att de gör ABC80 till en maskin med stora resurser. Vid premiärvisningen av nya generation i ABC familjen - ABC 800 - presenterades CHAIN med bibehållen variabellista. Denna variant är naturligtvis trevlig i och för sig, men resultatet torde bli en mängd onödiga inläsningar av grundprogremmet.
Jag tackar programmakarna till virtuella subrutiner för ett intressant prgrameringsverktyg och rekomenderar intresserade att kontakta Eric Norström på INOWE, tel 08/ 812160

STEN BURHAMMAR
V:a Bangatan 49A
195 00 MÄRSTA

Oslo, 1981-01-28.

Til ABC-bladet.

I all hast vil jeg si noen ganske få ord om de programmene jeg sender.
Programmet DIRFIX.BAC har følgende bakgrunn: En dag påstod plutselig min maskin at en viss skive inneholdt langt färre filer enn det jeg visste at den skulle inneholde. Aha, directory-feil! (Muligens i forbindelse med en 4116 RAM som gikk senere.) Som følge av litt nyfikenhet hadde jeg imidlertid en anelse om at det på hver skive eksisterer et reserve-direktory. Altså: kopir! Og så ble dette programmet til. Det kan brukes på to måter:
A. (den forsiktige måten) Kjör först funksjon 1; test skiven (f.eks. med LIB); kjör så funksjon 2.
B. (den raske måten) Kun funksjon 3.

Apropos 4116 RAM. Ifölge egne undersökelser ser posisjonene for de forskjellige bit'ene (0-7) på ABC80's hovedprint ut til å väre henholdsvis: D5, C5, D4, C4, D3, C3, D2 og C2. Altså er det galt som indikert i Markesjö's bok (Fig. 5.11, side 102).

Vennlig hilsen
Jon Kleiser
Frydensgt. 2 B
N - Oslo 5.

```
 10 REM ************************
 20 REM * DIRFIX.BAC           *
 30 REM * VER 1.0 / 1981-01-28 *
 40 REM * (C) ---------------- *
 50 REM * gjort av Jon Kleiser *
 60 REM * mod av ------------- *
 70 REM * innsendt av J.K.     *
 80 REM ************************
 90 REM
100 REM Programmets funksjon er å gjen-
110 REM skape directory (track 2,
120 REM hovedtrack) på grunnlag av
130 REM track 1 (reserve).
140 REM
150 ; CHR$(12%);
160 ; "*** DIRFIX - directory fix ***
170 ; : ;
180 DIM B$=256%,S$(2%)=256%
190 B$=SPACE$(256%)
200 A%=PEEK(65065%)+256%*PEEK(65066%)
210 POKE A%+6%,62720%,SWAP%(62720%)
220 ; "HVILKEN DRIVE BRUKES ? (0/1) ";
230 GET D$ : ; D$ : ; : ;
240 POKE -767%,VAL(D$)
250 PRINT "VELG FUNKSJON :" : ;
260 PRINT "  1) BYTT OM HOVED- OG ";
270 PRINT "RESERVE-TRACK"

280 PRINT "     (test så skiven)" : ;
290 PRINT "  2) KOPIER FRA HOVED- ";
300 PRINT "TIL RESERVE-TRACK";
310 PRINT "     ___" : ;
320 PRINT "  3) Kopier direkte fra ";
330 PRINT "reserve- til"
340 PRINT "     hoved-track" : ; : ;
350 PRINT "  ";
360 GET F$
370 IF F$<"1" OR F$>"3" THEN 360
380 PRINT F$"  - TRYKK RETURN ";
390 GET R$ : ; CUR(18%,7%)TAB(19%)
400 FOR S%=0% TO 7%
410 REM LES
420 FOR T%=1% TO 2%
430 Z%=CALL(24678%,SWAP%(T%)+32%*S%)
440 IF PEEK(-747%)=0% THEN 470
450 ; CUR(18%,7%)CHR$(7%)"LESEFEIL  "
460 GOTO 430
470 S$(T%)=B$
480 NEXT T%
490 REM SKRIV
500 T1%=2%
510 FOR T%=1% TO 2%
520 IF T%=1% AND F$="2" THEN 600
530 IF T%=2% AND F$="3" THEN 590
540 B$=S$(T%)
550 Z%=CALL(24675%,SWAP%(T1%)+32%*S%)
560 IF PEEK(-747%)=0% THEN 590
570 ; CUR(18%,7%)CHR$(7%)"SKRIVEFEIL"
580 GOTO 540
590 T1%=1%
600 NEXT T%
610 NEXT S%
620 PRINT CUR(18%,7%)"UTFÖRT     " : ;
630 END
```

Sven Wickberg

Datorer kan
gör vad prograi
mer eller min
datorer som a
som spelar me
'övar upp sig'
bättre.
I Scientific
kanska forska
poker. De lä
spelar bättre
också en artil
Backgammon (
som lyckades
ren! Hur det
inte.)
Nu skulle m
lärande progr
ett enkelt pr
en nybörjare,
det spelar?
Ett spelprog
lig algoritm, a
som skall ge
början ha n
draget är ell
vissa spel fir
matisk strateg
Ett enkelt
finns i många

'Vi lägger 21
omväxlande 1-
stickan har v
Med en stu
komma på h
svår variant
'Vi lägger va
antal högar.
högst alla s
drar sista sti
Även om n
är det näst
rätta draget
det fort som
den får cha
rätta draget.
Sådana spe
när man väl g
man har mo
med det rätt
inget utrymn
De flesta
komplicerade
mycket om me
kärt barn...
Det spelas
går utpå att
in motstån
dem, dvs
finns ingen e
skulle behöva
samt alla tän
är fullt för
ger mest br
hinna med (
Den varia
sig med att
position so
motståndar
säkert att
leder till slu
Om datorn
righetsgrade
för 'bästa d

```
 10 REM ...SAVE KALAHA...v.4.6% 80 03 1
    8
 20 REM EFTER 'AWARI' I BASIC COMP.GAME
    S
 30 REM Sven Wickberg 0750-50456
 40 REM Baldersv 43, 130 54 Dalarö
 50 REM ny vers.m.ext.'minne' 800604
 55 REM förb.form, procent 81 03
 60 REM ------------------------------
    ---
 65 REM GOSUB 1000
 70 DIM F(50%)
 75 OPEN "KALAHA.DAT" ASFILE 1%
 77 INPUT §1%,N% : FOR I%=0% TO N%-1% :
    INPUT §1%,F(I%) : NEXT I% : CLOSE 1
    %
 80 ; CHR$(12%)
 90 DIM B%(13%),G%(13%)
100 ; 'UTGANGSSTÄLLNING (välj 1-6):'
110 FOR I%=0% TO 12% : B%(I%)=3% : NEXT
    I%
120 C%=0% : B%(13%)=0% : B%(6%)=0%
125 F(N%)=0%
130 GOSUB 490 : REM --- RITA BRÄDET
140 ; "DITT DRAG"; : GOSUB 310
150 IF E%=0% THEN 230
160 IF M%=H% THEN GOSUB 290
170 IF E%=0% THEN 230
180 ; "MITT DRAG:"; : GOSUB 690
190 IF E%=0% THEN 230
200 IF M%=H% THEN ; "VÄNTA - JAG HAR EX
    TRADRAG:"; : GOSUB 690
210 ;
220 IF E%>0% THEN 140
230 ; : ; : ; "SLUT PÅ SPELET"
240 D%=B%(6%)-B%(13%) : IF D%<0% THEN ;
    "JAG VANN MED";-D%" POÄNG" : GOTO 2
    70
250 N%=N%+1% : IF D%=0% THEN ; "OAVGJOR
    T" : GOTO 270
260 ; "DU VANN MED";D%" POÄNG"
270 ; : ; "NYTT SPEL"; : INPUT W$ : IF
    W$="J" THEN GOTO 100 ELSE GOTO 890
280 REM --- NYTT DRAG ---
290 ; "EN GÅNG TILL";
300 REM --- DITT DRAG -----------------
310 INPUT M$ : IF ASC(M$)<49 THEN 290
315 M$=LEFT$(M$,1%) : IF M$<"7" THEN M%
    =VAL(M$) : M%=M%-1% : GOTO 330
320 ; "OTILLÅTET DRAG" : GOTO 290
330 IF B%(M%)=0% THEN 320
340 ; CHR$(12%)'DITT DRAG: 'M$
350 H%=6% : REM --- DIN KALAHA
370 REM --- UTFÖR & MINNS DRAG ---
380 K%=M% : GOSUB 610 : REM -- UTFÖR DR
    AGET
390 E%=0% : IF K%>6% THEN K%=K%-7% : RE
    M -- DATORNS DRAG
400 C%=C%+1% : IF C%<9% THEN F(N%)=F(N%
    )*6%+K%
410 REM --- E=0 => KULOR & SPEL SLUT
420 FOR I%=0% TO 5% : IF B%(I%)<>0% THE
    N 450
430 NEXT I%
440 GOTO 490
450 FOR I%=7% TO 12% : IF B%(I%)<>0% TH
    EN E%=1% : GOTO 490
460 NEXT I%
480 REM --- RITA BRÄDET ------------
490 ; : ; "   ";
500 FOR I%=12% TO 7% STEP -1% : GOSUB 5
    80 : REM ---SKRIVER ANT KULOR
510 NEXT I%
515 ; TAB(30%)N%;C%
520 ; : I%=13% : GOSUB 580
530 ; "                    "; : ; B%(6%)
    : ; "   ";
540 FOR I%=0% TO 5% : GOSUB 580
550 NEXT I%
555 ; TAB(28%)F(N%);
560 ; : ; : RETURN
570 REM --- SKRIVER UT ANT KULOR ---
580 IF B%(I%)<10% THEN ; " "; : REM ---
    FLYTTAR ENTALSSIFFRA HÖGER
590 ; B%(I%); : RETURN
600 REM --- UTFÖR DRAGET --------------
610 P%=B%(M%) : B%(M%)=0%
620 FOR P%=P% TO 1% STEP -1% : M%=M%+1%
    : IF M%>13% THEN M%=M%-14%
630 B%(M%)=B%(M%)+1% : NEXT P%
640 IF B%(M%)=1% IF M%<>6% IF M%<>13% I
    F B%(12%-M%)<>0% THEN 670 : REM SIS
    TA KULAN I TOM GROP
650 RETURN
660 REM ---SISTA KULAN I TOM GROP ---
670 B%(H%)=B%(H%)+B%(12%-M%)+1% : B%(M%
    )=0% : B%(12%-M%)=0% : GOTO 650 : R
    EM ALLT TILL KALAHA
680 REM ------ MITT DRAG --------------
690 D%=-99% : H%=13%
700 FOR I%=0% TO 13% : G%(I%)=B%(I%) :
    NEXT I%
710 FOR J%=7% TO 12% : IF B%(J%)=0% THE
    N 860
720 Q%=0% : M%=J% : GOSUB 610
730 REM ---UTVÄRDERAR SPELARENS MOTDRAG
740 FOR I%=0% TO 5% : IF B%(I%)=0% THEN
    790
750 L%=B%(I%)+I% : R%=0%
760 IF L%>13% THEN L%=L%-14% : R%=1% :
    GOTO 760
770 IF B%(L%)=0% THEN IF L%<>6% THEN IF
    L%<>13% THEN R%=B%(12%-L%)+R%
780 IF R%>Q% THEN Q%=R% : REM --- SPELA
    RENS BÄSTA MOTDRAG :      Q SPELARE
    NS POÄNG
790 NEXT I%
800 Q%=B%(13%)-B%(6%)-Q% : IF C%>8% THE
    N 840 : REM ---Q ÄR NU DATORNS BÄST
    A POÄNG
810 K%=J% : IF K%>6% THEN K%=K%-7%
820 FOR I%=0% TO N%-1% : IF F(N%)*6%+K%
    =INT(F(I%)/6^(7%-C%)) THEN Q%=Q%-2%
830 NEXT I%
840 FOR I%=0% TO 13% : B%(I%)=G%(I%) :
    NEXT I%
850 IF Q%>=D% THEN A%=J% : D%=Q%
860 NEXT J%
870 M%=A% : ; A%-6% : GOTO 380
890 REM --- SPARA INLÄRDA SPEL ------
900 PREPARE "KALAHA.DAT" ASFILE 1%
910 ; §1%,N%
920 FOR I%=0% TO N%-1%
930 ; §1%,F(I%)
940 NEXT I%
950 CLOSE 1%
960 ; CHR$(12%)CUR(10%,10%)"TACK FÖR I
    DAG"
970 END
1000 REM --- REGLERNA -------------
1010 ; CHR$(12%) : ; 'KALAHA ÄR ETT URGA
     MMALT ORIENTALISKT'
1020 ; 'SPEL. SPELPLANEN SER UT SÅ HÄR'
     : ;
1030 ; '    13 12 11 10  9  8'
1040 ; ' 14                  7'
1050 ; '     1  2  3  4  5  6' : ;
1060 ; 'VID VARJE SIFFRA FINNS EN GROP.'
1070 ; '1-6 ÄR DINA GROPAR, 8-13 DATORNS
     '
1080 ; '(SOM KOMMER ATT KALLAS 1-6 I SPE
     LET)' : ;
1090 ; 'VID SPELETS START FINNS 3 KULOR
     I VARJE'
2000 ; 'GROP. GROPENS NUMMER ÄR DITT DRA
     G.' : ;
2005 ; ' 7 ÄR DIN KALAHA, 14 ÄR DATORNS'
2010 ; 'DET GÄLLER ATT FÅ SÅ MÅNGA KULOR
     I SIN'
2020 ; 'KALAHA SOM MÖJLIGT.'
2030 ; : ; '(Tryck RETURN)'; : GET W$ :
     ; CHR$(12%)
2100 ; : ; 'MAN DRAR VÄXELVIS GENOM ATT
     TA ALLA'
2110 ; 'KULOR I NÅGON (EGEN) GROP OCH LÄ
     GGA EN'
2120 ; 'I VARJE GROP (INKL KALAHA) MAN P
     ASSERAR'
2125 ; 'PÅ VÄG MOTURS.'
2130 ; : ; 'HAMNAR DEN SISTA KULAN ENSAM
     I EN GROP'
2135 ; 'OCH DET FINNS KULOR I GROPEN MIT
     TEMOT,'
2140 ; 'TAR MAN BÅDE DEN OCH KULORNA I G
     ROPEN'
2150 ; 'MITTEMOT OCH LÄGGER I EGEN KALAH
     A.' : ;
2160 ; 'OM MAN I FÖRSTA DRAGET LÄGGER SI
     STA'
2170 ; 'KULAN I EGEN KALAHA, FÅR MAN ETT
     DRAG'
2180 ; 'EXTRA.'
2190 ; : ; 'SPELET ÄR SLUT NÄR EN SPELAR
     E HAR ALLA'
2200 ; 'GROPAR TOMMA.'
2210 ; : ; '(Tryck RETURN för att börja)
     '
2220 ; '(      U för att se reglerna ige
     n)'; : GET W$
2230 IF W$='U' OR W$='u' GOTO 1000 ELSE
     RETURN
```

Hej.

Här några småtips:
Grafiken tar inte alltid, åtminstone inte på min maskin, på sista raden. Det ordnar du med POKE 32720,23. Så här t. ex.:
FOR I=0 TO 23 : ; CHR$(151) : NEXT I :
POKE 32720,23
(Om du lägger ";" efter utskriften,
"; CHR$(151);", så fungerar det.
Reds. kom.)
Om du rensar skärmen många gånger i ett program, så kan du starta med t ex
A$=CHR$(12) : ; A$
Sedan räcker det att skriva A$. Gäller naturligtvis också för 151 eller 23 och 135 m fl.
Varning: Förmodligen har du någon gång råkat kapa en programrad vid ED-användning. Normalt förlorar du lite text eller får ERR-någonting. Men akta dig för att kapa DIM-satser. Pröva:
10 DIM A$=100
Tag ED 10 och kapa med RETURN efter 10 DIM A$, så får du se. Retfullt om du inte hunnit spara programmet.

Tabeller

Hur enkelt få ental, tiotal och så vidare under varann? Låt A eller A% vara den variabel som skall skrivas. Om A% är ett tal >=0 kan du skriva:
; TAB(10+(A%>9))A%
om A är ett flyttal fungerar:
; TAB(10+(a>=10))A
d v s om A<100. Annars är det bara att fortsätta:
; TAB(10+(A>=10)+(A>=10- 0)+(A>=1000))A
och A kan ligga mellan 0 och 9999.99. Ett annat sätt är med LOG10(A). T ex för heltal större än noll:
; TAB(10-LOG10(A+1)).
För negativa tal och decimaltal blir det värre. Då är strängar bättre. Pröva med det här programmet där raderna 30, 40 och 50 är aktuella.

```
10 FOR R=1 TO 20
20 A=RND*100-50
30 A$=NUM$(A)
40 A$=ADD$(A$,'0',2)
50 ; TAB(20-LEN(A$))A$
60 NEXT R
70 GET G$
```

Rustan Ligander

Bäste Tad

Som "profes
dvs ingenjör
floppy!) för
dessutom fe
har avancera
en kombinatio
av ABC80-kl
vilja att ABC
som värkt fr

1. Väldigt
speciellt dom
jobbet, har
kassettspelar
en hel del
modem (se m
bl a kunna h
och hämta oc
Problemet är
program fo
T80PRT som
kan vi kans
på modemet)
kassettspela
kan hämtas
har CASMIN
man bara h
ning bara p
CASMINI put
två gånger?
nästa numm

2. I senaste
rades ett tä
för kommur
modem. Klu
detta till C
eller ändra
är möjligt. D
av alla anp
(som ju ocks
heter) när a
datorn elle
tar det int
i anspråk!
Det vore
ring sådan
modem me
värdefulla
för många
om dem.

3. Det hän
program få
bara kan st
urkoppling.
pade progr
har utvecl
kassett. (Fö
sker nog
-förbättring
ning gradvi
och fördöm
nödtvungna
Nu har ju
efter RESE
man förargl
i minnet?
handböcker
dess progr
klar anvisn
törer på
av ABC-bl
programm
X skulle v
som längd
aldrig fått
fungerade
Senaste nr
"...det finn
65060,0:GO
oss amatö
vi ska gör
Så några
bladet och
```
```

Bäste Tad

Som "professionell amatör" på minidator, dvs ingenjör som använder min ABC80 (utan floppy!) för att lösa arbetsuppgifter (och dessutom för nöjes skull) men som inte har avancerade programmeringskunskaper - en kombination som nog gäller väldigt många av ABC80-klubbmedlemmarna - skulle jag vilja att ABC-bladet behandlar några frågor som värkt fram, och verkligen värkt!

1. Väldigt många av oss, och jag gissar speciellt dom som inte har sin ABC80 genom jobbet, har inte floppy utan nöjer sig med kassettspelaren (bl a av kostnadsskäl). Men en hel del har, liksom jag, skaffat sig modem (se medlemsregistret) för att därmed bl a kunna hålla kontakt med klubbens dator - hämta och ev sända intressanta program. Problemet är att för att hämta (eller sända) program fordras dels Metric's program T80PRT som kostar en bra slant (OK, det kan vi kanske klara som tillägg till priset på modemet) men därtill för oss med enbart kassettspelare programmet CASMINI som kan hämtas från klubbdatorn om man redan har CASMINI (!) eller floppy men inte om man bara har kassett. Varför denna inriktning bara på floppyägare? Varför har inte CASMINI publicerats än men ABCMINI redan två gånger? Bättring efterlyses redan till nästa nummer!

2. I senaste numret av ABC-bladet publicerades ett tämligen kort program från Skåne för kommunikation mellan två ABC80 via modem. Klubben bör snarast låta anpassa detta till Gunnar Tidners ABC80 monitor eller ändra Monitor så att sådan anpassning är möjligt. Det är få av oss som har behov av alla anpassningsmöjligheter hos T80PRT (som ju också tycks ha sina allvarliga svagheter) när användningen begränsas till klubbdatorn eller till andra ABC80. Dessutom tar det inte så stor del av våra 16k byte i anspråk!

Det vore också önskvärt med en modifiering sådan att kassettägare med tvåvägsmodem men utan floppy kan sända sina värdefulla program till klubben till glädje för många medlemmar som nu går miste om dem.

3. Det händer då och då att ett utarbetat program får datorn att låsa sig så att det bara kan stoppas genom RESET eller total urkoppling. Man tappar då hela det inknappade programmet eller den del som man har utvecklat men inte hunnit lagra på kassett. (För alla oss som inte har printer sker nog programutveckling och speciellt -förbättring eller -utvidgning i stor utsträckning gradvis på det av proffsen föraktade och fördömda sättet på skärmen med dess nödtvungna nackdelar.)

Nu har ju inte programmet helt försvunnit efter RESET. Men hur komma åt det som man förargligt nog vet finns kvar någonstans i minnet? Ingenstans i alla de utmärkta handböcker som publicerats om ABC80 och dess programmering har jag hittat någon klar anvisning lämpad för oss normala amatörer på programmering. I första numret av ABC-bladet stod bland de uppskattade programmeringstipsen POKE 49152,X där X skulle vara 2-225 men "helst lika stort som längden av första raden". Det har jag aldrig fått att fungera trots provning på fungerade program enligt anvisningen. I det senaste nr 1/81 finns en antydan på sid 27 "...det finns ju alltid möjlighet att slå POKE 65060,0:GOTO70...". Skriv en för många av oss amatörer angelägen beskrivning av **hur** vi ska göra och **varför**!

Så några allmänna reflexioner om ABC-bladet och klubben. Fortsätt för allt i världen att trycka nya program av skiftande typer och författare. Det är där vi andra lär oss programmeringsknep som inte står i handböcker. Men publicerade program kunde ofta åtföljas av en kort förklaring varför man gjort si eller så. Att knipa till om sina knep för inte klubbmedlemmarnas kunnande framåt!

I linje härmed bör klubben snarast utarbeta en kompletterande bruksanvisning till ABC80. Det finns några få tryckfel där och den senare hälften borde nog förses med en egen bruksanvisning för den nye ABC80-ägaren, men det finns också en hel del enkla nyttigheter som **inte** står där men som proffsen väl känner till (t ex att man ofta kan sloppa THEN) och det finns rätt många hjälprutiner som borde finnas samlade (t ex hur man gör en subrutin för rötter eller för trigonometriska funktioner med fler än sex siffror där den enkla ASCII-aritmetiken slår slint). Många har väl sin bruksanvisning full med anteckningar och tillägg. En mycket angelägen uppgift som kunde lösas med successivt utsända lösblad efter en första utgåva. Tillsatt genast en arbetsgrupp.

På den utvidgade Basic som det finns plats reserverad för i våra ABC80-datorer får vi tydligen vänta förgäves eftersom det kunde störa lanseringen av den nya (och mycket dyrare) ABC800. Men klubben bör med trycket från nu befintliga 15000 ABC80 inte utan vidare acceptera detta utan ta initiativet till ändring. Basic II för ABC800 bör kunna monteras in i befintliga ABC80, kanske tillsammans med anpassning till ev ändrade interna adresser. I annat fall utmanas härmed våra duktiga och okonventionella småföretag på programmakarområdet att i samarbete med klubben ta saken i egna händer. Vi är dock många klubbmedlemmar och andra ABC80-ägare som inte vill sidsteppas av de företag vi köpt våra datorer av.

Din tillgivne
Göran Tengner

Det är ett flertal intressanta frågor som Göran Tengner tar upp i sitt brev. Vad gäller önskemålen om kommunikationsprogram för ABC80-ägare utan tillgång till ett flexskiveminne ber vi på redaktionen er läsare av ABC-bladet om hjälp. Tänk till, programmera och skicka in lösningsförslag. Vi vet att intresset för datakommunikation är växande över hela fältet av ABC80-ägare.

Vad gäller CASMINI så publicerar vi det programmet på annan plats i tidningen (se programsidorna).

Sedan har vi det här med hur programrader lagras i datorns primärminne. Som Göran riktigt påpekar är det möjligt att med Poke-kommandon återfå ett program som "gått förlorat" i samband med Reset. Det är i den första minnespositionen av använd programarea i primärminnet denna manipulering ska ske. Beroende på var vi har valt att lägga primärminnets golv - i ABC80 kallat BOFA - ska olika Poke-adresser användas. Med 16k byte RAM blir adressen 49152, med 32k byte RAM blir den 32768 osv.

Men vad ska vi lägga för information i den aktuella minnescellen? Totalt finns ju 256 alternativ (0-255). Jo, programmets första minnescell ska innehålla det numeriska värde som svarar mot första programradens antal byte i primärminnet. För att kunna räkna ut detta kan följande enkla exempel vara på sin plats: Låt oss studera hur programraden

10 REM * DETTA ÄR EN KOMMENTARRAD *

ter sig i primärminnet. Förutom de olika ASCII-tecknen läggs nämligen in kod för dels kommandot REM och radnummer. Det senare representeras med två byte vilket ger totalt 256x256 möjliga adresser. Vidare tar REM-kommandot två byte. Slutligen behövs ett radslutmärke (vagnretur) vilket tar ett byte i anspråk och ett byte åtgår i början av hela programraden för att ange hur många kommandon som ingår i den aktuella raden. Vårt exempel tar med alla tecken upp 34 minnesceller, vars innehåll i decimal form blir enligt följande: 34,10,0, 134,132,42,32,68,69,84,84,65,32,91,82,32,69,78, 32,75,79,77,77,69,78,84,65,82,82,65,68,32,42,13.

Radnummer i Basic framgår av minnesinnehållet i cellerna 2 och 3 och beräknas med givna värden 0x256+10=10. De därpå följande numeriska värdena 134,132 är den decimala koden för Basic-kommandot REM. Därefter som väntat de olika tecknens ASCII-värde med mellanslag konsekvent representerade av det decimala värdet 32. Slutligen avslutas den aktuella programraden med 13 vilket är ASCII-värdet för radslut=vagnretur. Så långt hur programrader byggs upp i primärminnet.

Nu till problemet med hur en "skenande" program ska kunna avsökas genom att starta vid olika programrader. Som Göran påpekar är det med programkommandot Poke 65060,0: GOTO XX som konstgreppet utförs. Lägg märke till att detta kommando måste skrivas in i ett program, dvs föregås av ett radnummer. Vad vi gör med detta kommando är att påverka minnesinnehållet i datorns sk systemvariabelarea, närmare bestämt den cell i vilken datorns arbetsstatus (runmode) anges. Logisk nolla anger att programexekvering pågår medan logisk etta anger vilande funktion. För er som vill prova olika ingångar i era program är det således bara att som första rad lägga in exempelvis 10 POKE 65060,0 :GOTO XX.

När det gäller proffsens sätt att programmera ABC80, t ex att slopa THEN i sk IF-satser finns en hel del att säga. Visserligen sparar "smartkodaren" lite plats i sitt datorminne, men samtidigt bryter han/hon mot allt vad som heter ANSI-Basic, dvs den standardnomenklatur för programmering i Basic som blivit internationellt fastställd. Av skadan vis kan jag konstatera att det är bra mycket enklare att hålla sig till skolboken än att försöka hålla reda på alla olika datorers dialekter och "datorslang".

När det gäller proffsens rent allmänna trix, t ex smarta algoritmer o dyl, är det naturligtvis redaktionens förhoppning och målsättning att så pass rikhaltiga förklaringar som möjligt ska finnas till de insända programexemplen. Här syndas tyvärr fortfarande en hel del varför således alla bidragsgivare uppmanas att på bästa sätt försöka bifoga en utförlig programförklaring till insända program. Beträffande Göran's förslag om att tillsätta en arbetsgrupp för just programmeringstips och rättelser av fel i befintliga instruktionsböcker lovar redaktionen att återkomma vid ett senare tillfälle.

Slutligen tar Göran upp frågan om en utökad Basic (24k byte) till ABC80. Det här är en fråga som inom ABC-klubben stötts och blötts åtskilliga gånger med såväl medlemmar som Luxor, Diab, ScandiaMetric och Sattco. Något rakt svar till följd av dessa diskussioner kan vi dessvärre inte ge. Men helt klart är att problemet inte är så enkelt som att enbart "plocka fram ett nytt PROM". Det finns såväl ekonomiska som upphovsrättsliga aspekter som komplicerar bilden. Tills dessa är lösta lär vi ABC80-ägare snällt få vänta på en ny och utökad version av ABC80-Basic. Det är dock ABC-bladets förhoppning att maskinvaruleverantörerna ska känna ett fortsatt ansvar för de ca 15000 ABC80-ägarnas datorinstallationer. Den som lever får väl se!

red

A B C-bladet Samlingsnummer
1981 : 2 sidan 12

ABC-Monitor flitigt utnyttjad

I slutet av januari kom ABC-Monitor på tel 08-80 15 23 igång i klubblokalen. Efter någon dag upptäcktes en bug i DOS-promet till vår nyförvärvade DataDisc 88. Det byttes snabbt ut och utrustningen har därefter varit utsatt för ett verkligt hårdtest. DataDisc 88 har alltsedan dess varit i kontinuerlig drift (24 tim per dygn) och bestått provet utan anmärkning.

Trots att de program som ingick i mitt monitor-paket borde vara väl uttestade och avlusade genom att systemet använts av många i mer än ett halvt års tid, visade det sig att det obemannade systemet spårade ut titt och tätt när det öppnades r allmänt bruk. Då var det inget annat att göra en att någon fick bege sig till klubblokalen och starta upp på nytt.

Vissa programfel har relativt lätt kunnat avhjälpas men andra har varit mer besvärliga att komma till rätta med eftersom det inte går att ta hand om alla fel med NERRORGOTO. Förutom rena överförings- l (det verkar som akustiska modem har större benägenhet att ge sådana) har för långa rader vid inskrivning av text varit en av de vanligaste felkällorna. Någon gång kan det ha berott på strömavbrott att systemet fallit ur.

I systemet finns en option med utbytbara meddelandetexter som skrivs ut när man får kontakt med systemet, ett kort meddelande före och ett efter inloggning. Dessa texter har utnyttjas för att varna för sånt som kan orsaka fel samt för att annonsera klubbmöten och andra nyheter.

Sedan mitten av mars har vi genom särskild hårdvara fått en tillfredsställande lösning på problemet att starta upp systemet när det faller ur. Vår DataDisc 88 har försetts med en Autostart från EMH-Data AB. Det är EMH-Data AB i Eskilstuna tel 016-14 38 11, Anders Isaksson, som tagit fram och säljer Autostart för DataDisc 80, 82, 84, 86 och 88 samt Luxor ABC för 950 kr per st. Vidare har vi kopplat in en särskild "burk" som ger RESET-signal ' ABC-bussen (och därmed initierar auto- arten) när bärvågen försvinner mer än 2 sek. Burken har tagits fram av SELIC AB (f d Vektorprodukter tel 031-23 91 96) som med denna burk förverkligat ett hopkok av ideer från flera personer. Vi räknar med att publicera ett schema på burken i nästa nummer.

Monitor-systemet är fn mycket hårt utnyttjat. Det kan vara mycket svårt att komma fram på telefon. Så fort någon loggat ut ringer det strax på nytt och nästa användare loggar in. Vi har därför sett oss nödsakade att införa en strängare behörighetskontroll av de som använder systemet.

Inom kort sänds de nya medlemskorten ut till alla som betalt 1981 års avgift. I samband härmed kommer inloggningsrutinen att ändras så att endast klubbens medlemmar kan utnyttja systemet. Varje medlem tilldelas ett inloggningsnummer och password som skall användas tillsammans med medlemsnumret.

Inloggning går till så att man först frågas efter loggnr, sedan efter medlemsnummer, därefter pasword. Om det inte stämmer kan man göra ytterliggare 3 försök innan man kastas ut av systemet.

Glöm inte att slå av modemet efter avslutad körning annars kan telefonförbindelsen stå kvar uppkopplad (kan bli dyrt) och systemet spärras för andra användare.

Vid körning mot ABC-Monitor användes något av programmen ABCMINI, ABCTRANS, LOGIN eller CASMINI. ABCMINI har publicerats i nr 1 och nr 4 årgång 1980 av ABC-bladet. På begäran publiseras CASMINI som är en specialversion för kassett av ABCMINI. Samtliga dessa program kommer att finnas på den första programkassett som sänds till alla medlemmar (som betalt 1981 års avgift). Med ABCMINI och CASMINI kan man endast hämta program (med kommandot GETFIL) ej sända till Monitorn. Det är inte någon avsiktlig diskriminering mot alla dem som endast har kassettminne, att det ännu ej släppts ut något bra program för att sända fil från kassett. Det sammanhänger med vissa tekniska svårigheter, när man läser från fil på kassett sändes skräp ut på V24-utgången som stör modemkommunikationen. Jag har gjort ett program CASSEND.BAS som hjälpligt klarar att sända från kassett till Monitorn men jag anser det inte vara någon lyckad lösning. Programmet kan av intresserade hämtas från klubb-Monitorn, det ligger fn på drive 1 men kan hämtas av de som är intresserade att prova. Plats för förbättringar!

De program och textfiler som ligger lagrade på 8-tums disketten i drive 0 tar fn upp 42% av det tillgängliga utrymmet och motsvarar mer än 5 1/2 fullt utnyttjade vanliga 5-tums disketter med enkel täthet. De är alltså en ordentlig kapacitet som finns i klubbens program- och informationsbank! För att snabbare kunna se tillskotten till programbanken har LIB-programet ändrats så att filerna visas i baklänges ordning mot normalt dvs. filer med högt filnummer visas för filer med lågt. Vill man inte läsa hela lib så kan man avbryta med CTRL-D <RETURN>.

De Monitor-system som är tillgängliga för klubbmedlemmar på andra orter i landet, se nr 1 av ABC-bladet, betjänas av enskilda klubbmedlemmar som välvilligt ställt upp sin egen utrustning. Du som har förmånen att köra mot dessa system bör respektera dessa medlemmars behov att ibland utnyttja sin egen utrusning för andra uppgifter.

Det finns säkert behov av Monitor-system på flera platser i landet. Medlemmar som är villiga att sätta upp utrusning dit andra klubbmedlemmar kan få ringa kan kontakta undertecknad.

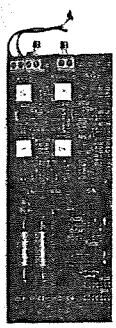
Gunnar Tidner
tel 08-758 35 74

```
 10 REM CASMINI filöverföring till egen
    ABC-80
 15 REM Program ABCMINI av Gunnar Tidner
 17 REM Modifiering till kasetthantering
    av Thomas Bergstam 1980-08-23
 19 REM Smärre mod 1981-02-05 G Tidner
 20 OPEN "V24:KB.1" ASFILE 1% : L%=1%
 30 Z$="/*" : REM Filslutmärke
 40 DIM B$=255%,D1$=255%,D2$=255%
 50 ; "          *** Half Duplex ***"
 60 INPUT §L%,A$
 70 ; "Överföring av fil från monitorn?"
 80 ; "Tryck tangent!"; : GET C$ : ; C$
 90 ; "Filnamn i MONITORN:"; : INPUT F1$
100 ; "Eget filnamn:"; : INPUT F2$
110 ; §L%,"Hejsan"
120 INPUT §L%,B$
130 IF B$<>"SYNK" THEN 110
140 FOR T%=1% TO 100% : NEXT T%
150 ; §L%,"SYNKSVAR"
160 ; §L%,F1$
170 INPUTLINE §L%,B$ : B$=LEFT$(B$,LEN(B
    $)-2%)
180 IF B$<>"OPENED" THEN ; B$ : FOR T%=1
    % TO 3000% : NEXT T% : GOTO 60
190 D0$=F2$ : D0%=3% : GOSUB 410
200 FOR I%=1% TO 10000%
205 FOR T=1 TO 100 : NEXT T
210 ; §L%,'>'
220 INPUTLINE §L%,B$
230 IF LEFT$(B$,2%)=Z$ THEN 270
240 D2$=B$ : D0%=3% : GOSUB 330
250 ; B$;
260 NEXT I%
270 D0%=3% : GOSUB 510
280 ; CHR$(7%)"Överföringen klar!"
290 ; "Filnamn: ";F2$;" antal rader=";I%
    -1%
300 FOR T%=1% TO 2000% : NEXT T%
310 ; §L%,"Tack!"
320 GOTO 60
330 IF LEN(D1$)+LEN(D2$)>253% THEN 360
340 D1$=D1$+D2$+CHR$(13%)
350 GOTO 400
360 GOSUB 570
370 GOSUB 450
380 OUT 58%,INP(58%) AND 223%
390 D1$=D2$+CHR$(13%)
400 RETURN
410 PREPARE "CAS:"+D0$ ASFILE D0%
420 OUT 58%,INP(58%) AND 223%
430 D1%=PEEK(65021%)
440 RETURN
450 PRINT §D0%,D1$
460 IF PEEK(65021%)<>D1% THEN GOSUB 480
470 RETURN
480 FOR Z%=0% TO 1500% : NEXT Z%
490 D1%=PEEK(65021%)
500 RETURN
510 GOSUB 570
520 GOSUB 450
530 IF D1%<>PEEK(65021%) THEN 550
540 PRINT §D0%,CHR$(0%); : GOTO 530
550 CLOSE D0%
560 RETURN
570 OUT 58%,INP(58%) OR 32%
580 FOR Z%=0% TO 2000% : NEXT Z%
590 RETURN
```

NYA TILLBE

Hos Mikronil
och digitala
enheter till
Digitalkor
klara för dire
på ABC80.
bitar ut, ka
bitar in kall.
Grundvari.
2x8 bitar in
och 2x8 bita
Digitalkor
ende även 8
från UT0. I
8 optoisolera
Digitalkor
i övrigt lika
Digitalko
ingångar och
Analogkc
av 2 analoga
gar. Maxima.
Man kan v
för in- och

För ytter
MIKRONIK



ETT NYTT

Det nya mi
passar direk
användarmi
Kortet ar
i ABC80's t
ningar till
kort kräves
Expansic
miska RAM
PROM för
medföljer
ändra tillba
monterings
Kortet
och kostar
Ytterliga
kan fås fr
tfn 090-11
08-7182640

APPLIKATIONSEXEMPEL

ABC-80 som filmtextmaskin.

Som säkert läsarna har märkt, har videon översvämmat vårt land, inte minst tack vare publiciteten i det största videoföretaget SvTv.

Utbudet från de privata videoföretagen består till övervägande del av utländsk film. För att kunna sälja den i Sverige krävs att den antingen dubbas till svenska eller textas. Dubbningen är mycket kostsam och kräver ett uppbåd av artister. Metoden används huvudsakligen till barnfilm eftersom barnen inte kan läsa. Textning däremot är en relativt enkel metod.

Till detta ändamål lämpar sig ABC-80, med våra extra tillsatser, alldeles utmärkt. Vår textmaskin är ca. 4 gånger billigare än andra fabrikat. Trots det lägre priset får man en dator på köpet som kan användas bokföring eller annat. Vi kan dessutom leverera vår maskin med mycket kort leveranstid på grund av att mycket standarddetaljer har använts.

Genom att komplettera och modifiera det textgeneratorkort som finns till ABC-80, samt tillverka en lämplig blandare för bild, -ken och "box" (den svarta ramen runt -ten) kunde vi sätta text till befintliga bilder.

Systemets uppbyggnad framgår av fig 1. Nästa utvecklingsfas var att få ABC-80-datorn att "upphöra som dator" och bli textmaskin. Här ställde Systemkonsult Göran Österman upp och löste en mångfald problem. Textmaskinen är nu mycket lättarbetad både vad gäller in- och utskrift. Utskriften kan ske manuellt från tangentbordet, med fjärrstyrningskabel till den s.k. "V24"-kontakten eller automatiskt med tonstyrning. Om man väljer det automatiska förfarandet, behöver inte personal bindas för passning av textmaskinen. Dessutom finns ett mycket bra och väl använt rättningssystem, varmed hela ord kan bytas eller enkla stavfel korrigeras. Mer om detta på annan plats i detta nr.

Alla repliker lagras på diskett. En enda diskett rymmer ca 1000 repliker. En normal 1 1/2-timmes spelfilm bör därför rymmas på en diskett, en Tafi-film på mindre än en halv eller en ordrik italiensk film kanske kräver två disketter. Lagervolymen är med -ra ord minimal.

Om du sett videofilm har du förmodligen också sett resultatet, i annat fall visas exempel enl bild 2.

Lennart Ståhlberg
TELEVISION SYSTEM AB

I praktiken

Jag fick erbjudande av ITV Television System AB, att utveckla programbiten till den beskrivna filmtextmaskinen. Detta system tänkte jag beskriva och samtidigt försöka ge tipps om problem som kan uppstå.

Systemet kan indelas i två faser. Dels inmatning och redigering av text, och dels utskrift av den lagrade texten.

Inmatning och redigering av text är ju egentligen vanlig ordbehandling, så det vållade inga bekymmer. För att krångla till det lite, ställdes krav på en kapacitet mellan ett- och tvåtusen textsidor. En textsida skulle bestå av 1 till 3 rader om max 35 tecken per rad, alternativt en helsida som kan bestå av upp till 11 rader text.

För att klara dessa funktioner och kapacitetskrav, bestämde jag mig för att lagra textraderna med variabel längd, d.v.s. enbart den del av raden som inehöll text. Avslutas varje rad med en speciell kod, kan radslut i alla fall hittas. Helsida kunde lösas på samma sätt, om en speciell kod inledde helsidestext och en annan kod avslutar den.

Dessa koder måste naturligtvis väljas så att de ej kan förväxlas med den vanliga filmtexten. Vid läsning från en sekvensiell fil kan ju bara tecken med ascii-värde mellan 32 (blank) och 127 (CTRL + <>) användas, vilket inte skulle räcka till. Används däremot direktläsning, kan alla asciikoder mellan 0 och 255 användas.

Med detta lagringsförfarande blev kapaciteten 600 till 1500 textsidor per diskett, så för att klara upp till 2000 sidor användes 2 disketter.

Utskrift av texten sker via ett färgvideo interface-kort som drivs av en programrutin i prom. Färgvideokortet öppnas som en enhet, med namnet "PR:R". Till enheten skrivs sedan all information som ska presenteras på videofilmen, inklusive styrkoder för att redigera texten till önskat format.

Signalen att lägga ut, respektive ta bort en bildsida, tas in via V24-kontakten. Här uppstod det första problemet. Skickades signal för bildväxling "för tätt" och det råkade vara mitt under en inläsning, så tappades den signalen bort. Problemet rättades lätt till, hårdvarumässigt. ABC-80 fick helt enkelt "kvittera ut" signalen genom att skicka en signal tillbaka, först därefter togs den ingående signalen bort.

Nästa problem upptäcktes efter det att 3 felfria videofilmer hade textats. Om så lång tid förflöt mellan signalerna för bildväxling, att motorerna i floppydisken hann stanna, uppstod en fördröjning vid nästa inläsning, eftersom motorerna då först måste startas upp och komma upp i rätt varvtal. Under tiden kunde signal komma både att ta bort och lägga ut nästa textsida.

Problemet löstes genom att räkna upp en heltalsvariabel, under tiden som V24-kontakten kändes av för signal. När räknaren nådde 20.000 lästes den aktuella sektorn ytterligare en gång. På så sätt kunde floppymotorerna hållas i gång, i det oändliga om så skulle behövas.

Nu har systemet brukats i ett halvår och synpunkter på systemet har kommit in. Dessa synpunkter ska nu undersökas och om de är intressanta ska de implementeras i systemet. Om detta en annan gång.

Göran Österman

3M OM VÅR FLEXSKIVETEST

Kommentarer med anledning av diskett-test

I föregående nummer av ABC-bladet utförde en testgrupp en undersökning av ett antal disketter av skilda fabrikat. 10 disketter av varje fabrikat testades.

Teste utfördes på sk minidisketter, dvs disketter av 5.25 tums storlek. I denna test visade det sig att alla fabrikat uppvisade en viss felprocent. SCOTCH disketterna uppvisar här ett högre procenttal fel än genomsnittligt. Detta förvånar oss på 3M. Anledningen till detta är att vi har tillgång till kontinuerliga tester utförda i USA. Dessa tester är inte enbart utförda på SCOTCH disketter, utan även på våra konkurrenters. Resultaten ger en helt annan felfördelning än vad ABC-bladets test visat. Visserligen har dessa kontinuerliga tester givit varierande resultat, men tendensen är att 3M alltid är placerad främst eller bland de främsta diskettfabrikaten.

För att verifiera resultatet och undersöka orsaken har vi begärt att få tillgång till de testade disketterna. Dessa kommer att sändas över till USA för kontrollprovning. Dessutom kommer 20-30 andra disketter från samma sändning att kontrolleras.

Så snart vi får veta resultatet av testerna kommer vi att presentera det för ABC-bladet och då kunna kommentera undersökningen baserat på de fakta vi får från forskningslaboratoriet.

I avvaktan på testresultatet kan vi bara understryka att 3M's egna kontinuerliga tester och kvalitetsresultat avviker från de resultat som ABC-bladets test givit.

Dessutom vill vi framhålla att på 8 tums disketter är 3M utnämnda som referensdiskett. Med referensdiskett menas att SCOTCH av de internationella standardiseringskommisionerna, ANSI/ECMA, är utsedd som normgivande för diskettmedia. Detta betyder att SCOTCH disketter från 3M leder kvalitetskraven för de andra fabrikanterna. Någon referensdiskett på 5.25 tums sidan är ännu icke utsedd, men 3M använder samma tillverkningsmetod och samma oxid på minidisketterna som på 8 tums disketterna.

Vi hoppas alltså kunna återkomma i nästa nummer av ABC-bladet och kommentera testresultatet med de fakta vi då har tillgång till.

3M Svenska AB

Från TRANFOR fick vi följande meddelande:

Beträffande DataDisc 82/84/86 och 88
En förbättrad version av programmet som styr flexskiveenheterna har tagits fram. Bland annat är väntetiden mellan motorstart och läsning/skrivning ökad. Vissa exemplar har medfört problem vid t ex CHAIN av ny programmodul. Detta gäller för enheter med serienummer lägre än 1377.

Den som önskar få denna förbättring sänder in sitt kontrollerkort till TRANFOR DATA AB, Pyramidv. 9c, 171 36 Solna. Kortet modifieras och returneras inom två dagar utan kostnad.

Kontrollerkortet är det kort som har en flatkabel ansluten. OBS! Sänd ej in PROM-kortet som sitter längs till höger i DataDiscen.

Med vänlig hälsning
Tranfor Data AB

```
10 REM §
11 REM §
12 REM §
13 REM §
14 REM §
15 REM §
16 REM §
17 REM §
18 REM §
19 REM §
20 REM §
21 REM §
22 REM § Bibl
23 REM § i er
24 REM §
25 REM § Anro
26 REM § LIS:
27 REM § Önsl
28 REM § dri
29 REM § RUN
30 REM §
31 REM § Om
32 REM § lis
33 REM § anv
34 REM § enh
35 REM §
36 REM § Rut
37 REM § pro
38 REM § gen
39 REM §
40 REM § OPE
41 REM §

42 ONERRORGO
   : CLOSE 1
   HAIN göra
43 POKE 6489
   249%
44 POKE 6374
   %,33%,130
   %,22%,249
45 POKE 6376
   49%,62%,2
   6%,214%,
46 POKE 6377
   4%,40%,4
   1%
47 POKE 6379
   8%,36%,2
   %,1%,253
48 POKE 638
   %,58%,21
   6%,249%,
49 POKE 638
   3%,46%,2
   0%,15%
50 POKE 638
   ,229%,42
   %,229%
51 POKE 638
   ,254%,21
   ,18%
52 POKE 638
   32%,18%,
   244%
53 POKE 638
   49%,225%
   225%,44%
54 POKE 63
   4%,253%
   ,2%,225
55 POKE 63
   ,225%,2
   137%,25
56 POKE 63
   53%,175
   ,18%

57 POKE 63
   %,6%,8%,
   44%
58 POKE 63
   ,24%,25
   24%
59 A$=CALL
60 REM § H
```

NYFIKEN

del 3

av Magnus Lundberg

Headerblockets funktion.

Glädjen och nyttan av att känna till hur filernas HEADER-block är uppbyggda motiverar egentligen inte ansträngningen för en förklaring. Nedan följer en förklaring ändå, blott för att stilla oron över att ha gått miste om något väsentligt. ABC80's filhanteringssystem löser faktiskt dessa problem själv. Man kan ju läsa de flesta filer med RANDOM ACCESS-metoden, trots att de är sekvensiella eller till och med programfiler i diverse olika språk. (Försök med ndfiler!)

För att få överblick över en skivas alla filer kompletterar jag DISKSTAT-programmet med en hard copy funktion. (programmet är publicerat i bladet).

De stora bokstäverna representerar HEADER-blocken, och de små utpekar sektorer använda r data i respektive fil. När alfabetet tar slut börjar man helt enkelt om på A igen. Valet av bokstäver har inget med filerna i övrigt att göra utan skall endast ge överdligheten.

HEADER-blocken pekas ut av bibliotekssektorerna 2,0 - 2,7 (vilka det är lönt att leta i framgår i slutet av sektor 0,6) se för övrigt artikeln i ABC-blad nr 4 1980. Låt oss titta närmare på ett HEADER-block och se vad vi kan få ut av det. Låt oss välja filen G (andra omgången). Det är en fil som är tillräckligt stor (>32 sekt) för att vi skall förstå hur fragmenterade filer kan pekas ut av endast ett HEADER-block. Byte nr 0. anger filnr i en egendomlig ordning. HEX 10,20,30,,, E0,F0,11,21,,, E1, F1,12 osv. För att få dessa i en riktig nummerföljd lägger vi det decimala värdet av denna BYTE i variabeln A% och får filens löpnummer så här

N%=(A% AND 240%)/16%+(A% AND 15%)*15%.

N% får då värdet 1,2,3 osv uppåt. D v s första halvbyten anger en-talen och andra halvbyten anger 15-talen. Definiera en heltalsfunktion för ändamålet!

* Byte nr 1. anger blocknr i filen. Det är noll i HEADERN, 1 i första datablock o s v. Jfr randomläsning där 1:a blocket anropas med 0.(och ej 1)
* Byte nr 2.(3:e) är alltid noll.
* Byte nr 3 är alltid FF.

Från och med byte nr 4 skall vi läsa dem två och två för varje fragment av filen. Den första anger spårnummer och den andra skall delas upp i 3 msb och 5 lsb. De tre mest signifikanta bitarna anger sektornummer och de fem minst signifikanta anger längden på fragmentet -1.

```
TRACK/SECTOR MAP DRIVE 0  FILE SPACE:
          1         2         3
01234567890123456789012345678901234567890
=========================================
  'AbceeGghiJjklmno.qrrsuuuwwxzzzzaabbbc
  'aCaeegghijjklNno.qrSsuuuwwYzzzAaabbbc
  'acDeFgghijjkMnno.qrsTuuVwwyzzzaaabbbc
  'acdefgghijjkmnOo.qrstuuvwXyzzzaaBbbCc
  'BcEefggIijjLmnooQqrsUuuvwxyzzzaabbbcc
  'bceefgHiijjlmnooqRrsuuuvwxyzzzaabbbcc
  'bceefghiijKlmnooqrrsuuuvwxyzzzaabbbcc
  'bceefghiijklmnooqrrsuuuWwxZzzzaabbbcc
=========================================
ccddddfgggggghIkllpn.....................
ccddddEfgggggghikllpn....................
cDddddefgggggghikllp.....................
cdddddefgggggghJkllo.....................
cdddddefgggggghjLlvM.....................
cdddddefgggggghjllzm.....................
cddddde Ggggggghjllpm....................
cddddfFgggggggIKllpN.....................
=========================================
40 FILES, 162 SECTORS FREE IN 2 BLOCKS
Push "RETURN"  (P = hard copy)
```

Den studerade filen G har tydligen två fragment fast dessa kommer i en följd. Ett utpekat fragment kan ju inte bli större än 31 + 1 sektorer med endast 5 bitar som längdangivelse. Dessa parvisa "fragmentpekare" upprepas så länge det finns några fler fragment av filen att utpeka. När hela filen på detta sätt är kartlagd kommer minst en byte med FF som signalerar att nu är det slut. Resten av HEADER-blocket används inte, utan är "skräp". (Där kan Du lagra annat, ex.vis datum el dyl). Man behöver alltså endast läsa in HEADER-blocket i minnet för att få klart för sig hur filen ligger på disketten.

Var kommer nästa fil att hamna? Antag att Du har en skiva med ungefär den struktur som i exemplet ovan och att Du tar bort några filer i början för Du vill spara ett ofta kört program på en snabb plats. Kontrollera att det verkligen blir tillräckligt med sammanhängande utrymme. Gör "SAVE" på Ditt program. Kontrollera sedan var det hamnade.!! Just precis. Inte alls där det var tänkt, utan i slutet. Eller snarare i början på det största sammanhängande utrymmet. En bristande kontroll av utrymmesbehovet alltså, och något att tänka på när man vill reorganisera sina skivor.

Vid prepareringen av en ny fil "hugger" ABC80 för sig 4 hela spår (eller rättare 32 sektorer) i taget, och blir det något över lämnas det tillbaks vid "CLOSE". Så är emellertid inte fallet om Du sparar ett litet program där det förut låg ett stort. Då får Du inte tillbaka något. Har Du komprimerat ett program kan det alltså vara ide att göra "UNSAVE" eller "KILL" på den gamla versionen innan Du sparar den mindre versionen, om Du vill hushålla med sektorerna.

Programlagring.

Skall man spara programmen i .BAC eller .BAS -format? "SAVE"-kommandot (BAC) sparar i kompilerad form. Går snabbare att ladda, men kan inte läsas av textbehandlingsprogram. Man får ogiltiga tecken och kanske för långa rader vid "INPUTLINE". "LIST"-kommandot (BAS) ger lagring i ASCII (text)-format. Blanktecken är packade (tab). Man kan vinna några sektorer vid långa program eftersom sektorerna utnyttjas helt. Medan kompilerade programrader inte delas av då sektorn tar slut, utan sektorn lämnas med spill och ny sektor inleds med ny rad, kan listade program delas av inuti raderna.

Filslutet på sekvensiella filer känns igen på att 6 nollor följt av 03 (ETX) inleder sista datablocket, som för övrigt är likadant som det näst sista (det sista med egentlig information). Filslut på random-filer, BAC-filer och ABS-filer tar inte en hel extra sektor i anspråk.

PS. Du har väl skaffat Dig mina förträffliga analysprogram DDISP och FDISP med flera? Om inte så hör av Dig för omgående leverans!

Om nästa kapitel kommer, så skall det handla om BASIC.

Med vänliga hälsningar

Magnus Lundberg
Nyfors
190 40 Rosersberg
tfn 0760/380 25

```
Sektor 47, 6 (HEX display)
  0:32 00 00 FF 2F DF 33 D1 FF FF FF FF FF FF FF FF '2.../_3Q........'
 16:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
 32:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
 48:FF FF FF FC 00 00 00 00 00 00 00 00 00 00 00 00 '...ö............'
 64:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '................'
 80:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
 96:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
112:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
128:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
144:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
160:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
176:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
192:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
208:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF '................'
224:FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 10 '................'
240:10 04 01 01 01 01 01 00 00 00 00 00 00 00 00 00 '................'
256:''
```

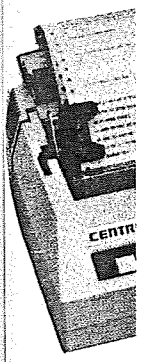
Så här pekas filens beståndsdelar ut av byte nr 4-5 och 6-7 i headern för vår betraktade fil.

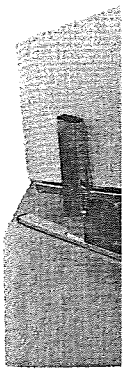
2F DF

0010 1111 1101 1111
spår 47 s 6 längd 31+1

33 D1

0011 0011 1101 0001
spår 51 s 6 längd 17+1





SMARTAID

av Göran Sundqvist

Troligen var jag bland de allra första som kom över "SMARTAID" i början av december 1980. Som summering av mina erfarenheter kan jag helt enkelt säga att den är vanebildande och att den snabbt gjort sig oumbärlig.

Smartaid är alltså ett litet kretskort med en PROM-kapsel som innehåller en expansion av ABC80-basic med de funktioner som man annars saknar. Kretskortet är inneslutet i en krympslang med europakontakter enligt ABC-buss-standard på långsidorna, så att det kan anslutas till busskontakten på tangentbordet. Busskabeln till flexskiveenhet eller expansionslåda kan sedan anslu- till "smartaids" andra kontakt. Det går rs utmärkt att använda "smartaid" utan flexskiveenhet med grundsystemet och enbart kassettbandspelare.

Uppstart sker med ;CALL(29696). Följande nya kommandon i BASIC blir då tillgängliga:

O n - automatisk radnumrering, n=steg.

ex. skriver man bara AUTO följt av RETURN får man steglängden 10. Har man redan ett program i minnet börjar numreringen med sista radnummer+10

START n - uppstart från rad nummer n med alla variabler nollställda.
ex. Vill man hoppa över början på ett program och börja fr.o.m. rad 511, skriver man: START 511.

DEL n,m - tar bort alla rader mellan n och m.

DEL n, - tar bort alla rader fr.o.m. n

DEL ,m - tar bort alla rader från programbörjan till och med rad m
anm. Man har samma konventioner som LIST vid listning av ett program på bildskärmen.

VAR - skriver ut en lista på alla använda ..ibler i ett program.
.. Utskriften sker i den ordning som variablerna står i variabellistan. Vill man ha dem sorterade i bokstavsordning kan det ordnas med att använda nedan beskrivna bildskärmseditor för att tillverka datasatser till ett sorteringsprogram.

Bildskärmseditorn

Den mest användbara funktionen är i alla fall möjligheten att ändra direkt på bildskärmen med 'SMARTAIDS' editeringsfunktion.

Det finns två varianter av 'SMARTAID', den första där man måste göra CTRL Ä varje gång man skulle använda editfunktionen, och den nyare 'SMARTAID 2' där man alltid har möjlighet att flytta markören och hämta tecken från önskad plats på bildskärmen.

Eftersom den gamla versionen troligen bara är aktuell om man har flera ABC 80 och tidigare köpt den gamla versionen och vill ha samma handhavande på alla arbetsplatser, kan vi glömma den och ägna oss åt version 2.

Vill man tömma bildskärmen trycker man CTRL L. Med CTRL Ä tömmer man inmatningsbufferten. Med nedtryckt CTRL-tangent och W flyttas markören uppåt och med Z nedåt, A åt vänster och B åt höger.
Med dessa fyra tangenter flyttas markören till lämpligt textavsnitt på bildskärmen. Med högerpilen flyttas markören över den text man vill ha in i inmatningsbufferten och med RETURN-tangenten verkställes inmatningen och lagras in till BASIC. Har man gjort något fel får man de normala BASIC-utskrifterna. Har man gjort ett invecklat svep över bildskärmen och hämtat text från olika ställen, är det lämpligt att flytta markören till en tom rad och trycka CTRL B, varvid man får en kontrollutskrift av inmatningsbufferten. Detta kan man vara tacksam för om man gjort något fel, eftersom man då bara behöver gå tillbaka till den rad som skrevs ut med CTRL B och rätta den.

Ett litet exempel. Inmatning av:

```
10 A9%(I1%,J1%)=A9%(I1%,J1%)+1
```

görs på följande sätt: Man skriver först 10 A9%(I1%,J1%)= och sedan med CTRL A för man markören tillbaka till A i A9%, med högerpilen för man fram markören till =, skriver in + 1 och trycker RETURN. Ingen risk att man glömt något %-tecken eller fått in något annat fel.

Det behöver inte vara synliga tecken man hämtar in genom att föra markören över skärmen med högerpilen. Med POKE-kommandot kan man lägga ut kontrolltecken i bildminnet och läsa in dessa. Användbara tecken är t.ex. koden för start och avslutning av grafik som kan läggas in direkt i ascii-strängar. Man kan t.o.m. göra en ovanlig form av "listskydd" med koden 8 som betyder backspace och gör att utmatade tecken försvinner vid listning på skärmen. Slutligen, med CNTRL P kan man packa ihop den text som står lagrad till höger om markören.

Enda nackdelen jag hittat med nya SMARTAID2 är att man tydligen använder samma arbetsområde i minnet som SATTCOs printerprom. Man är nämligen tvungen att skriva ut hela parameterföljden vid programlistning, tex LIST PR: VSA25D66.2 varje gång man vill lista ett program på skrivaren. Har man nyss gjort en utskrift och beordringen står kvar på bildskärmen kan man förstås flytta upp markören och hämta in tecken med högerpilen.

Göran Sundqvist
tfn 0756/30310

KOLLISION AV OLIKA PROGRAMRUTINERS INTERNA LAGRING AV DATA I BILDMINNETS OUTNYTTJADE ADRESSAREA

Ett meddelande från OWOCO AB

SMARTAID (R) utnyttjar inget RAM-minne hos ABC80 utom 1 byte som är ett lagringsvärde för AUTOmatiska radnumreringens stegvärde som lagras på ett ställe i bildminnets outnyttjade adressarea.

På grund av att det inte finns någon officiell information om olika rutiners förehavande i bildminnet, uppstår lätt en programkollision. Eftersom vi har upptäckt att SATTCOs printer-PROM utnyttjade samma bildminnesarea som SMARTAID, har vi nu ändrat den på SMARTAID.

Anledningen till att vi utnyttjar bildminnets lediga area är bla för att användaren skall kunna utnyttja den sk poke-arean för egna rutiner samt att undvika störningar med program i arbetsminnet som kanske manipulerar med "stackpekare" och dylikt.

SMARTAID finns i två typer varav den äldre lagrar 1 byte på adress 32767 och den nyare, med automatisk kontroll av ABC80's checksummor 9913, 10042 och 11273, bildskärmsdump och förbättrad bildskärmseditor, lagrar 1 byte på adress 32504. Om någon användare har problem med detta, kan denne vända sig till våra återförsäljare eller direkt till oss, så försäker vi hjälpa till att lösa det.

Vi bifogar följande "programsnutt", som säkert kommer att vara värdefullt för Göran Sundqvist och andra SMARTAID-användare vid initiering av SMARTAID från annat program utan att programkörningen avbrytts. Lägg in "snutten" i början av alla dina program (se upp så det inte finns samma radnummer i ditt program) så får du automatisk initiering av SMARTAID när programmet körs.

Vänliga hälsningar

Bernt Lindgren
OWOCO AB

```
1 POKE 65408%,175%,50%,244%,253%,33%,0%,
  116%,198%,62%,150%,40%,27%,33%,148%,25
  5%,1%
2 POKE 65424%,19%,0%,24%,56%,197%,210%,2
  10%,160%,184%,7%,160%,168%,211%,205%,1
  93%,210%
3 POKE 65440%,212%,193%,201%,196%,169%,1
  3%,10%,214%,246%,50%,255%,127%,198%,10
  6%,237%,71%
4 POKE 65456%,1%,0%,4%,62%,83%,237%,177%
  ,62%,77%,190%,32%,247%,205%,147%,2%,43
  %
5 POKE 65472%,126%,35%,203%,127%,40%,20%
  ,33%,165%,255%,1%,2%,0%,205%,11%,0%,17
  5%
6 POKE 65488%,103%,108%,43%,229%,225%,12
  4%,181%,200%,32%,248%,18%,19%,24%,226%
7 A%=CALL(65408%)
```

Bra vare om alla leverantörer av programprodukter specificerar om program utnyttjar adresser i poke- och bildminnesarean eller om programmet gör "utflykter" till andra adresser än huvudprogrammets normala minnesarea.

red

LOKAL FÖRE

Nils Larsson
lokalförening
(hela Östergö
Nils. Skriv gär
tag i per telef

Musketörgatan
582 30 Linköpi
tfn 013/134565

Hej medlemm

Jag är en AB
av mitt geo
telefonräkning
mig för att
möjligt upplåt
så att även
kommunicera
äger jag bara
citeten är ga
tacksam om ev
med mig och
av program s

Hälsningar
Bengt Lindma
Medlem 831
Tallbacka 9L
971 00 MALM

MASKINNÄRA PROGRAMMERING

Bland klubbmedlemmarna finns ett intresse att kunna hantera maskinspråksprogram. Scandia-Metric's Assemblerskiva kostar c:a 800:-, och det är inte alla som vill spendera så mycket. Med hjälp av Z80 Programming Manual kan man nödtorftigt handassemblera memokoden till hexformat. Sedan skall ju denna kod på något sätt laddas in i datorn. I instruktionsboken finns ett programexempel hur man kan ladda hexkod direkt i minnet. Detta är ett dåligt sätt att hantera program på, särskilt om de skall köras mer än en gång. Man får skriva ett smidigare basicprogram än det i instruktionsboken. Tidigare har ABC-bladet publicerat "HEXMON", här har vi en annan variant där hexkoden lagras i datasatser i programmet.

En vanligt förekommande fråga i samband med maskinnära programmering är frågan om CTRL C. En del säger sig kunna stänga av funktionen genom diverse OUT-kommandon, utan att ...tligen veta vad som händer. Genom två på varandra följande ...-kommandon till tangentbordets kontrollport kan man göra en maskprogrammering av tangentporten. Skicka först ut 255% till kontrollporten (betyder att nästa byte till porten är en 'mask') sedan själva masken. Fundera ut ett lämpligt bit-mönster med hjälp av en ASCII-tabell.

Jag själv har också funderat på hur man stänger kontroll c, och redovisar här ett maskinspråksprogram assemblerat med ...ic's 800-kronors hjälpprogram. Det förutsätter att DOS-buffer ... ledig. Genom att ändra ORG kan man lägga programmet på annat ställe. Tänk dock på att interuptvektorn byggs upp av innehållet i minnesceller med låga adressbyten konstant enligt mitt exempel och höga adressbyten kommer från I-registret i processorn. Den konstiga lilla mellanformen kallas ITH-format, även beskriven i ett tidigare blad. Denna programform är smidig, för den lagras på skivan i ASCII-format och kan behandlas som en vanlig textfil. Den kan sändas via modem till kompisen i Haparanda etc. ITH-formatet skall sedan laddas i RAM-minnet med pokesatserna. Man får inte glömma att lägga dit en rad med CALL för att få igång funktionen.

ML

```
NYFORS DATA SCIENCE     SIDA 1
CTRL C  OFF   81-08-15

 0000           ;      +---------------------------+
 0000           ;      !  LITET DEMOPROGRAM        !
 0000           ;      !  AV  MAGNUS LUNDBERG      !
 0000           ;      !  FÖR ABC-BLADET           !
 0000           ;      !                           !
 0000           ;      !  PROGRAMMET STÄNGER AV    !
 0000           ;      !  KONTOLL C -FUNKTIONEN    !
 0000           ;      !  ÄNDA TILLS RESET TRYCKS  !
 0000           ;      +---------------------------+
 0000           ;
 FA00           CTRLC  EQU   0FA00H   ;DOSBUF 5
 0000           ;
 FA00                  ORG   CTRLC
 FA00 3EFA      INIT   LD    A,.HIGH.CTRLC
 FA02 ED47             LD    I,A        ;SÄTT ITERRUPT-VE
 FA04 C9               RET
 FA05           ;
 FA05           ;
 FA05           ;
 FA05 F5        ENTRY  PUSH  AF         ;DETTA ERSÄTTER
 FA06 3E80             LD    A,80H      ;INTERUPTRUTINEN
 FA08 D306             OUT   (6),A      ;I BASICPROMMET
 FA0A 32F5FD           LD    (0FDF5H),A
 FA0D 3E46             LD    A,46H
 FA0F 32F7FD           LD    (0FDF7H),A
 FA12 F1               POP   AF
 FA13 FB               EI
 FA14 ED4D             RETI
 FA16           ;
 FA16           ;
 FA16           ;      INTERRUPTADRESSERNA
 FA34                  ORG   CTRLC+34H
 FA34 05FA             DEFW  ENTRY      ;TANGENT-ADRESS
 FA36 9405             DEFW  0594H      ;KASSET-ADRESS
 FA38           ;
 0000           ZSLUT END

NYFORS DATA SCIENCE     SIDA 2
CTRL C  OFF   81-08-15
CTRLC  FA00    ENTRY  FA05
INIT   FA00    ZSLUT  FA38

ANTAL FEL = 00
```

```
   REM .            ASSLOAD
 20 REM .      FÖR LADDNING AV ASSEMBLER-
 30 REM .      PROGRAM I HEX.FORM.
 40 REM insänt av .......
 50 DIM A$=16%
 52 ; : ; 'Ett ögonblick!';
 60 ONERRORGOTO 160 : REM DATA SLUT
    X%=1% : READ A$
 80 FOR I%=53000 TO 64500
 90 A1%=ASC(MID$(A$,X%,1%))
100 A2%=ASC(MID$(A$,X%+1%,1%))
110 Z%=(A1%-48%-A1%/65%*7%)*16%+(A2%-48%
    -A2%/65%*7%)
120 POKE I%,Z%
130 X%=X%+2%
140 IF X%>LEN(A$) THEN READ A$ : X%=1%
150 NEXT I%
160 POKE 65052,252,208 : REM HÖJ BOFA
170 ; CHR$(13%);'CHAIN CASTEST   ' : CHA
    IN 'CASTEST'
180 REM HÄR NEDAN SKA HEXKODERNA LIGGA
190 REM I DATASATSER.
200 REM EX :
210 REM DATA EF,12,CD3212,EF80
220 REM MAX 8 BYTE MELLAN KOMMATECKNEN.
230 REM
240 DATA F3,FDE5,FD2A01C0,21FFFF,2203C0,
    00,AF,3200C0,5F,1601,CD94D0,FE16,20F
    7
250 DATA CD92D0,FE16,28F9,FE02,20E7,0000
    ,3EFF,3200C0,210300,DD21FFFB,0E00,DD
    23,CD92D0,19,DD7300,0D,20F4,000000
260 DATA CD92D0,FE03,C46ED0,000000,CD92D
    0,BD,2004,CD92D0,BC,C46ED0,000000,21
    01FC,3A03C0,BE,2005,23,3A04C0
270 DATA BE,C475D0,000000
280 REM BLAD 2
290 DATA 3A00FC,FEFF,2061,000000,CD82D0,
    000000,ED5B01C0,211000,19,EB,ED5301C
    0,FD2A01C0,000000,ED5B01C0,21F0FD
300 DATA 010300,EDB0,3AF0FD,FDBE00,20EC,
    000000,2103FC,010B00,EDB0,000000,21E
    3CF,010200,CD0B00,2103FC,11D77F,0108
    00
310 DATA EDB0,13,010300,EDB0,000000,C34B
    D0,0D,0A,2103FC,ED4B01FC,78,B1,202E,
    7E,000000
320 REM BLAD3
330 DATA FE82,2006,FD360E43,1817,FE83,20
    09,FD360E52,180D,000000,FD360E53,AF,
    3205C0,1809,3EFF,3205C0,23,000000,3A
    05C0
340 DATA A7,2819,7E,FE01,2838,FE00,281E,
    4F,0600,09,CB44,28EF,CD6ED0,1811,000
    000,0606,7E,A7,2008,23,10F9,1819,000
    000
350 REM BLAD 4
360 DATA 2A01FC,23,2203C0,C316CF,000000,
    E1,21FFFF,1806,000000,210000,CD82D0,
    FDE1,FB,C9,000000,3A00FC,FEFF,2805
370 DATA FD360F45,C9,FD361F45,C9,000000,
    FD7E0E,32E57F,FD7E0F,32E87F,C9,00000
    0
380 REM BLAD 5
390 DATA 1608,CB3B,CDD5D0,000000,DB38,FE
    95,CA58D0,06FF,DB3A,E680,280F,10F8,3
    A00C0,A7,28E9,CD6ED0,E1,C37DCF,0676
400 DATA 10FE,CDD5D0,0676,10FE,DB3A,E680
    ,2002,CBFB,15,20C4,7B,C9,000000,DB3A
    ,E6BF,D33A,F640,D33A,C9
```

```
:10FA00003EFAED47C9F53E80D30632F5FD3E46325B
:06FA1000F7FDF1FBED4DD6
:04FA340005FA940536
:00000001FF
```

```
 1 POKE 64000%,62%,250%,237%,71%,201%,245
   %,62%,128%,211%,6%,50%,245%,253%,62%,7
   0%,50%
 2 POKE 64016%,247%,253%,241%,251%,237%,7
   7%
 3 POKE 64052%,5%,250%,148%,5%
 4 Z%=CALL(64000%)
```

LIST efter RESET

POKE 49152,0 (om 32k byte minne: POKE 32768,0) återställer pekaren till lägsta adress för program i arbetsminnet. Nu kan man LISTA programmet, t.ex. på printer. Därmed räddade man det undan förgängelsen, men man kan inte köra det igen, utan måste göra RESET och skriva in det på nytt. Om man i stället för noll kan skriva den siffra som anger antalet bytes i första programraden, räddar man programmet också för körning (se annan artikel i bladet).

RUN ONLY

kan åstadkommas genom programraden POKE 65061,32 i början av programmet.

:T

utan att stanna upp exekveringen A=INP(56)-128 128 tar bort den bit som meddelaratt tangent tryckts ned.

TANGENT NEDTRYCKT

signaleras av att adress 65013 är ettställd. För att ta bort icke önskade nedtryckningar, t.ex. före GET skrivs POKE 65013,0

DELETE M

- (från programrad M och vidare) POKE CALL(3897,M),1%

- N (till men inte med rad N) A=CALL(3897,N) POKE -484,A, SWAP(A)

CONT

(hoppa in på rad X i programmet) POKE 65060,0:GOTO X

TANGENTBORDET STÄNGS AV med OUT 57,3

FD2 STÄNGS AV med Z=INP(7) eller ;INP(7)

BANDSPELAREN STARTAS med OUT 58,32

:R - markören kan flyttas genom
1. ;CUR(R,K)
2. POKE 65011,R,K
3. ;CHR$(27,61,32+R,32+K);
4. A$=CUR(R,K):;A$ - om man behöver denna förflyttning ofta.

INTELLIGENTA (?) KNEP PÅ ABC80

av KaJ & MONTY

Var och en av oss ABC80-ägare har väl med tiden samlat på oss en del knep som kan underlätta programmeringsarbetet. Varför skall vi behålla våra knep för oss själva? Eftersom jag inte funnit något förnuftigt svar på den frågan beskriver jag här kort ett antal praktiska detaljer samt, om möjligt, omnämner den källa ur vilken knepet hämtats. I artikelns första hälft är en stor del hämtade ur tidigare ABC-bladet, men en sammanfattning kan väl inte skada.

Att lista ett program efter RESET eller NEW är möjligt genom POKE 49152,0 (alt. POKE 32768,0 för oss med 32 k), det lärde vi oss i ABC-bladet nr 1/1980 sid 2.

För att få ERASE (DELETE) behöver vi inte vänta på ROMet i ABCV800. Det går, som Gunnar Tidner lärde oss i nr1/1980 sid 4 lika bra att använda de enkla POKE-satserna i sammanfattningen invid. Lika litet behöver vi vänta på CON och RESUME: det går också med POKE 65060,0:GOTO <önskat radnr>, visserligen med risk att hela programmet kollapsar (ABC-bladet 2/1980 sid 14).

Vill vi stänga av CTRL-C av någon anledning (det kan vara bra om man t.ex visar sin ABC80 vid ett allmänt tillfälle och inte vill ha sina grafikprogram avbrutna av illistiga besökare) kan man göra OUT57,3 (ABC-bladet 3/1980 sid 6), varvid man dock stänger av hela tangentbordet. Dels kan man göra det på litet originellare vis med CALL(121), varvid dock diverse ERR-meddelanden kan uppstå. Ibland hjälper det inte ens att ha ONERRORGOTO aktiverad.

Markörhanteringen är ett viktigt kapitel inte minst vid inmatning av data i användarvänliga program. Här erbjuder ABC80 verkligen många möjligheter: för den kräsne finns det nämligen fyra alternativ att flytta markören till position R,K. Om han är lika slätstruken som de flesta andra skriver han rätt och slätt ;CUR(R,K);. Vill han däremot skilja sig från mängden skriver ha POKE 65011,R,K. Är han ännu litet originellare kanske han vill skriva ;CHR$(27,61, 32+R,32+K);. Och anser han alla dessa metoder vara för ordinära skriver han A$= CUR(R,K):;A$. Allvarligt talat är dock den senaste metoden mycket praktisk om samma kursorposition behövs på många ställen i programmet. Vill man spara minne är det dock bäst att först skriva DIM A$=4, varigenom man sparar närmare 100 bytes.

Att bandspelaren startas med OUT 58,32 torde inte vara någon nyhet (bruksanvisningen s 28). Lika föga revolutionerande är väl att POKE 65013,0 hindrar ett onödigt GET - ABC80 glömmer alla tangentnedslag som ägt rum före denna poke-sats.

Har vi bråttom att försätta rutan i blinkmod (om vi exempelvis vill se hela ruta plötsligt blinka samtidigt) är det bäst att begagna det korta 9-bytes maskinspråksprogrammet i sammanfattningen. Om vi selektivt vill försätta delar av rutan i blinkmod går det dock enklare att rätt och slätt använda BASIC-funktionerna i PRITEL-programmet.

Så till programmet PRITEL som, liksom namnet säger, innehåller smått pritel (så säger vi i Finland i alla fall). Det är så gott som helt egen produktion). Programmet är avsett att utgöra en grund för fortsatt programmering, varför REN 10000,1 utförts. PRITEL kan alltså absolut säkert MERGEas utan risk för utsuddning av huvudprogrammets rader ifall man redan hunnit börja detta program innan man laddar in PRITEL.

De olika funktioner som ingår förklaras kortfattat i programmets senare del, men kan behöva utredas närmare.

FNM anger återstående minne och används kontinuerligt vid programutveckling genom ;FNM. Definitionen kan f.ö. förkortas till 1000 DEFFNM = SWAP% (PEEK(65064)-PEEK (65057)) + PEEK(65063)-PEEK(65056)

FNS% är praktiskt vid hantering av inmatade strängar. Om A$ är svaret på en JA/NEJ-fråga ger FNS% svaret -1 om A$ börjar på N eller n, svaret 1 om A$ börjar på J, j eller är CHR$(13) och i övriga fall 0. FNS% används exempelvis enligt följande:

```
10 REM * SPELET BÖRJAR HÄR *
.
.
.
90 ;'ETT SPEL TILL (J/N)?':GET A$: ON
FNS%(A$)+2% GOTO 91,90,10
91 ;'TACK FÖR I DAG!':END
```

FNK och den dårpå följande POKE-satsen är beksrivna redan i ett tidigare ABC-blad (4/1980 s 19). De utgör båda medel för kontroll av ABC80:s klocka. Dock bör vi se upp med FNK såsom vi lärde oss i Avancerad programmering på ABC80, kap 8.5. Använd FNK t.ex enligt följande:

```
100 K=FNK:IF K<>FNK then 100
```

varpå variabeln K säkert har rätt klockslag i sig.

FNR och FNR(R,K) ger båda svaret -1 om R,K befinner sig innanför rutan. En skillnad mellan de båda funktionerna finns dock: i det första fallet är R,K lokala, i det andra globala variabler. Vi bör även minnas att endast en funktion åt gången får användas.

FNM, FNM1 och FNM2 har jag beskrivit i ABC-bladet 3/1980 sid 5. De anger MOD dvs modulo samt MIN- resp MAX-funktionerna. X = FNM!(A,B) tilldelar således X värdet av den mindre av A och B.

POKE FNB1(R,K),FNB2(A$) åstadkommer ett blinkande tecken med ASCII-värdet ASC(A$) i position R,K på rutan. Michael Widenius funktion FNB1 kan även ersättas med Per Lindbergs något snabbare funktion

```
FNA(R,K)=31744+5*(R and 24)+1-28*(R and
7)+K
```

Variabeltilldelning enligt rad 10011 och 10012 kan i vissa fall minska minneskonsumtionen om man använder exempelvis T$ (T som i Töm), G1$ (Grafik på), G0$ (Grafik av) eller P$ (Pip!) extra flitigt. Särskilt om man endast har 16 k minne tillhanda rekommenderas dock att man tar bort de variabeltilldelningar och dimensioneringar man inte uytnyttjar.

Färdiga rutiner för grafikprogram kan avsevärt förenkla nybörjarens grafikprogrammering. I PRITEL-programmet ingår därför även sådana rutiner. Rad 10013 tömmer rutan och försätter den i grafisk mod medan raderna 10014-10017 ritar koordinataxlar som sätter ut X och Y på skalan på lämpliga ställen, dock under förutsättning att skalan av en viss variabel var lagrad i variabeln S.

De sista raderna är grafikdelens kanske mest praktiska del. Denna linjesubrutin är skriven av mig och optimerad med avseende på exekveringstid (OBS! ej med avseende på minneskonsumtion!) av Michael Widenius alias MONTY. För att inte förstöra huvudprogrammets variabler, som ju ofta kan heta I%, har möjligast exotiska variabelnamn valts, eftersom vi nu ändå inte kan få lokala variabler till ABC80. Indata utgörs av R1%,K% samt R%,K% vilka båda markerar ändpunkterna för den linje som skall ritas. Kvickare program med samma resultat betvivlar jag att någon kan åstadkomma, åtminstone utan att ta till maskinspråk!

Hoppas dessa små knep givit dig några ideer då det gäller att knåpa ihop dina egna program och kanske rent av yppa dina egna, allra hemligaste knep i ett senare ABC-bladet.

Kaj S. Arnö/Michael Widenius dvs. KAJ & MONTY

Anm

För att texten skulle bli mera lättläst har många %-tecken efter heltalsvariabler uteslutits. Red. räknar med att läsaren själv kan avgöra när han vill ha heltalsvariabler.

DATUMMÄRKI

I förra numre
förbigående
spår 0, sektor
datummärkning
letteras, och
av programme
stående progra
ständigheterna
Rad 1 i pr
ett självständ
återkommer
programkomm
om att 0,6 i
utan att bl.a.
datering.
Varje gång
på en skiva,
för att regist
skivan blir upp
bearbetas och
blir dock reg
minnet till d
(Så sker förre
När 0,6 ä
över själva p
formas först
där bara ett
vänding. Åter
de, vilket bl
i de tre sis
på det sparac
Programme
begära uppgi
läsas (140),
läser av de tr
teras på bil
märkning. Vil
J och avslut

fall skriver
månad och
sista platser
läses in på
För kontr
ningen och
redan påpeka
att få samn
Nu skulle
märkningen
Det sker n
består av e
DATERA. N
minnet hamr
där den läs
Men nu
FÖRST gör
MERGE <pr
inlästa pro
att finnas
får man mä
t ex 99 9
DATERA
hindrar all
ju begränsa
gen kan u
särskilt me
program ka
Program
ter med e
läser byte
informatio
mets plats
behandlin
den nya h
Observera
göra LOAD
ram>, an
som får b
PDAT på
så att ma

DATUMMÄRKNING

I förra numret nämnde Magnus Lundberg i förbigående att några bytes i slutet av spår 0, sektor 6 (0,6) kunde användas för datummärkning. Uttalandet kan behöva kompletteras, och det sker med publiceringen av programmet DATERA härintill. Nedanstående programkommentarer förklarar omständigheterna.

Rad 1 i programmet är i själva verket ett självständigt program DATUM som vi återkommer till nedan. Sedan kommer en programkommentar som mycket kort talar om att 0,6 inte använder alla sina bytes, utan att bl.a. de tre sista kan brukas för datering.

Varje gång ett nytt program skall sparas på en skiva, sker först en förändring i 0,6 för att registrera att ytterligare en del av s[illegible] blir upptagen. 0,6 läses in i DOSBUF0, be[illegible]etas och läses tillbaka. Innehållet förblir dock registrerat i DOSBUF0 i arbetsminnet till dess något annat skrivs in där. (Så sker förresten överallt i datorn.)

När 0,6 är klar, skall datorn också föra över själva programmet till skivan. Då utförs[illegible]s först huvudsektorn (header record), dä[illegible]ra ett fåtal bytes kommer till användning. Återstående bytes förblir oförändrade, vilket bl.a. innebär att datummärkning i de tre sista hänger med i huvudsektorn på det sparade programmet.

Programmet DATERA fortsätter med att begära uppgift om vilken drive som skall läsas (140), läser in 0,6 i DOSBUFO (360), läser av de tre sista bytes (380) som presenteras på bildskärmen som skivans datummärkning. Vill man behålla den, svarar man J och avslutar programkörningen. I annat fall skriver man in år (två sista siffrorna), månad och dag. Dessa sätts in på de tre sista platserna i DOSBUF0 (210), som sedan läses in på skivan (220).

För kontrollens skull upprepas nu avläsningen och resultatet visas på nytt. Som redan påpekats kommer nu varje nytt program att få samma märkning i sin header.

Nu skulle man ju gärna vilja läsa datummärkningen också på de olika programmen. Det sker med programmet DATUM, som består av en enda rad, insatt som rad 1 i D[illegible]RA. När ett program laddas in i arbetsmi[illegible]t hamnar nämligen headern i DOSBUF0, där den läses med PEEK!

Men nu är det viktigt med ordningen! FÖRST gör man **LOAD DATUM** - sedan MERGE <program>. Det är nämligen senast inlästa programs huvudsektor som kommer att [illegible]innas i DOSBUF0. För säkerhets skull få [illegible]n märka DATUM med något orimligt, t ex 99 9 9.

DATERA-programmet har kontroller som hindrar alldeles orimliga värden. Man kan ju begränsa dem ytterligare. Den här ordningen kan upplevas som alltför omständlig, särskilt med tanke på att endast nyskapade program kan märkas så här.

Programmet PDAT gör motsvarande konster med ett redan befintligt program. Man läser byte 5-6 i headern, som innehåller information om spår och sektor för programmets plats på skivan. Sedan sker motsvarande behandling av de tre sista byten, varpå den nya headern skrivs direkt på skivan.
Observera: Även här måste man FÖRST göra LOAD PDAT och sedan MERGE <program>, annars blir det PDATs huvudsektor som får behandlingen. Bäst att märka även PDAT på något uppseendeväckande sätt så att man blir varnad om man gör f[illegible]

```
  1 ; CHR$(12%)'Programmet datummärkt:'
    PEEK(-2563%)PEEK(-2562%)PEEK(-2561%
    )'  <CR>' : GET W$ : REM ...SAVE DA
    TUM SW
 10 REM ...SAVE DATERA...(ide ML,Nyfors
    )81 11 01 SW
 20 ; CHR$(12%)'DATUMMÄRKNING AV FLEXSK
    IVA'
 30 ; : ; 'Detta program märker byte 25
    3-255'
 40 ; 'i spår 0, sektor 6, med datum'
 45 ; '(ex 81 10 22)'
 50 ; : ; 'År, månad och dag måste vard
    era'
 60 ; 'anges med ett tvåsiffrigt TAL.'
 70 ;
 80 ; 'Eftersom sista raden i 0,6 komme
    r med'
 90 ; 'i programhuvudet när man SKAPAR
    eller'
100 ; 'ÄNDRAR STORLEK på  en fil med SA
    VE/LIST'
110 ; 'får man datummärkning också på'
120 ; 'nyskapade/ändrade program.' : ;
    : ; 'Denna märkning läses med rad 1
    .' : ;
130 ONERRORGOTO 130
140 ; 'DRIVE    ='; : INPUT D1% : IF D1
    %<0% OR D1%>1% 140
150 POKE -767%,D1%
155 GOTO 350
160 ; : ONERRORGOTO 160
170 ; 'DATUM år='; : INPUT A% : GOSUB 1
    000
180 ; '     mån='; : INPUT M% : GOSUB 1
    000
190 ; '     dag='; : INPUT D% : GOSUB 1
    000
210 POKE -2563%,A%,M%,D%
220 Z%=CALL(24675%,6%*32%) : Z%=PEEK(-7
    47%) : IF Z% 900

350 REM ---LÄS MÄRKNING I (0,6)
360 Z%=CALL(24678%,6%*32%) : Z%=PEEK(-7
    47%) : IF Z% 900
370 ; : ; 'Märkning i 0,6: ';
380 ; PEEK(-2563%)PEEK(-2562)PEEK(-2561
    )
390 ; : ; 'Vill du ändra (J/N)'; : INPU
    T W$
400 IF W$='J' OR W$='j' THEN 160
410 ; : ; 'K L A R T'
420 END
900 ; : ; 'DISKFEL'Z% : END
1000 REM ---KONTROLL
1020 IF A%<81% OR A%>99% THEN 170
1030 IF M%<1% OR M%>12% THEN 180
1040 IF D%<1% OR D%>31% THEN 190
1050 RETURN
```

```
   1 GOSUB 64000
64000 REM ...SAVE PDAT...81 11 01 SW
64010 ; CHR$(12%)'DATUMMÄRKNING AV PROGR
      AM I HEADERSEKTORN'
64020 ; : ; 'Detta program märker byte 2
      53-255'
64030 ; 'i headersektorn på skivan med d
      atum'
64040 ; '(ex 81 10 22)'
64050 ; : ; 'År, månad och dag måste var
      dera'
64060 ; 'anges med ett tvåsiffrigt TAL.'
64070 ;
64080 ONERRORGOTO 64080
64090 S%=256%*PEEK(62724%)+(PEEK(62725%)
      AND 224%) : REM ---LÄSER ADRESS TI
      LL HEADER PÅ SKIVAN
64100 GOTO 64170
64110 ; : ONERRORGOTO 64110
64120 ; 'DATUM år='; : INPUT A% : GOSUB
      64260
64130 ; '     mån='; : INPUT M% : GOSUB
      64260
64140 ; '     dag='; : INPUT D% : GOSUB
      64260
64150 POKE -2563%,A%,M%,D%
64160 Z%=CALL(24675%,S%) : Z%=PEEK(-747%
      ) : IF Z% 64250
64170 REM ---LÄS MÄRKNING I HEADER
64180 Z%=CALL(24678%,S%) : Z%=PEEK(-747%
      ) : IF Z% 64250
64190 ; : ; 'Märkning i header: ';
64200 ; PEEK(-2563%)PEEK(-2562)PEEK(-256
      1)
64210 ; : ; 'Vill du ändra (J/N)'; : INP
      UT W$
64220 IF W$='J' OR W$='j' THEN 64110
64230 ; : ; 'K L A R T'
64240 END
64250 ; : ; 'DISKFEL'Z% : END
64260 REM ---KONTROLL
64270 IF A%<81% OR A%>99% THEN 64120
64280 IF M%<1% OR M%>12% THEN 64130
64290 IF D%<1% OR D%>31% THEN 64140
64300 RETURN
```

SOM AMATÖRBOTANIKER I DATORSTACKEN

"Här kommer några tips som passar såväl kassettbitare som floppydumpare." Så börjar ett jättelångt brev från Bernt Figaro i Örebro, som klassificerar sig själv som Botanicum Vaccinicum Myrtillus Datorae - vad nu det kan betyda.

Bernts brev handlar bl a om musik, och eftersom flera andra frågat efter detta sätt att göra musik på ABC80, vill vi ta in ett avsnitt.

Det är ursprungligen Per Lindberg ("the mad programmer") som visat metoden, men Bernt har enligt egen utsago gjort några förenklingar.

Metoden bygger på två maskinprogram, "svarta lådor" som man inte behöver analysera närmare, och på att man vid uppspelningen kortsluter kassettutgången. De två kontakter som sitter på var sin sida om jacket för jordskenan, alltså högst upp, måste förbindas med t ex ett gem.

Melodin bestäms sedan av två variabelvärden, ett för frekvensen F% och ett som anger längden på tonen, alltså den tid den ljuder, T%.

F% bestämmer tonfrekvensen och kan varieras från 16 till 1024, dvs 6 oktaver! Högre tal ger lägre ton. F%=5000 ger paus.

T% måste tas fram genom en viss bearbetning, så att F%*T% hamnar inom ett visst talintervall. Hur detta hänger ihop rent teoretiskt skall vi inte fördjupa oss i. Här räcker det med följande tumregel:
Mycket kort ton: T%=8060%/F%
Dubbelt så lång ton: T%=2%*8060%/F%.
På så sätt kan man få allt längre toner.

När talet man dividerar med F% blir stort (Bernt anger 64480) måste man använda ASCII-räkning fram till slutmomentet, då vi återgår till heltal.

T% och F% får aldrig vara noll, så det är klokt att ha en rutin i programmet som silar bort dessa värden.

Nu gäller det att bestämma vilka toner man vill ha. När man har räknat fram F% och T%, skriver man in dem par och par i DATA-satser.

Bernt har sänt oss flera musikstycken som demonstration, bl a Säkkijärvenpolka och Carnevalen i Venedig, som kanske kommer någon gång via kassett eller monitor till intresserade läsare.

För att göra det hela litet mera åskådligt (?) tar vi med ett av Bernts kortare musikstycken. Även om läsaren inte orkar knappa in programmet OBJ(ekt)8, kan det vara bra att titta på för att förstå gången i det hela.

På rad 2070 definieras M1% som anger startadressen för inläsningen av de "svarta lådorna". Data för dessa följer i 2130-2140. Den första slutar med värdet 201, 9 steg före avslutande 201, som betyder slutet på den andra. Den sista sekvensen gör vissa initieringar, bl a rensar skärmen, öppnar dataporten och maskar bort de bitar man inte behöver. Detta uträttas med CALL(65488%) i rad 2090.

Nu är det klar för spelning (om gemet enligt ovan sitter på plats). Datavärden för melodin finns på 2380-2440 resp 2480-2520. På rad 2180 väljer man första melodin. Sedan läser datorn första dataparet F%,T%. F% pokeas in på M2% (och M2%+1) sedan anropas maskinprogrammet i M1% med CALL(M1%,T%). Man hör en ton! Och så fortsätter det tills programmet (eller data) är slut.

I detta fall ligger M1% uppe i poke-arean, medan M2% (som också definierats på rad 2070) ligger på en icke använd del av bildminnet.

Bernt har provat fram en smart metod att lagra både maskinprogrammet och musikdata inuti själva BASIC-programmet. Det hela går ut på att reservera plats genom ett antal långa strängar, som sedan klurigt fylls med data-värdena. Sedan kan man spola både inläsningsprogrammet och datasatserna och behöver bara den korta "musikaliska avdelningen" tillsammans med sina preparerade stängar. Bernt har lovat att återkomma om den saken.

```
1940 REM OBJEKT8
1950 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
1960 REM $$ /VINNARMUSIK VID SPEL/ $$
1970 REM $$ VER 1.0 / 1981-08-13   $$
1980 REM $$ INSÄNT AV              $$
1990 REM $$ BERNT FIGARO           $$
2000 REM $$ 019-11 29 12           $$
2010 REM $$ HÖGLUNDAGATAN 90       $$
2020 REM $$ S-70368 ÖREBRO         $$
2030 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2040 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2050 REM $$ ADRESS- OCH POKEAREA   $$
2060 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2070 M1%=65408 : M2%=32125 : RESTORE 213
     0
2080 FOR I%=0% TO 47% : READ M% : POKE M
     1%+I%,M% : NEXT I%
2090 X%=CALL(65448%) : REM CALL CLEAR AN
     D  OUT(59)
2100 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2110 REM $$ 48 DATAVÄRDEN FÖR POKE $$
2120 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2130 DATA 245,197,33,125,125,70,35,78,33
     ,0,0,197,125,211,58,11,120,177,188,
     194,143
2140 DATA 255,125,238,64,111,193,197,27,
     122,179,254,0,194,140,255,193,193,2
     41,201,62,7,211,59,205,118,2,201
2150 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2160 REM $$ MUSIKALISKA AVDELNINGEN$$
2170 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2180 RESTORE 2370 : ONERRORGOTO 2180
2190 FOR I%=1% TO 92% : READ F%,T%
2200 IF F%=0% 2230 : REM $$ SÄKERHET MO
     T
2210 IF T%=0% 2230 : REM $$ URSPARNING
2220 POKE M2%,SWAP%(F%),F% : M6%=CALL(M1
     %,T%)
2230 NEXT I%
2250 RESTORE 2480 : GOTO 2190 : REM ÄNDR
     A HÄR TILL VALFRI UTGÅNG
2260 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2270 REM $$ MUSIKDATA I PAR SOM EJ $$
2280 REM $$ FÅR OMKASTAS INBÖRDES  $$
2290 REM $$ MEN PARVIS BAKLÄNGES   $$
2300 REM $$ ELLER PARVIS BLANDAT   $$
2310 REM $$ GÅR BRA !"             $$
2320 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2330 REM
2340 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2350 REM  DATA FÖR CONGRATULATION
2360 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2370 DATA 492,53,480,54,468,56,456,57,44
     4,59,432,60,420,62,408,64,396,66,38
     8,67,376,69,364,72,352,74,340,77

2380 DATA 328,80,316,83,304,86,292,89,28
     0,93,268,97,256,102,244,107,232,112
     ,220,119
2390 DATA 212,123,189,138,168,156,159,33
     0,212,330
2400 DATA 159,165,168,156,159,165,142,36
     9,189,370,189,138,168,156,159,165,1
     26,208,142,184,142,184,159,165,159,
     165
2410 DATA 168,156,189,138,168,156,212,12
     3,159,55,142,61,159,55,168,52,159,5
     5,168,52,189,46,168,52,189,46
2420 DATA 212,123,212,123,189,138,168,15
     6,159,330,212,330,159,165,168,156,1
     59,165,142,369,189,370,189,138,168,
     156
2430 DATA 159,165,126,208,142,184,142,18
     4,159,165,159,165,168,156,189,138,1
     68,156,159,165,159,165,212,61,189,6
     9
2440 DATA 189,92,212,30,189,138,159,165,
     212,61,189,69,189,92,212,30,253,103
     ,318,220,5000,5,168,78,159,165
2450 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2460 REM  DATA FÖR FANFARER
2470 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2480 DATA 425,27,425,27,425,27,357,32,35
     7,32,357,32,284,41,284,41,284,41,21
     2,55,212,55,212,55,284,41
2490 DATA 284,41,284,41,357,32,357,32,35
     7,32,425,27,425,27,425,27,568,41,42
     5,55,357,65,425,55,357,65
2500 DATA 284,82,357,65,284,82,212,294,3
     57,32,357,32,357,32,284,41,284,41,2
     84,41,238,49,238,49,238,49
2510 DATA 178,65,178,65,178,65,238,49,23
     8,49,238,49,284,41,284,41,284,41,35
     7,32,357,32,357,32,477,49
2520 DATA 357,65,284,82,357,65,284,82,23
     8,98,284,82,238,98,178,351
```

80 kolumners l

Många ABC-
önskat sig att
80 tecken pe
vidden på bi
med 40 tecke
större och lät
ligtvis en för
inte har beho

Det finns nu
för den som v
kom CAT me
på ett kort
bussen eller s
rutin på flex
hjälpas att C
och klumpig
som GeJo oc
och GeJo:s s
skiljer sig ifrå

I standardu
minne på 1K
till 32767.
utnyttjar bil
1024. Det fi
många fiffig
för olika sa
om det. Föl
uppstått kolli
jar samma mi
som utnyttjat
AID och Sa
versioner a
att se upp m
när man bygg

I både G
det nya bildr
bordet. Sam
marna ut mo
förändringen
man ändrat
884 och so
0 till 23 i
några ställe
radlängden

Det går
anpassad fö
radlängd 80.
och alltså
skärm med
som också
Det går al
radlängden

i programm
som alltid
skriva Z=
kol till 40

GeJo ut
dels det
på adress
det gamla
och 2K
för 80 k
gör att 40
exakt sa
bilminnet

Jag ha
till 80 k
med den
bästa upp
göras i
med lödk
(kom ihå
gärna öv
densator
in en kon
R806 och
man änd
på att
få bättr
bandspela

80 kolumners bildskärm för ABC-80

Många ABC-80 användare har säkerligen önskat sig att bildskärmen ibland kunde ha 80 tecken per rad som ju är den normala bredden på bildskärmar. Det är klart att med 40 tecken per rad så blir ju tecknen större och lättare läsbara och det är naturligtvis en fördel i sammanhang där man inte har behov av längre rad.

Det finns nu tre olika varianter att tillgå för den som vill ha 80 kol bildskärm. Först kom CAT med sitt system CAT-80, byggt på ett kort som skall anslutas till ABC-bussen eller stoppas in i en FD2. En driv-rutin på flexskiva medföljer. Det kan inte hjälpas att CAT:s lösning verkar något dyr och klumpig i förhållande till den lösning som GeJo och MyAB kommit med. MyAB:s och GeJo:s system har stora likheter men skiljer sig ifråga om bildminnets adressering.

Standardutförandet har ABC-80 ett bildminne på 1K som ligger på adresserna 31744 till 32767. Med 24 rader om 40 tecken utnyttjar bildminnet 960 minnesceller av 1024. Det finns 64 outnyttjade celler som många fiffiga programmerare har utnyttjat för olika saker utan att kanske alltid tala om det. Följden har blivit att det ibland uppstått kollisioner när olika program utnyttjar samma minnesceller. Exempel på program som utnyttjat celler i bildminnet är SMART-AID och Sattco:s printerprom samt tidiga versioner av T80PRT. Därför gäller det att se upp med de förändringar som inträffar när man bygger in 80-kol-minne i sin ABC-80.

I både GeJo:s och MyAB:s system byggs det nya bildminnet för 80 kol in i tangentbordet. Samtidigt byts ett av BASIC-prommarna ut mot två Eprom-kapslar. Den enda förändringen som görs i Basictolken är att man ändrat tabellen som börjar på adress 884 och som innehåller adresserna för rad 0 till 23 i bildminnet, vidare ändras på några ställen värdet på konstanter som styr radlängden 40 resp 80.

Det går till så att den ena kapseln är anpassad för radläng 40 och den andra för radlängd 80. Man växlar mellan dessa kapslar och alltså mellan 40 resp 80 teckens bildskärm med kommandot ;INP(3) resp ;INP(4) som också kan skrivas in i ett program. Det går alltså att skriva program så att radlängden ställs om automatisk mitt inne i programmet. INP(3) och INP(4) är uttryck som alltid får värdet 255. Man kan även skriva Z=INP(3) för att ställa om från 80 kol till 40 kol.

GeJo utnyttjar för det utökade bildminnet det gamla bildminnet dels 1K RAM på adress 23552 till 24575, MyAB utnytjar det gamla bildminnet för 40 kol varianten och 2K RAM på adress 22528 till 24575 för 80 kol bildminnet. MyAB:s utformning gör att 40 kol varianten kommer att behålla exakt samma radadressering och luckor i bilminnet som orginalutförandet.

Jag har låtit bygga om mitt tangentbord till 80 kol bildminne och är mycket nöjd med den tekniska kvaliteten. För att få bästa upplösning skall även vissa ändringar göras i TV-skärmen. Med lite händighet med lödkolv kan man klara av det själv (kom ihåg att dra ut nätsladden och vänta gärna över natten så att högspänningskondensatorn kan laddas ur). Man skall lägga in en kondensator på 330 pF över motståndet R806 och kortsluta induktansen L802. (Medan man ändå är inne och rotar bör man passa på att ta bort kondensatorn C101 för att få bättre kvalitet på ingående signalen från bandspelaren, se ABC-bladet 1981:2 sid 14.)

Att den tekniska kvaliten blir mycket god är en sak men hur var det med programvaran? Skaffar man sig 80 kol bildskärm vill man naturligtvis ha programvara som fullt ut kan utnyttja den "bredare" skärmen. Program skrivna i basic som skriver på skärmen med vanliga printsatser med eller utan cursoradressering fungerar utan vidare, däremot är det problem med program som arbetar mer direkt mot bildminnet och program som utnyttjar luckor i bildminnet.

I orginalutförandet är bildminnet disponerat på följande sätt: De första 120 cellerna (fr o m adress 31744) i bildminnet utnyttjas av raderna 0, 8 och 16 i bildskärmen, därefter kommer en lucka på 8 celler, de följande 120 cellerna utnyttjas av raderna 1, 9 och 17, sedan kommer en ny lucka på 8 celler osv. Totalt finns 8 luckor om 8 byte. Var varje rad börjar står i den ovannämda tabellen på adress 884 till 931.

I GeJo:s version är det så ordnat att det nya bildminnets 240 första celler (fr o m adress 23552) utnyttjas för rad 0, 8 och 16, därefter kommer en lucka på 16 byte osv. Det nya bildminnet utnyttjas av raderna 0-3, 8-11 och 16-19 och det blir 4 luckor om 16 byte. Det gamla bildminnet med början på adress 31744 utnyttjas av de övriga raderna och får 4 luckor om 16 byte. Det finns alltså fortfarande 64 byte som inte utnyttjas av bildskärmen men problemet är att luckorna delvis ligger på andra platser. Av de 8 tidigare luckorna finns varannan kvar, de som börjar på 31992, 32248, 32504 och 32760. De övriga 4 tidigare luckorna hamnar nu i rader på den högra delen av 80-kol bildskärmen. Information som lagras i dessa påverkas när man använder bildskärmen i 80-kol mode.

Ta t ex T80PRT, i en äldre version som jag hade lades vissa parametrar (kontrolltecknen) i bildminnets första lucka med början på 31864. När jag försökte använda T80PRT i 80 kol mode så kunde jag se vissa underliga tecken på den högra delen av bildskärmen, när skärmen skrollades förstördes naturligtvis innehållet i cellerna. Dessbättre lagrar senare versioner av T80PRT parametrarna i POKEarean i stället.

MyAB har kommit ifrån problemmet med luckorna genom att använda friska 2K RAM för det nya bildminnet i 80-kol mode medan det gamla minnet användes oförändrat i 40 kol mode. Kvar står dock problemen med program som skriver eller läser direkt i bildminnet.

Jag tyckte det var minst sagt retsamt att inte kunna utnyttja den fulla större skärmbredden för terminal och textbehandling därför gav jag mig i kast med att försöka modifiera en del program så att de kunde utnyttja den större radlängden. I förhoppning att mina erfarenheter kan vara till hjälp för dem som vill ge sig på andra program skall jag relativt detaljerat redovisa hur jag burit mig åt med TV-editorn och T80PRT.

TV-editorn

Genom att ta bort den knepiga rad 10 i TVMAIN (den har ingen annan funktion än att ge omstart utan nollställning när man ger kommandot LIST) kan TVMAIN listas och det går att se hur programmet är uppbyggt. På ett antal ställen görs CALL till maskinspråksrutiner som lagrats in från TVSUBR.ABS. Med Arne Stockmans program DISASEM (kommer på ABC-kassett nr 3) disasemblerade jag rutinen och letade sedan efter konstanter med värden 40,39. Jag hittade 40 och på flera ställen 39, samt dessutom 38. Vidare letade jag efter adresser till bildminnet eller radadress-tabellen på 884 till 931. Jag fick såsmåningom min hypotes bekräftad att jag hittat de ställen som kontrollerade radlängden. Jag chansade då på att med pokesatser ändra 40, 39 och 38 till resp 80, 79 och 78, vidare öka på en blank rad till 80 tecken, samt ändra startadresssen till början av vissa rader i bildminnet. Det visade sig vara vad som behövdes göras för att editorn skulle fungera. Sedan kan man bygga på med sådant som gör att programmet är generellt användbart oberoende av vilken utrustning man råkar ha.

T80PRT

Styrkt av framgången med TV-editorn tog jag därefter itu med T80PRT och utgick då från en version som tar upp 4740 bytes av RAM-minnet i terminalmode och där kontrolltecknen lagras i POKE-arean.

Jag disassemblerade den maskinkod som lagras in av T80PRT och använde samma metod att leta efter konstant 40 och bildminnesadressering. Dessutom letade jag efter adressering av de celler i pokearean där kontrolltecknena lagras. Konstanten 40 fanns på flera ställen men det är bara på två ställen den behöver bytas ut. I 80-kol mode är det lämpligt att sätta kontrolltecknena "WEFV" ur spel, vilket enklast görs genom att poka 0 i cellerna där dessa lagras. Efterssom ändring bara behöver göras på två ställen i T80PRT:s maskinkod, nämligen på orginal-BOFA plus 2383 resp 2603, kan man lika lätt göra denna ändring från det program som utnyttjar T80PRT. När man kör T80PRT så höjs BOFA med 4740. Det nya BOFA blir orginal-BOFA plus 4740. Låter man sitt program som utnyttjar T80PRT läsa av var det nya BOFA ligger så är det bara att låta programmet poka in aktuell radbredd på adresser som kommer att ligga under den nya BOFAn.

I programmet TERMINAL görs detta i rad 360. För att växling mellan 80 kol och 40 kol mode terminal skall kunna göras smidigt så har jag låtit programmet själv hålla reda på vilket värde (40 eller 80) som skall läsas in. Aktuell radbredd finns lagrat i cell Å%. För tangentbord med checksumma 11273 är Å%=472, men vid andra checksummor har Å% annat värde. Därför låter jag programmet själv på raderna 30 till 80 leta fram Å% och skriva ut värdet. I rad 20 ligger en test huruvida tangentbordet har byggts ut för 80 kol bildskärm. Rad 60 läser av den nya BOFAn och rad 70 bestämmer vilka kontrolltecken som skall användas i 40 kol mode för att i T80PRT styra bildskärmsfönstret. Rad 310 ställer om radbredden, rad 320 tar bort funktionen av kontrolltecknena och rad 340 återställer funktionen.

Gunnar Tidner

Modem eller inte modem? That's the question!

Eftersom jag ännu befinner mig i marginalen vad avser datakommunikation antar jag att jag inte helt begripit vilken finess modem är. Jag ser det närmast som en möjlighet att hämta hem program från ABC-monitorn. Men vad finns det för program att hämta?

Jag hade hoppats att det skulle dyka upp en förteckning i ABC-bladet vad monitorn innehåller. Denna förteckning kan vad avser program som varit publicerade i bladet enbart hänvisa till nr och sida vad avser programförklaring och i övriga fall enbart ha en kort sådan, som t ex:
HUSKAL.BAS - Kalkylerar boendekostnad
EXTRACT.BAS - Eleminerar rader i början och slut av program

Denna presentation vore inte enbart värdefull för en presumtiv MODEM-köpare utan gagnar i ännu större grad medlemmar i förskingringen som ännu inte har modem. På det sättet skulle "icke-stockholmare" slippa att till dryga telefonkostnader bläddra genom bibloteket för att se om det finns något av intresse utan kunde gå direkt till skott.

Nytillkommna program kan lämpligen presenteras i bladet efterhand som de läggs in och en gång om året sammanfattande.

En annan fråga är: Hur får jag veta var närmaste monitor finns? Av Gunnar Tidners artikel i ABC-bladet 1981:2 förstår jag att det finns ett antal monitorsystem på andra orter. Men var? Även här tror jag att det vore till nytta med en förteckning.

Hälsningar
Lennart Lindulf

Jag hoppas att min artikel i detta nummer ger dig svar på de flesta frågor du ställt. När du ser den långa LIB-listan förstår du nog att det är ett ganska omfattande arbete som måste utföras innan dina önskemål kan bli tillgodosedda. Dina synpunkter har vidarebefordrats till dem som medverkar i "Programbiblioteksprojektet". Vi hoppas att få fler som vill medverka, även "icke-stockholmare" kan hjälpa till med detta arbete.

Att LIB-listan är lång och det tar dyr telefontid att bläddra igenom den är vi väl medvetna om. Om du ansluter en printer med serie-interface kan du få listan utskriven sedan behöver du bara titta på förändringar som normalt står först i listan. Med jämna mellanrum brukar någon vänlig själ lägga in en fil med namnet DATABANK.LIB men det är svårt att hålla filen helt aktuell. Det är möjligt att det så småningom kan göras automatiskt utan att det inkräktar på den tid systemet är tillgängligt.

Gunnar Tidner

```
 10 REM TERMINAL med möjlighet att växla
    skärmbredd
 15 REM Program av Gunnar Tidner 1981-12
    -20
 20 IF INP(4) AND PEEK(885)=124 THEN 100
    ELSE ; CHR$(12)
 30 FOR Ä%=0% TO 500%
 40 IF (INP(3) AND PEEK(Ä%))<>(INP(4) AN
    D PEEK(Ä%)) THEN 60
 50 NEXT Ä%
 60 B%=PEEK(65052)+256*PEEK(65053) : REM
    BOFA
 70 Ä$="WEFV" : REM Kontrolltecken i T80
    PRT
 80 ; "Ä%=";Ä% : FOR T=1 TO 1500 : NEXT
    T
 90 GOTO 300
100 ; CHR$(12);CUR(5,10)"Meny:"
110 ; : ; " 1    Terminal Halv Duplex"
120 ; " 2    Terminal Full Duplex"
130 ; " 3    Skärmbredd (";PEEK(Ä%);" )
    Växla?"
140 ; " 4    Exit"
150 ; CUR(12,0)"Välj:"; : GET C$ : ; C$
160 IF ASC(C$)<49 OR ASC(C$)>52 THEN 100
170 ON VAL(C$) GOTO 180,190,300,290
180 OPEN "V24:KB.1" ASFILE 1 : GOTO 200
190 OPEN "V24:KA.1" ASFILE 1
200 INPUT $1,A$
210 REM plats för rader motsv ABCTRANS
270 REM ---"---
280 GOTO 100
290 END
300 REM Bryt skärmbredd
310 IF PEEK(Ä%)=40 THEN Ö%=INP(4) ELSE Ö
    %=INP(3)
320 IF PEEK(Ä%)=80 THEN POKE -21,0,0,0,0
    : GOTO 360
330 FOR Ö%=1% TO 4%
340 POKE Ö%-22%,(ASC(RIGHT$(Ä$,Ö%))-ASC(
    "")) OR 128%
350 NEXT Ö%
360 POKE B%-2357,PEEK(Ä%) : POKE B%-2137
    ,PEEK(Ä%)
370 GOTO 100
```

TV skärmeditor

På ABC-kassett nr 2 fanns en skärmeditor som är mycket användbar. Denna text är en uppdatering av filen TV.TXT som också fanns på kassett nr 2. Det har visat sig att editorn innehåller ytterligare en del nyttiga funktioner. På ABC-kassett nr 3 kommer en version som gör att denna editor även kan användas av de som har enbart kassett. Vidare kommer en modifiering till 80 kolumners bildskärm. Vad som ännu saknas är möjlighet att flytta ett textstycke från en plats till en annan.

Utan extra minne kan bufferten innehålla maximalt 4769 tecken, i kassettversion 5537. Med 16 K extra minne kan bufferten innehålla 21153 resp 21921 tecken. Därtill kommer den särskilda inskrivningsbufferten på 500 tecken. Det gör ingenting om den senare bufferten blir full, då tömms texten automatiskt över i den stora bufferten men man tappar normalt det sist inskrivna tecknet.

Cursorkontroll

ü	uppåt
RETURN	nedåt
<-	vänster
->	höger

Kommandon

alla kommandon avslutas med två st. högerpilar

;Y läs fil
;X skriv fil med exit
;S skriv fil utan exit
nD delete n tecken
nK kill n rader
nL flytta n rader framåt
nC flytta n tecken framåt
J flytta markör till första tecken i buffert
ZC flyttar markör till efter sista tecknet i bufferten
JZDdelete hela bufferten
I insert, inmatning av ny text flera rader kan matas in på kontrollraderna längst ner på skärmen. Avslutas med två högerpilar. Observera att även alla controlltecken (utom CTRL-C och CTRL-I) kan matas in. Dessa blir på skärmen markerade som tex ^D för CTRL-D.
Stextsträng<h-pil> söker "textsträng" i buffert
nStextsträng<h-pil> söker n:te förekomsten av "textsträng" i buffert
Rgammaltext<h-pil>nytext<h-pil> söker efter "gammaltext" i bufferten och byter ut den mot "nytext"
CTRL-B flyttar markören till skärmens mitt

Z= skriver ut antal tecken i buffert
.= skriver ut markörens position

Flera kommandon kan ges i en och samma kommandosträng. Kommandona utföres först när man skrivit två högerpilar efter varandra.

Den som vill kan byta ut standardmarkören <*> mot "fyrkant" (ASCII 127) genom att göra CTRL-C och därefter POKE 65532,1 och POKE 64136, 127. Gör sedan LIST för att komma in i programmet. Pokesatserna kan även läggas in i TV.BAC resp TVSUB.BAC som kommer på ABC-kassett nr 3.

Gunnar Tidner

BEHÖRIGHETSKONTROLL

För alla användare av ABC80 är lätthanterliga program ett önskemål. Det kan dock finnas tillfällen när svårtillgängligheten måste få komma i förgrunden. Låt oss göra ett tankeexperiment: Vi antar att du får besök av bekanta. De har ingen egen dator, men kan ändå så mycket att de kan ladda och köra färdiga program. Då får du frågan: Har du något intressant program till datorn ? Blygsam som du är, fast generös, svarar du: Ett eller annat kan ju alltid finnas. Du kan söka där. Du kan naturligtvis inte neka dem att provköra. Dina bekanta sitter nu vid datorn, och du själv står i köket för att koka kaffe, om dom nu kan ha tid med det. Då inträffar plötsligt en KATASTROF ! På bildskärmen finns dina ekonomiska transaktioner snyggt uppradade, med kontonummer och allt. Även påminnelsen från banken att räntor och amorteringar skall betalas i tid, finns där. En blixt rusar efter ryggraden, och efteråt kommer tanken: Vad skall jag göra nu ? Emigrera, eller...? Det enda rätta svaret måste bli att det helt enkelt inte får hända!

Lösningen på problemet är att införa behörighetskontroll i HEMLIGSTÄMPLADE program. Behörighetskontrollen kan på olika sätt göras enklare eller svårare. Den ger dock alltid samma resultat: Den obehörige förlorar programmet.

Den rent allmänna principen består i att den som är behörig har en nyckel (kod) till programmet. Nyckeln kan bestå av ett antal siffror, bokstäver eller andra tecken, skrivna i en viss kombination. I de fall där endast ett fåtal speciellt utvalda personer får bruka programmet, kan det vara ordnat så att flera personer 'låser upp' programmet med sin personliga kod. Det betyder alltså att operatören måste vara godkänd av samtliga 'nyckelpersoner'.

I privata sammanhang är det säkert alldeles tillräckligt om det finns en nyckel, vilken innehas av programmets användare. Nyckelkoden torde inte heller behöva vara särskilt lång eller svår. Men det finns ju folk som känner till kommandot LIST. Hur listigt det kan vara visar nedanstående exempel,som är en del av ett program.

```
500 ;CHR$(12%): POKE 2^16%-2^7%, 83%,
80%, 65%, 82%, 66%, 65%, 78%, 75%, 69%,
78%, 10%, 10%, 13%, 75%, 79%, 78%
510 POKE 2^16%-2^7%+2^4%, 84%, 79%, 32%,
110%, 114%, 58%, 42%, 32%, 49%, 48%,
55%, 45%, 57%, 48%, 53%, 49%
520 POKE 216%-27%+25%, 56%, 57%, 57%,
45%, 52%, 50%, 69%, 65%, 32%, 42%, 10%,
10%, 13%, 85%, 116%, 103%
530 POKE 2^16%-2^7%+2^5%+2^4%, 46%, 32%,
98%, 101%, 104%, 46%, 32%, 56%, 49%,
46%, 48%, 56%, 46%, 48%, 49%, 58%
540 FOR N%=0% TO 65% : ;CHR$ (PEEK
(2^16% -2^7%+N%));: NEXT N%
550 READ C,A%,B%
560 A$=ADD$ (-NUM$(410*A%), NUM$
(-9%*B% +C/100%), 2%): ;A$
570 DATA 23,241,79
```

Som du själv ser är det inte speciellt mycket som den oinvigde får ut av programlistan. Det krävs vissa kunskaper i programmering för att tolka den. Är du intresserad, så knappa in och kör det.

Programlistan är ett exempel på hur ett svårtolkat program kan byggas upp. Nu skall vi återgå till behörighetskontrollen, och inriktar oss då på privatpersoner och deras hemligheter. Den allra första programraden bör innehålla ;CHR$(12%) och utskrift av texten BEHÖRIGHET. Sedan kommer vi till inmatningen av behörighetskoden. Här har vi en stor möjlighet att göra kodinmatningen verkligt kritisk. Vi kan nämligen använda GET-satser, och då finns det bara en möjlighet att göra rätt!

Använder vi oss dessutom av direktkontroll tecken för tecken, då försvinner programmet omedelbart om ett tecken är fel. Det finns även möjligheter att kontrollera koden sedan den är helt inmatad, men det blir ingen skillnad. Måste programmet försvinna om koden är fel? Svaret ligger hos programmeraren. Man kan även begära ny behörighet, men det kan innebära ett utslitet tangentbord. Den bästa lösningen är nog trots allt att programmet utplånas om koden är fel. Vi skall nu konstruera ett konkret program för behörighetskontroll.

Den första raden kan exvis se ut så här:

```
10 ;CHR$(12%) CUR(12%,10%) 'B E H Ö
R I G H E T'
```

Inmatningen av koden lägger vi i en loop.

```
20 FOR N%=1% TO 5% : GET Q$(N%)
```

Kontrollen kan vi göra så här:

```
30 READ A%(N%)
40 IF ASC(Q$(N%))<>A%(N%) THEN CHAIN
' 'ELSE NEXT N%
50 DATA 49,65,50,66,51
```

Inmatningsloopen på rad 20 har slutvärdet 5 vilket betyder att koden består av 5 tecken. Rad 30 läser data i tur och ordning från rad 50. Observera att indexet för Q$ på rad 20 och A% på rad 30 är lika. På rad 40 kontrolleras om den inmatade koden är rätt eller fel. Om den är rätt fortsätter inmatningsloopen med NEXT N%, annars CHAIN ' '. CHAIN ' ' betyder att datorn skall ladda programmet ' ', och det är ju inget program!

Rad 40 kan också utformas på annat sätt. Några exempel:

```
40 IF ASC(Q$(N%))<>A%(N%) THEN Z=CALL
(0%) ELSE NEXT N%
```

eller:

```
40 IF ASC(Q$(N%))<>A%(N%) THEN OUT
57%,3% : END ELSE NEXT N%
```

CALL till adressen 0% är att anropa BASICromets reset. Det är samma sak som att trycka på resetknappen. OUT 57%,3% är en order till tangentbordskontrollen att utestänga alla teckeninmatningar som kommer. Man kan bara 'låsa upp' med resetknappen, och då är programmet försvunnet. Datasatsen på rad 50 är kodens ASCII-värden. Man behöver alltså inte skylta med koden utskriven i klartext. Med ovan angivna datasatser blir koden: 1A2B3.

Jag hoppas att detta gett de tipssom behövs för att du skall kunna komponera din egen kod,med den svårighetsgrad som du själv finner lämplig.

LYCKA TILL

568 Sigvard Holmlund

Hej

Här kommer ett tips för de som använder ABC800 med högupplösningsgrafik. På skärmen kan man visa 240x240 punkter samtidigt. Grafikminnet på 16k rymmer emmellertid 256x256 punkter. Nästan hälften av dessa överskjutande punkter som endast innehåller "skräp" kan man se om man utför OUT 6,N% där N%=100% exempelvis. Vad som händer är att bilden ritas upp med början från en annan startposition i HR-minnet.

Problemet är nu att de överskjutande punkterna inte kan nås från Basicen. För x större än 239 eller mindre än 239 ger FGPOINT x,y error 176 (grafisk punkt utanför skärmen). Inte heller kan man med PEEK eller POKE nå HR-minnet eftersom detta (som delvis har sama adresser som Basicrommet) endast går att adressera då instruktioner på adresserna 30720-32767 (grafikrommet) exekveras. Jag ringde därför till Luxor i Motala och fick tipset att i de sista 3 bytarna i grafik-rommet (adress 32765-32767) ligger instruktionerna LDIR, RET.

För den, som liksom jag, endast har rudimentära kunskaper om assembler, kommer här en förhoppningsvis begriplig förklaring av hur man kan utnyttja detta.
LDIR-instruktionen är en slags loopinstruktion för kopiering av en sekvens data från en del av minnet till en annan. För det behövs en pekare, som pekar på adressen till den byte som skall kopieras. Som pekare används ett register i CPUn som kallas HL. Vidare behövs en pekare, som pekar på adressen dit byten skall kopieras. För denna pekare används CPUns DE-register. Till siste behövs också en räknare som håller reda på hur många fler byte som skall kopieras. Som räknare används CPUns BC-register. Instruktionen RET är helt enkelt return.

För att överföra data till HR-minnet behöver vi alltså ladda HR-, DE- och BC-registren med lämpliga värden och göra ett hopp till adress 32765. En praktisk användning av det här förfarandet, nämligen nollställning av hela HR-minnet, illustreras i bifogade programexempel. Programmet fungerar så här:
På rad 130 nollställs byten på adress i HR-minnet.
På rad 140 läggs ett litet assemblerprogram i variabeln A$.
På rad 150 hoppar vi ur Basicen till adressen där ABCn behagade placera A$ samtidigt som vi laddar DE-registret med adressen 1.

Själva assemblerprogrammet gör följande (koderna decimalt inom parentes):
Ladda BC (1) med talet 16383 (255,63).
Ladda HL (33) med adressen 0 (0,0).
Gör ett subrutinhopp (205) till adressen 32765 (253,127).
Avsluta assemblerprogrammet (201) och återgå till Basicen.

Vad hela programmet gör är alltså att först nollställa byte 0 i HR-minnet, sedan kopiera den i byte 1. Härefter kopieras byte 1 i byte 2 osv tills hela HR-minnet är nollställt.

På programrad 160-320 visas en enkel användning av OUT 6,... instruktionen för rörelse av bilden i Y-led. Skulle inte hela grafikminnet först ha nollställts så blev utseendet av bilden helt annorlunda. Den som vill kan roa sig med att lägga till följande rad: 135 FGPOINT 2,239,3.

Uppenbarligen kan man genom rätt enkla assemblerprogram innehållande hopp till adress 32765, flytta data mellan HR-minnet och det vanliga RAM-minnet, för att sedan exempelvis lagra bilder i externminne. Det bör också vara möjligt att ta sig en titt på vad grafik-rommet innehåller.

Per Ahlin

A B C-bladet Samlingsnummer
1981 :4-5 sidan 24

MERA MINNE I GAMMAL FÖRPACKNING MED 2716-MINNEN PÅ ETT 2708-KORT.

Har du också sneglat lite mot minneshålet i ABC 80 ? Dvs. adresserna 16384-24575 (4000H-5FFFH) vilka nu, då det inte tycks komma någon 24k BASIC, oftast är helt lediga. Vore det inte en god ide att kunna placera något ofta kört program där i PROM?

Men måste man inte skaffa ett nytt PROM-kort för detta? Kanske inte, det beror på vad du redan har för utrustning.

Skrivaren av dessa rader, lycklig ägare till en trotjänare i form av en DataDisc 80, har nu kört i åtskilliga månader med ett PROM-kort som är modifierat att ta ?k byte 2716-minnen i stället för 1k byte)8.

I min flexskiveutrustning ingår ett kort benämnt "4680A32K 8k RPROM", avsett för 8 st 2708-minnen. Ett försök att skaffa scheman från tillverkaren (representanten) slog slint, dessa bevakas där som kronjuvelerna. Nå, detta är ett skäligen enkelt krets-t varför det inte tog någon längre stund "rita kartan efter verkligheten".

Det visade sig vara relativt enkelt att modifiera kortet till att använda 2716-minnen. Sex avbrott och sex byglingar är allt som behöver göras på mönsterkortet. Här följer detaljerade anvisningar över modifieringarna för just detta EPROM-kort.

Tillvägagångssättet bör vara snarlikt för andra, liknande 2708-kort. Från början täckte kortet adressområdet 24576-31743 (6000H-7BFFH). Efter modifikationen gäller 16384-31743 (4000H-7BFFH). Man når inte ända upp till 32768 (8000H) eftersom videominnet ligger här i ABC 80 och datorn därför aldrig släpper ut kontrollsignaler för PROM-läsning i dessa adresser. Därför är vid denna användning detta i själva verket ett 7-kretsars 2708-kort och efter modifieringen ett 7 1/2 kretsars 2716-kort.

Modifieringen går så till att man först programmerar 2716-minnen, det blir max. fyra stycken, att innehålla samma data som de existerande 2708-orna. Med en lämplig programmeringsutrustning är detta enkelt. ?t kan vara lämpligt att också spara deras ta i form av filer. För observera att du inte kan läsa 2708-or i framtiden efter modifieringen. Om du på annat sätt, t.ex. med en EPROM-programmerare, kan läsa av kapslarna kan du naturligtvis arkivera dem direkt.

Nu kommer vi till det elektriska. Se ur 1. Ändringarna genomförs enkelt av arje normalhändig person, riktiga verktyg fordras dock, och handlar om två saker:

1. Två ben har skilda funktioner på 2708 och 2716. På 2708 är pin 21 -5 V matnings-spänning, på 2716 används den vid program-mering (Vpp) och skall normalt vara +5 V. Pin 19 används på 2708 för +12 V, medan 2716 här har adressbiten A10.

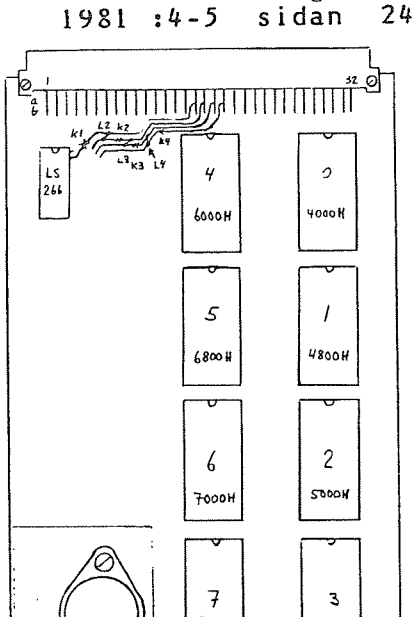


Fig. 2

Kortets ovansida efter modifiering.

2. Adressavkodningen på kortet skall ändras. Adressbitarna A11-A13 förskjuts i princip uppåt ett steg, A14 och A15 är oförändrade eftersom vi är kvar inom samma 16k-grupp och den frigjorda adressbiten A10 används på 2716-orna.

Rent praktiskt består arbetet i att med skalpell eller hellre ett roterande slipstift kapa sex ledarbanor (K1-K6 i fig. 1). Man skall också löda dit lika många jumpringar (hopkopplingar) med virtråd eller liknande (L1-L6). K1-K4 och L2-L4 utförs på kortets komponentsida, se figur 2. Övriga ändringar göres på ledarsidan. Se upp med att göra avbrotten K5 och K6 på rätt sida om glättningskondensatorerna, de skall inte vara med i fortsättningen. L5 dras direkt från kort-kontaktens stift a19 till närmaste EPROM's pin 19. L6 är enklast att införa där ledarna för -5 V och +5 V går parallellt.

Fyra 2708 med DOS i adress 6000H-6FFFH satt tidigare i PROM-platserna 0-3, men observera att de två 2716-minnena med samma innehåll skall placeras i positionerna 4-5. Se för övrigt figur 2 för adresseringen av enskilda 2716-PROM inom kortet.

Esko Lehtinen

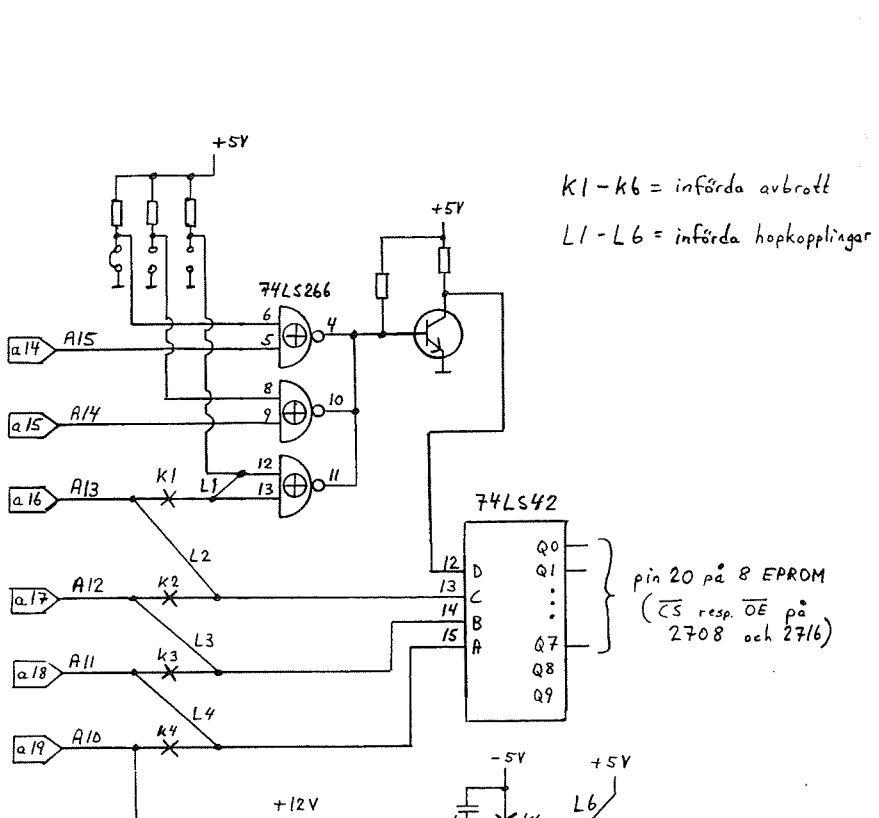


Fig. 1

2716 i st. f. 2708 på kretskort 8k RPROM.

UPPDELNING I FAKTORER

Från skolmatematiken minns säkert många de allmänna bråken, där man skulle uppdela täljare och nämnare i primtalsfaktorer för att kunna förkorta.

Det var relativt små faktorer i skolboken:2, 3, 5 och 7, och talen var inte heller så stora. Men hur bär man sig åt för att uppdela verkligt STORA tal i faktorer? Det är ett datorjobb! Nedanstående programsnutt visar principen:

```
 10 REM FAKT
100 ; 'SKRIV TALET: '; : INPUT X : ; 'FA
    KTORER: ';
110 F=2
120 FOR I%=1% TO 10000%
130 IF F*F>X THEN 210
140 IF X/F=INT(X/F) THEN 180
150 IF F=2 F=3 ELSE F=F+2
    NEXT I%
170 REM --- HITTAT FAKTOR
180 ; F',';
190 X=X/F
200 GOTO 130 : REM ---. SAMMA F
210 REM --- SLUT
220 ; X
```

börjar alltså med faktorn 2 (119), ökar sedan till 3 och därefter till de följande udda talen (150). Går F jämnt upp är det napp (140) och man hoppar ur loopen, skriver upp faktorn (180), dividerar bort den och hoppar in i loopen igen. Man behöver nu inte börja om från F=2 utan fortsätter med den faktor man hunnit till.

Med detta program kan man få även stora tal uppdelade i faktorer ganska snabbt. Det brukar imponera på den datakunnige man demonstrerar det för. Men flyttalsaritmetiken har sina gränser. Mer än sexsiffriga tal klarar den inte av. Prova förresten med 999999.

Med hjälp av ASCII-räkningen kan man lösa problemet. Det är gjort i programmet FAKTOR. Det programmet är litet noggrannare genomarbetat. Dels kommer programmet att 'stå still' på skärmen och dels finns en 'pausfågel' som talar om vad programmet håller på med och vilka faktorer som för tillfället prövas. Med hjälp av det programmet kan man faktoruppdela RIKTIGT STORA tal.

S. Wickberg

```
 0 REM ...SAVE FAKTOR...v.2 81 03 09
 20 REM Sven Wickberg, 0750-50456
 30 REM Baldersv 43, 130 54 Dalarö
 40 REM
 50 REM Uppdelning av tal i faktorer
 60 REM
 80 ; CHR$(12)CUR(6,0)'VILKET TAL SKA UP
    PDELAS I FAKTORER'; : INPUT W$
 85 ONERRORGOTO 80 : W=VAL(W$)
 90 X$=W$+'='
100 F$='2'
110 ; CUR(10,0)X$;
120 ; CUR(21,0)SPACE$(39)
130 ; CUR(21,0)'REST SOM TESTAS: 'W$
140 ; CUR(23,0)'PROVAR TAL:';
150 FOR I%=1% TO 10000%
160 ; CUR(23,12)F$;
170 Q$=MUL$(F$,F$,0) : IF COMP%(Q$,W$)=1
    % 260
180 Q1$=DIV$(W$,F$,2)
190 Q2$=DIV$(W$,F$,0)
200 IF COMP%(Q1$,Q2$)=0% THEN 230
210 IF F$='2' F$='3' ELSE F$=ADD$(F$,'2'
    ,0)
220 NEXT I%
230 X$=X$+F$+','
240 W$=DIV$(W$,F$,0)
250 GOTO 110
260 X$=X$+W$
270 ; CUR(10,0)X$;
280 ; : ; : ; 'MERA (J)'; : INPUT W$
290 IF W$='J' OR W$='j' OR W$='' 80
300 ; CHR$(12)CUR(10,10)'PÅ ÅTERSEENDE!'
310 END
```

Intresseförderlning mm bland ABC-klubbens medlemmar

Vi har bearbetat underlaget för den nya matrikeln vad gäller de förkryssningar som gjorts av medlemmarna på inbetalningskorten vid betalning av 1981 års avgift. Av de 1912 medlemmar som finns med i den nya matrikeln finns intressemarkering för 1674 st (88%). För dessa 1674 är den procentuella fördelningen:

Administrativa rutiner 51.9
Datakommunikation 45.4
Föreningsliv, div tillämp. 23.8
Matematik och statistik 46.2
Programmeringsteknik 79.8
Spel, grafik och musik 52.0
Styr- och mätteknik 58.7
Textbehandling 37.0
Hjälpa till 10.0
Har tillgång till:
Flexskiva 54.5
Modem 300 baud 18.9

Det betyder att numera har över hälften av klubbens medlemmar tillgång till flexskiveutrustning. En närmare udersökning av inloggningarna på Monitorn (se separata artikeln) tyder på att c:a 31% av klubbens medlemmar torde ha tillgång till modem.

Tidigare redovisning av intresseförderlning mm fanns i ABC-blad 1981:2.

MERA OM RESET

I nummer 2/81 skrev Göran Tengner att han inte kunde få fram programmet efter RESET. Det finns flera sätt att lösa det problemet på: Man kan chansa, vilket inte rekommenderas då det kan förstöra programmet för gott! Man kan använda PEEK(49155)... och fortsätta tills man hittar värdet 13. Lite besvärligt, men fullt genomförbart. Det bästa sättet är naturligtvis att använda datorn!

Följande program återställer ordningen i minnet! Programmet är skrivet i maskinspråk och måste POKEas som kommando. Försök inte göra ett BASIC-program av det, för då förstörs programmmet som du vill rädda.

Gör följande:
POKE 65408, 33, 0, 192(*),229, 30, 0, 62, 13
POKE 65416, 28, 237, 161, 32, 251, 225, 115, 201
;CALL(65408)

Och allt är frid och fröjd.

Det tal som skrivs ut efter CALLet behöver du inte bry dig om (det är adressen till första byte på nästa rad). Talet 192 som är märkt (*) i programmet (du skall givetvis inte skriva parenteserna och stjärnan) är koden för hur stort RAM-minne som finns i maskinen. 192 gäller för en standard ABC80 med 16 kB RAM. Har du 24 k skall du skriva 160 i stället, och har du 32 k så skriv 128.

Därmed skulle allt vara förklarat.

Vänliga hälsningar

Bjarne Jensen

ON INSTR + 1

Det eviga problemet hur man filtrerar bort otillåtna svar vid t.ex menyval löses snyggt genom en kombination av en INSTRing-sats och en ON sats.

Du har skrivit menyvalet på skärmen, t.ex. 1-5 och cursorn står nu och blinkar på GET-satsen och väntar på ditt svar:

```
100 GET G$: ON INSTR(1,'12345',G$)+1
GOTO 100, 200, 300, 400, 500, 600
```

För alla svar utom de rätta blir INTR=0, dvs. ON-satsen kommer envist tillbaka till GET satsen.

Samma system används givetvis när man kan svara JA eller NEJ på en fråga i stället för att traggla med IF G$='J' OR G$='j' etc.etc.

```
100 GET G$ : ON INSTR(1,'JjNn',G$)+1
GOTO 100, 200, 200, 300, 300
```

Bengt Sagnell
Meyrin, Schweiz

GALAX

Hej!

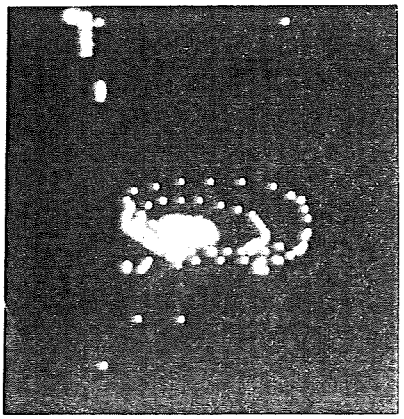
Detta program är skrivet efter en artikel i tidningen Byte aprilnumret 1979. Artikeln hette "A Simulated View of the Galaxy" och innehöll programförslag i "generalized" Fortran. Efter diverse prövningar lyckades jag skriva om det till ABC80 BASIC.

Programmet innehåller positioner för 465 stjärnor. Exekveringstiden är litet över tre minuter. Trots att ABC80:s grafik är så grov, får man ändå en bra framställning över hur stjärnhilmen ser ut från olika platser i rymden.

Istavn Gersner
Lund

- Istvan skickar med tre fotografier, av vilka två i en stereoviewer kan ge en tre-diminsionell bild av vintergatan, sedd från 100 ljusårs avstånd! Den tredje bilden återges här och visar hur det ser ut ca 20 000 ljusår ifrån solen.

Programmet är alltså sådant att man kan välja att betrakta bilden från olika avstånd och riktning. Detta används mycket vidd datorstödd konstruktion, och visar vad man kan åstadkomma även med BASIC.



Detta är ett utdrag av sidorna 47-50 i ABC-klubbens Rapport nr 1
av Arne Stockman. Renskrivningen är gjord av Modulservice S.Å
****

SYSTEMVARIABLER.

64768-82 Kopia av dosfilbeskrivning för den fil som arbetas med.

64786-87 Pekare i DOS-buffert 0.
DOS-buffertarna ligger efter varandra. Vid initiering sätts DOS-buffert 0 till adress 62720 men kan i prinsip sättas till vilken adress som helst.

64789 FELKOD från controller.

64790-91 Här lägger DOS-et pekare till basicens filbeskrivning, för Random Access.

64792 Antal omförsök kvar vid läsning/skrivning.

64796	Lagringsplats för kommando vid	RESIZE
64797-98	" " BC	OPEN
64799-800	" " DE	CLOSE
		PREPARE

64801-02	Tabell med hopp-	Checksummafel vid läsning.
64803-04	adresser dit man	" " skrivning.
64805-06	skall hoppa om	Felaktigt recordformat.
64807-08	det inträffar	Recordnummer utanför filen.
64809-10	ett fel. står	Filen skrivskyddad.
64811-12	det 0 här an-	Filen raderskyddad.
64813-14	vänds adressen	Skivan full.
64815-16	som står på	Skivan ej klar.
64817-18	64819-20.	Skivan skrivskyddad.

64819-20 DEFAULT hoppadress.

64821-27 DEFAULT device i Devicelista.

64832-847	Dosfilbeskrivning 0	En dosfilbeskrivning
64848-863	" " 1	anses som ledig, om
64864-879	" " 2	den andra byten är
64880-905	" " 3	255 (DRIVE-byten).
64896-911	" " 4	
64912-927	" " 5	
64928-943	" " 6	
64944-959	" " 7	

Dosfilbeskrivning 0 liksom dosbuffert 0 används internt av DOS-et.

64960-67 Antal använda Directory entries, används endast slumpmässigt.

64981-007 Temporära buffertar, aritmetik och printrutinen Basic.

65008-10 Klockan.

65011 Kolumnposition bildskärm.
65012 Radposition "

65013 Interuptflag tangentbord.
65014-15 Timer för repeatfunktionen tangentbord.

65017-18 Aktuell buffert kasett.
65019-20 Pekare alternativ input GETLI.
65021-22 Recordräknare kasett.
65023-24 Aktuell position kasettbuffert.
65025 Byte in från kasett.
65026 Bitcount inläsning kasett.
65027-28 Aktuell rutinadress när byte inläst från kasett.
65029-30 Checksumma kasett.

65031 CTRL-C flag.
65032
65033
65034-35 Devicerot.
65036
65037
65038
65039 Bit 0 satt om BASICERR.SYS öppen.
65040
65041
65042
65043
65044
65045
65046 Hit pekar IY.
65047
65048
65049
65050
65051
65052-53 BOFA.
65054-55 EOFA.
65056-57 HEAP.
65058
65059 Filnummer i print och input.
65060 Runmode (0 under programexekvering).
65061 Bit 4 = Traceflag, Bit 5 = Runonlyflag.
65062 Programmet "fixat".
65063-64 Stack.
65065-66 Variabelrot.
65067 Senaste OUT 1,X (Selectat kort).
65068-69 Adress this line (exekvering-fixprogram).
65070-71 Datapointer (exekvering av READ).
65072-73 Linedatapointer (adress till raden med Datasats).
65074-75 Filrot.
65076-77 ONERRORGOTO-adress.
65078-79 Runtime-stackpointer.
65080 ERRCODE.
65081-85 Slumptal.
65086-87 Pekare till argument FN-funktion.
65088-207 Radbuffert.

A B C-bladet Samlingsnummer
1981 :4-5 sidan 27