LEDAREN.

är nu mer än fem år sedan ABC80 kom ut på marknaden. Fem år är en ganska lång tid när det gäller datorer. Det är imponerande hur väl ABC80 har kunnat hänga med i utvecklingen under denna tid. Förklaringen torde vara dels den ursprungliga försäljningsframgången dels en från början ovanligt vällyckad konstruktion. Tack vare den stora spridning som ABC80 fick har det funnits en marknad för kompletteringssatser som gjort det möjligt att uppgradera grundutrustningen t ex vad gäller minneskapacitet och 80 teckens bildskärm utan att kompabiliteten till tidigare program gått förlorad. Att det funnits en livskraftig användarklubb där medlemmarnas erfarenheter kunnat tas till vara och spridas till övriga användare har säkert också varit en bidragande förklaring.

Hur länge till ABC80 klarar att hänga med är en öppen fråga.

ABC80 i grundutförande med endast 16 K Ram och 40 teckens bildskärm kan med dagens mått inte anses fylla kraven på en persondator för allroundbruk. Men den kan fortfarande vara det ekonomiskt bästa alternativet vid vissa speciella tillämpningar. Rent tekniskt kommer apparaten att hålla för många års ytterligare användning. Men förr eller senare kommer den dag då man byter till en modernare utrustning. Det måste vi som blivit känslomässigt fästa vid våra apparater inse.

De som nu skaffar sig datorutrustning väljer i stället någon ur ABC800-serien. För professionellt bruk torde de flesta välja ABC806.

För den som redan har en ABC80 och investerat mycket i egna program kan det dock fortfarande vara ett alternativ att gradera upp sin utrustning.

Man skulle kunna uttrycka det så att det pågår en form av skalömsning bland våra medlemmar och datoranvändare. Det är en process som inte alltid går helt smärtfritt.

Det är nu fyra år sedan ABC-klubben bildades.

År 1983 har även på ett annat sätt inneburit en skalömsning för klubben. Den kraftiga ökningen av antalet medlemmar - medlemsantalet passerade under året 4000 - har medfört vissa påfrestningar på klubbens administrativa rutiner. Tidigare har klubbadministrationen kunnat skötas på huvudsakligen ideell basis. Mot slutet av året visade sig detta allt svårare. Det ledde till en akut kris som inte löstes förrän klubben i december fick ett nytt kansli med halvtidstjänstgörande kanslist.

För ABC-klubben blir det nödvändigt att anpassa sig efter de ändrade förhållandena om vi vill göra anspråk på att vara en klubb för alla användare av ABC-datorer. Och det är vad som beslöts på klubbens årsmöte.

Utvecklingen under de gångna åren visar att det finns ett stort intresse och behov av en användarklubb där vi kan utbyta och dela med oss av våra erfarenheter och få hjälp av varandra. Om vi skall kunna bibehålla det stora intresse som nu finns för klubben så måste vi kunna tillfredsställa vissa grundläggande krav från våra olika kategorier av medlemmar. Vi måste ge ut en tidning där alla kan finna saker som intresserar honom eller henne. Det måste även i fortsättningen komma fram nyttiga och bra användarprogram och vi måste på något sätt klara av att distribuera dem även till de som inte har någon kassettbandspelare.

ABC-klubben eftersträvar goda och förtroendefulla relationer till Luxor och andra leverantörer och tillverkare. Men klubben måste slå vakt om sin självständiga, av ekonomiska intressen oberoende ställning. När vi stöter på svagheter i hårdvaran eller mjukvaran är det vår uppgift att säga ifrån. ABC-klubben representerar tillräckligt många och kunniga medlemmar för att vi skall ha rätt kräva att våra synpunkter skall bli tagna på allvar.

Det är inte särskilt ovanligt att vi är först med att upptäcka ett systemfel eller annan svaghet i viss utrustning, fel som någon medlem i klubben hittat och rapporterat. I flera fall har det varit i samband med en udda tillämpning i klubbens egen verksamhet som ett fel först avslöjats. Det är ett rimligt krav att felen eller bristerna så småningom rättas till i nyproduktionen och att användare som drabbats av felen kan få den aktuella enheten utbytt om inte helt kostnadsfritt så i varje fall till en låg självkostnad.

Vi anser oss också ha rätt att ställa krav på att få tillgång till ordentlig dokumentation av utrustning och systemprogram. Här finns fortfarande allvarliga brister.

Gunnar Tidner

Snabbreferat.

Detta är ett snabbreferat av ABC-klubbens årsmöte. Avsikten är att som brukligt ha ett fylligt referat av årsmötesförhandlingarna. Detta är främst med tanke på att mötet besöks av en mycket liten andel av medlemmarna.

ABC-klubben hade årsmöte lördagen den 18 februari. Lokalen var Brommasalen, Gustavlundsvägen 168, Bromma. Samtidigt hade ordnats en ABC-dag där leverantörer av tillbehör m m till ABC-datorerna visade upp sina sortiment och diskuterade sina produkter med intresserade besökare.

Årsmötet öppnades av ABC-klubbens ordförande Gunnar Tidner som hälsade alla välkomna. Till mötesordförande valdes Torbjörn Alm och till mötessekreterare Bo Kullmar.

Årsmötet godkände verksamhetsberättelsen, resultat- och projektredovisning. Revisorerna redovisade revisionsberättelsen och tillstyrkte att styrelsen beviljas ansvarsfrihet för år 1983, vilket årsmötet beviljade.

Vidare beslöts att fastställa balansräkningen och att årets överskott, kronor 9.849:41, skall föras i ny räkning för verksamhetsåret 1984.

Beslöts att anta Stig Löfgren m fl:s motion om att stödet till lokalavdelingar inte skall vara tidsbegränsat till de två första åren.

Beslöts att fastställa styrelsens förslag om medlemsavgiften för 1984 till 140 SEK för seniorer respektive 80 SEK för juniorer.

Beslöts att ge styrelsen mandat att besluta om en preliminär medlemsavgift för 1985 om högst 160 kronor för senior och 100 kronor för juniorer.

Beslöts att med en mindre ändring fastställa budgeten för 1984 enligt styrelsens förslag.

Beslöts efter votering att välja Gunnar Tidner som ordförande.

Beslöts att välja Stig Löfgren som vice ordförande.

Beslöts att välja följande till ledamöter i ABC-klubbens styrelse:

Bo Kullmar omval
Rune Mattsson omval
Gösta Stenhorn nyval
Torsten Ljungström nyval
Jan Holmberg nyval

Beslöts att till suppleanter välja
Kalle Lindström och Björn Sjöborg.

Beslöts att välja Nils Larsson och Lennart Jansson till revisorer, och att välja Lars-Göran Göransson till revisorssuppleant.

Till valberedning utsågs Göran Sundqvist (sammankallande), Magnus Hedner och Allan Larsson.

Beslöts att ändra §1 till följande lydelse:

ABC-klubben är en ideell sammanslutning av användare av ABC-datorer. Klubben skall verka för att tillvarata gemensamma intressen av datorer och datortillämpningar, verka för ökade kunskaper inom dessa områden till nytta, utbildning och nöje.

Beslöts att §5 skall ha följande lydelse:

Medlemsavgiftens storlek bestäms av årsmötet. Medlemskap träder i kraft när medlemsavgiften erlagts. Medlemsavgift skall erläggas årligen före den 15 mars.

Dessa beslut noterades vara enhälliga.

Beslöts att styrelsen innan styrelsen tar beslut med större eknomiska konsekvenser skall styrelsen rådgöra med revisorerna.

Beslöts att styrelsens redovisningshandlingar till årsmötet skall innehålla en lista över inventarier. Listan skall upprättas i samförstånd med revisorerna.

Styrelsen meddelade att man avser att utarbeta ett förslag till reviderade stadgar för ABC-klubben. Årsmötet uttalade sig för att förslaget skall behandlas på ett extra årsmöte under hösten 1984.

Beslöts att på förslag av Gunnar Tidner vid avtackningen av avgående styrelsemedlemmar att utse **Ulf Sjöstrand, Kjell-Åke Johansson, Marianne Forsman och Joe Johnsson till hedersmedlemmar i ABC-klubben.** Gunnar Tidner framhöll att dessa har under tre mandatperioder haft mycket arbetskrävande uppgifter inom ABC-klubbens styrelse.

FORTH.

Hej!
Översänder här ett stränghanteringspaket för Forth att publiceras/distribueras till ABC-klubbens medlemmar. Programmen skrevs när jag fortfarande var ganska nybliven Forth-programmerare så det föreligger inga anspråk på att det är det 'elegantaste' sättet att lösa problemen.
Huruvida man verkligen har användning för ett sån't här paket överlåter jag åt var och en att avgöra. Programmen skrevs i första hand som programmerings övning.
Förhoppningsvis är alla 7 skärmarna 'bug'-fria men skulle någon hitta något, eller ha andra synpunkter på paketet, så hör av er.

P-E Martin <2277>
Västerås

PS
Här är några Forth-ord som används av sträng-rutinerna följt av några enkla exempel för att 'komma igång'. I övrigt hänvisas till manualen.

```
: 1-    1 - ;
: 2-    2 - ;
: PICK  SPÉ SWAP 2 * +
        DUP S0 É 1 - >
        IF ." Out of bounds!"
           DROP CR
        ELSE É ENDIF ;
: 2ROT  6 PICK 6 PICK SPÉ 2- DUP
        12 + DO I É I 4 + !
          -2 +LOOP 2DROP ;
: 2OVER 4 PICK 4 PICK ;
: NDROP 0 DO DROP LOOP ;
: <ROT  ROT ROT ;
```

Exempel:

```
15 $ FÖRNAMN  skapar variabeln
  FÖRNAMN med max 15 tecken-
  positioner.
  NAMN $<-" Sverker"  tilldelar
  ÖRNAMN innehållet Sverker.
```

```
FÖRNAMN $MAX  lägger det maximala
  antalet positioner för variabeln
  på stacken som är 15.
FÖRNAMN $SIZE  lägger den aktuella
  strängängden på stacken,
  som är 7.
FÖRNAMN $.  skriver ut innehållet i
  variabeln.


4 10 $VECT FÄRG  skapar en sträng-
  vektor med 5 element (0-4),
  med  vardera 10 teckenpositioner.
2 FÄRG $MAX  ger 10 på stacken och
2 FÄRG $SIZE  ger 0 .


0 FÄRG $<-" GRÖN"
1 FÄRG 0 FÄRG $<-
2 FÄRG $<-" RÖD"
3 FÄRG $<-" GUL"
4 FÄRG $<-" SVART"
```

```
1 FÄRG $CONC" ARE"  ger element 1
  innehållet GRÖNARE.

0 0 FÄRG $V.  ger utskriften

GRÖN GRÖNARE RÖD GUL SVART

  medan

1 0 FÄRG $V.  ger utskriften

GRÖN
GRÖNARE
RÖD
GUL
SVART

0 FÄRG $VCLR  nollställer size-
  byten i samtliga element så
  vektorn kan betraktas som 'tom'.
  Texten ligger dock kvar i det
  reserverade minnesutrymmet och
  kan raderas med
0 FÄRG $BLANKS 1 FÄRG $BLANKS osv
```

FORTH STRÄNG-PAKET V 1.0 av Martin

Symboler :

a adress till size byten i strängen.
m max antal tillåtna tecken i sträng.
s aktuellt antal tecken i strängen.
p position i strängen.
n ett enkeltal, oftast antal tecken eller index.
c ascii-kod.
b booleskt värde (Kan vara -1).

Observera att a normalt betecknar en adress men i texten används den för att symbolisera strängens innehåll. Till exempel vid $COMP betyder a1<a2 att strängen a1's ascii-koder är mindre än a2's koder inte att adressen a1 är mindre än adressen a2.

$
Anropas i formen m $ namn som skapar strängvariabeln namn med reserverad plats för m tecken.
Variabeln får då formen :

```
Byte  1   2   3   4  ...  m+2
-     max size c1  c2 ...  cm
```

där max är reserverad längd och size är aktuell längd. c är ascii-tecken. Vid anrop av namn lämnas adressen till size-byten (byte 2) på stacken.

Anropar : ($)

$. (a —)
Skriver ut texten i variabeln.

Anropas av : $V.

$< (a1 a2 — b)
b=1 om a1<a2 annars är b=0.

Anropar : $COMP

$<- (a1 a2 —)
Strängen a1 tilldelas a2's innehåll utan att a2 förändras. Om a2 är längre än a1's maxlängd kapas överskjutande tecken bort.

Anropar : $MAX
Anropas av : ($§) $MID $CONC $INS

$<-§ (a n —)
Tilldelar a en siffersträng skapad av enkeltalet n.

Anropar : ($§)

$<-D§ (a d —)
Som $<-§ fast med ett dubbeltal.

Anropar : ($§)

$= (a1 a2 — b)
b=1 om a1=a2 annars är b=0.

Anropar : $COMP
Anropas av : $SEEK

$> (a1 a2 — b)
b=1 om a1>a2 annars är b=0.

Anropar : $COMP

$BLANKS (a —)
Fyller hela reserverade utrymmet i a med mellanslag. Size = max.

Anropar : $FILL

$CLR (a —)
"Raderar" strängen genom att noll-ställa size-byten.
Texten finns alltså kvar.

$COMP (a1 a2 — b)
Jämför två strängar och lämnar värdet b på stacken.
b= 0 om a1=a2
b= 1 om a1>a2
b=-1 om a1<a2

Anropar : 2ROT NDROP
Anropas av : $= $< $>

$CONC (a1 a2 —)
a1 tilldelas den sammanfogade strängen a1+a2. a2 förändras inte.

Mellanlagrar text i PAD, data i returnstacken.

Anropar : $<-

$DEL (a p n —)
Tar bort n tecken från och med pos. p ur a och "drar ihop" texten.
Mellanlagrar data i returnstacken.

Anropar : ($IX2) $ERROR 2OVER 1-

$ERROR (n —)
Skriver ut felmeddelande n relativt rad 0 i screen 4, tömmer stacken och gör QUIT.

Anropas av : $VECT $STORE $SEEK $LAST $GET $PUT $MID $LEFT $RIGHT $INS $DEL

$FILL (a c —)
Fyller hela reserverade utrymmet i a med ascii-tecknet c. Size sätts lika med max.

Anropar : $MAX
Anropas av : $BLANKS

$GET (a p — c)
Lägger ascii-tecknet c, från pos. p , på stacken.

Anropar : ($IX1) $ERROR

$INS (a1 p a2 —)
"Klämmer in" a2 i a1 från och med pos. p. a1's längd + a2's längd måste rymmas i a1's maxlängd.
Mellanlagrar text i PAD, data i returnstacken.

Anropar : ($IX3) $MAX ($IX4) $ERROR 1- <ROT $<-

$LAST (a c —)
Lägger till ascii-tecknet c sist i strängen.
Mellanlagrar text i PAD, data i returnstacken.

Anropar : $MAX $ERROR

$LEFT (a n —)
Strängen a tilldelas de n första tecknen från sig själv.

Anropar : $MID

$MAX (a — m)
Lägger maxvärdet för strängen på stacken.

Anropas av : $VECT $<- ($V) $LAST $INS $FILL

$MID (a p n —)
Strängen a tilldelas n tecken från och med pos. p från sig själv.

Mellanlagrar text i PAD.

Anropar : ($IX2) $ERROR $<-
Anropas av : $LEFT $RIGHT

$PUT (a p c —)
Skriver över tecknet i pos. p i strängen med ascii-tecknet c.

Anropar : <ROT ($IX1) $ERROR

$RIGHT (a p —)
Strängen a tilldelas de sista tecknen från och med pos. p från sig själv.

Anropar : $MID

$SEEK (a1 p a2 — p2)
Söker efter a2 i a1 från och med pos. p1. Om a2 hittas är p2 positionen annars är p2=0.
Mellanlagrar text i PAD, data i returnstacken.

Anropar : <ROT $ERROR $=

$SIZE (a — s)
Lägger size-värdet för strängen på stacken.
Är ekvivalent med C@.

$STORE (a1 p a2 —)
Skriver över text i a1, från och med positionen p, med a2.

Anropar : ($IX3) ($IX4) $ERROR

$V. (b a0 —)

Skriver ut texten i samtliga element i vektorn. Om b är 0 skrivs ett mellanslag mellan elementen, om b är 1 görs en radmatning.

OBS! Måste anropas med element-index 0 !

Anropar : ($V) $.

$VCLR (a0 —)
"Raderar" elementen i vektorn genom att nollställa size-byten.
Texten finns alltså kvar.
OBS! Måste anropas med element-index 0 !

Anropar : ($V)

$VECT
Anropas i formen n m $VECT namn som skapar strängvektorn namn med n+1 element vardera med m reserverade byets för text.
Vektorn får då formen :

```
Element     0          1      ...     n
-       n m s cl-cm  m s cl-cm  ...  m s cl-cm
```

där n är högsta index. I övrigt har varje element samma form som enkla strängar (Se $).
Ett element anropas sedan i formen n namn där n är elementindex.
Vid anrop lämnas adressen till elementets size-byte på stacken.

Anropar : ($) $ERROR $MAX

($) (n —)
Lägger n i översta lediga byten i ordlistan följt av en byte 0 och reserverar sedan n bytes i ordlistan.

Anropas av : $ $VECT

($§) (a ud —)
Omvandlar ett dubbeltal utan tecken till en textsträng som variabeln a tilldelas.
Texten mellanlagras under PAD.

Anropar : 1- $<-
Anropas av : $<-§ $<-D§

($IX1) (a p — a p b)
Testar om positions-index hamnar utanför strängen. Lämnar en boolesk flagga på stacken.

Anropar : <ROT
Anropas av : $GET $PUT

($IX2) (a p n — a p n b)
Se ($IX1).

Anropar : 2OVER <ROT 1-
Anropas av : $MID $DEL

($IX3) (a1 p a2 — a1 p a2 p a1)

($IX4) (a1 p a2 p n — a1 p a2 b)
Se ($IX1).

Anropas i formen ($IX3) xxx ($IX4) där xxx är $SIZE eller $MAX. Vid anrop av ($IX4) är n alltså size/max-värdet.

Anropar : 2OVER
Anropas av : $STORE $INS

($V) (a0 — steg gräns start)
Räknar om adressen till första elementet i en vektor till stegvärde, gränsvärde och startvärde för en DO+LOOP.
OBS! Måste anropas med element-index 0 !

Anropar : $MAX 2-
Anropas av : $VCLR $V.

Extensions

$"
Tar in text från inströmmen avgränsad med ett " och lägger den ovanför ordlistan där första byten innehåller antal tecken. HERE läggs på stacken.

Anropas av : Alla ord som har " som sista tecken.

$<"
$<"
$="
$>"
$COMP"
$CONC"
$INS"
$SEEK"
$STORE"

Dessa anropas som sina huvud-ord med den skillnaden att den andra strängen skrivsdirekt efter ordet, avslutat med ett " .

Exempel :

I uttrycket

PERSON NAMN $<-

tilldelas strängen PERSON strängen NAMN's innehåll. Om NAMN innehöll 'Pelle Pettersson' så fås samma resultat med uttrycket

PERSON $<-" Pelle Pettersson"

Följande uttryck lägger talet 14 på stacken

SUFF $<-" son"
PERSON 1 SUFF $SEEK

Utrycket

PERSON 1 $SEEK" son"

ger samma resultat.

```
§ 50
( STRÄNG 1              830226  PEM )

: $ERROR CR MESSAGE SP! QUIT ;
: ($IX1) OVER C@ OVER DUP
         1 < <ROT < OR ;
: ($IX2) DUP 2OVER SWAP C@ <ROT DUP
         1 < <ROT + 1- ROT > OR ;
: ($IX3) DUP C@ 2OVER SWAP ;
: ($IX4) OVER 1 < SWAP ROT
         - 1+ ROT < OR ;
: ($)    DUP C, 0 C, ALLOT ;
: $      <BUILDS ($) DOES> 1+ ;
: $MAX   1- C@ ;
: $SIZE  C@ ;
: $VECT  <BUILDS SWAP DUP C, 1+ 0
         DO DUP ($) LOOP DROP
         DOES> OVER OVER C@ OVER
          0 < <ROT > OR
          IF 14 $ERROR ELSE 2+ DUP
          $MAX 2+ ROT * + ENDIF ;
-->
```

```
§ 51
( STRÄNG 2              830226  PEM )

: $<-    SWAP OVER C@ OVER $MAX
         2DUP > IF SWAP THEN DROP
         2DUP SWAP C! ROT 1+ ROT
         1+ ROT CMOVE ;
: $CLR   0 SWAP C! ;
: ($V)   DUP DUP $MAX 2+ DUP ROT
         DUP 2- C@ 1+ ROT * + ROT ;
: $VCLR  ($V) DO 0 I C! DUP
              +LOOP DROP ;
: $.     COUNT TYPE ;
: $V.    CR ($V) DO I $. OVER
          IF CR ELSE SPACE THEN
          DUP +LOOP DROP ;
: ($§)   <§ §S SIGN §> OVER 1-
         C! 1- $<- ;
: $<-§   DUP ABS 0 ($§) ;
: $<-D§  SWAP OVER DABS ($§) ;

-->
```

```
§ 52
( STRÄNG 3              830226  PEM )

: $COMP  COUNT ROT COUNT ROT 2DUP
         2ROT 2SWAP MIN 0 DO OVER
         I + C@ OVER I + C@ 2DUP
         > IF 4 NDROP -1 LEAVE
         ELSE < IF 2DROP 1 LEAVE
         ELSE I 1+ I' = IF 2DROP 0
         THEN THEN THEN LOOP DUP 0=
         2SWAP ROT IF 2DUP > IF
         3 NDROP 1 ELSE < IF DROP
         -1 THEN THEN ELSE 2DROP
         ENDIF ;
: $=     $COMP 0= ;
: $<     $COMP -1 = ;
: $>     $COMP 1 = ;
: $STORE ($IX3) C@ ($IX4)
         IF 16 $ERROR ELSE
          COUNT 2SWAP + SWAP CMOVE
         ENDIF ;
-->
```

```
§ 53
( STRÄNG 4              830226  PEM )

: $SEEK  <ROT OVER CÉ OVER < SWAP
         >R >R OVER CÉ OVER CÉ
         OVER - DUP 0< R> OR
         IF R> 2DROP $ERROR
         ELSE 2+ SWAP PAD C! R>
          2DUP > 0= OVER 1 < OR
          IF 16 $ERROR ELSE
          DO DUP I + PAD 1+ PAD
           CÉ CMOVE OVER PAD $=
           IF I LEAVE ELSE I 1+ I'
           = IF 0 THEN THEN
          LOOP <ROT 2DROP ENDIF
         ENDIF ;
: $LAST  OVER DUP CÉ 1+ DUP
         >R OVER $MAX >
         IF 16 $ERROR ELSE
          COUNT + C! R> SWAP C!
         ENDIF ;
-->
§ 54
( STRÄNG 5              830226  PEM )

: $GET   ($IX1) IF 16 $ERROR ELSE
          OVER + CÉ SWAP DROP
         ENDIF ;
: $PUT   <ROT ($IX1) IF
          16 $ERROR ELSE + C!
         ENDIF ;
: $MID   ($IX2) IF 16 $ERROR ELSE
          ROT DUP 2SWAP SWAP ROT
          + SWAP DUP PAD C! PAD 1+
          SWAP CMOVE PAD $<-
         ENDIF ;
: $LEFT  1 SWAP $MID ;
: $RIGHT OVER CÉ OVER - 1+ $MID ;
: $CONC  2DUP CÉ OVER CÉ + PAD C!
         COUNT PAD 1+ 2DUP + >R
         SWAP CMOVE COUNT R> SWAP
         CMOVE PAD $<- ;

-->
§ 55
( STRÄNG 6              830226  PEM )

: $INS   ($IX3) $MAX ($IX4)
         IF 16 $ERROR ELSE
          ROT OVER CÉ OVER CÉ +
          PAD C! ROT OVER 1+ OVER
          1- PAD 1+ 2DUP + >R SWAP
          CMOVE ROT COUNT R> 2DUP
          + >R SWAP CMOVE OVER
          <ROT OVER CÉ OVER - 1+
          <ROT + R> ROT CMOVE
          PAD $<-
         ENDIF ;
: $DEL   ($IX2)
         IF 16 $ERROR ELSE ROT
          DUP CÉ 2OVER + 1- - >R
          2DUP CÉ SWAP - OVER C!
          ROT + DUP >R + R> R>
          CMOVE
         ENDIF ;
§ 56
( STRÄNG 7              830226  PEM )

: $FILL   OVER DUP $MAX OVER C! 1+
          ROT $MAX ROT FILL ;
: $BLANKS 32 $FILL ;

: $"      34 WORD HERE ;
: $<-"    $" $<- ;
: $COMP"  $" $COMP ;
: $="     $" $= ;
: $<"     $" $< ;
: $>"     $" $> ;
: $SEEK"  $" $SEEK ;
: $CONC"  $" $CONC ;
: $INS"   $" $INS ;
: $STORE" $" $STORE ;
;S
```

FrågeBITen.

Hejsan igen!

Ja, nu är jag här igen. Jag gjorde ett uppehåll i förra numret p g a att det inte hade kommit några frågor av (vad vi bedömer) allmänt intresse till oss. Istället så har jag med desto fler frågor den här gången.

Rikard Edström från Järfälla har skrivit detta brev till klubben:

Hej

Jag är en kille på 14 år som har ett antal frågor om ABC80.

1. Jag har medlemsnummer 3513. varför står jag inte med i medlemsmatrikeln för 1983?

2. Jag tycker att det är alldeles för lite program i ABC-bladet. Det vore jättebra med en stor annonssida där man kan köpa, sälja eller byta program med varandra.

3. Hur är det egentligen med en joystick till ABC80? Kan man köpa vilken som helst och plugga in eller måste det vara något särskilt märke? I så fall, hur mycket kostar en?

4. Jag har hört talas om två stora adventure-program, det ena heter STUGA och det andra heter THE CAVE. Vad för sorts dator finns de på och kan man få köra ett sådant?

5. Finns det egentligen några avancerade adventure-program till ABC80? I så fall var och vad är priset?

6. Kommer alla program på ABC-- kassetterna från ABC-klubbens programbank?

Rikard Edström
Hjälmvägen 5
175 61 JÄRFÄLLA

Svar:

1. Medlemsmatrikeln innehåller alla medlemmar mellan nr 100 och 3492. Dessa togs ut ur registret i februari 1983 men har på grund av förseningar hos tryckeriet inte distribuerats förrän på hösten.

2. En sådan "stor" annonssida har vi redan, radannonserna. Det står var och en fritt att köpa, sälja eller byta vad han-/hon vill.

3. Vilken som helst går inte att ansluta men jag vet de som har anslutit en joystick från ett Atari-spel till en ABC-80. Jag skall försöka pressa fram en artikel ur en kille som har gjort en sådan anpassning. Vad jag vet så finns det bara en joystick att köpa och det är den som Comporian AB har tagit fram och den kostar 300-400 Sek. Adressen dit är:
Comporian AB
Box 4037
580 04 LINKÖPING
Tel: 013-14 80 92

4. Adventureprogram finns på de flesta stordatorsystem. Jag vet inte var man kan kan få tag i eller spela The Cave men jag vet att STUGA finns på QZ och att man kan köra det till reducerad taxa om man loggar in på ABC-klubbens konto, dock går det endast på det dyrare kontot. För att starta STUGA när man har loggat in i QZ skriver man RUN GAM:STUGA <return> så startar spelet. Det kostar i genomsnitt 1 krona per minut att spela.

5. Det mest avancerade adventure-program jag har sett till ABC80 är UPPSJÖ som kom ut på ABC-kassett 10. Kanske någon annan har ett bättre? Hör i så fall av er.

6. Ja, det gör de. Dock så finns inte hela klubbens programbank tillgänglig från monitorn med anledning av att vår 8-tummare inte rymmer allt. När vi till sommaren får igång vårt nya monitorsystem så kommer hela programbiblioteket att bli tillgängligt.

Det har kommit ett brev från Torbjörn Andersson <3448) som lyder så här:

JAG VILL LÄRA MIG MASKINSPRÅK !

Jag skulle vilja att ABC-klubben hade någon/några sidor där man skulle kunna lära sig maskinspråk från början (med många programexempel). Det finns tyvärr inte särskilt mycket böcker om maskinspråk för ABC80 så därför tycker jag att någon skulle kunna genomföra det här förslaget.

Kom inte och säg att det inte finns någon medlem som kan maskinspråk på ABC-80 för det räcker med att titta på era kassetter för att se att det måste finnas några medlemmar som kan maskinspråk.

Det är säkert fler än jag som skulle vilja att detta genomfördes så jag hoppas att ni skulle vilja göra det.

Torbjörn Andersson

Svar:

Det finns inga medlemmar som kan maskinspråk. Nej, så är inte i verkligheten. ABC-Stockholm har funderat på att göra en assemblerkurs för sina medlemmar och sedan ge ut den som en ABC-rapport eller som en artikel i tidningen. När det hela kommer i gång kan jag inte säga.

TÄVLING

Den som före 1984-04-01 har skickat in det bästa filräddningsprogrammet vinner 100 stycken C60-kassetter av fabrikat ACME. Programmet behöver bara fungera för disk eller kassett.

Ja, det var allt för den här gången. Skicka gärna in frågor. Ett tips: Fråga gärna vad det finns för list- och kopieringsskydd till ABC80/800.

Det var allt från nästa gång, vi skrivs.

The Computer Phantome

ABC klubben har fått nytt kansli.

I slutet av förra året träffade ABC-klubben avtal med ett företag om att sköta vissa kanslifunktioner åt ABC-klubben. Där [illegible]nstgör sedan början av december Berit [illegible]tavii på vanlig kontorstid som kanslist åt klubben. I hennes uppgifter ingår bl a att ta emot och fördela inkommande post, svara i telefon, effektuera beställningar och sända ut material till nya medlemmar, handha medlemsregistret och hjälpa till med bokföringen. Tills vidare kommer Berit att arbeta [illegible]t halvtid med uppgifter för klubben.

Med det stora antalet medlemmar som ABC-klubben numera har - f n över 4000 och antalet tycks fortsätta att öka i oförändrad takt - är det inte längre möjligt att sköta klubbadministrationen på enbart ideell basis, i varje fall inte om klubben skall kunna svara upp till de berättigade krav på service som medlemmarna ställer. Det här innebär naturligtvis en ökad belastning på ABC-klubbens ekonomi och har påverkat budgeten för 1984.

Innan vi kom fram till denna lösning har vi försökt klara de löpande uppgifterna med provisoriska arrangemang. Det har tyvärr inte alltid fungerat tillfredsställande och många, särskilt medlemmar som kommit med i klubben under hösten har fått vänta orimligt lång tid på att få sitt material. Jag beklagar djupt detta och ber de som drabbats om en allvarligt menad ursäkt. Det är min förhoppning att det i fortsättningen skall fungera bättre.

Berit Gustavii

ABC-klubbens postadress är oförändrat Vidängsvägen 1, 161 33 Bromma. Kansliet har fått eget telefonnummer, 08-53 57 50. När Berit inte kan svara kopplar hon in en telefonsvarare som även kan ta emot meddelanden. På telefon 08-80 15 22 kommer liksom hittills att finnas en telefonsvarare med aktuell klubbinformation. Denna telefonsvarare kan numera inte ta emot meddelanden. Telefon nr 08-80 15 23 går liksom hittils till monitorn i Alvik. Den planerade extra ingående linjen till den nya monitorn kommer att ha telefonnummer 08-80 15 26. Om man vill nå en person som tillfälligt vistas i klubblokalen kan man ringa 08-80 17 25, i normala fall är det dock ingen som svarar på detta nummer.

Beställning av samlingsnummer och övriga publikationer skall i första hand göras genom att använda postgirokonto nr 62 93 00-5, ABC-klubben Publikationer. Ange på talongen vad du vill ha och betala in det framräknade beloppet - observera att vissa priser justerades i början av februari. Medlemsavgifter skall betalas till postgirokonto nr 15 33 36-3, ABC-klubben. Betalning av körningar på Q-Zentralen görs till postgirokonto nr 43 51 74-8, ABC-klubben Q-Zentralen.

Gunnar Tidner

HISTORIK.

ABC80

Historien om ABC80 började med att Scandia Metric i februari 1978 tog kontakt med Luxor för tillverkning av bildskärmar. Scandia Metric visade då upp en mikrodator, TRS-80. Man försökte då få Luxor [illegible]esserad av att sälja en sådan dator. Vid denna tidpunkt var mikrodatorn inte alls känd i Sverige.

Luxors dåvarande ledning var inte intresserad av mikrodatoriden. Man tog dock kontakt med Gunnar Markesjö på KTH. Gunnar Markesjö var full av entusiasm inför mikrodatoriden. Efter nya diskussioner mellan Luxor och Scandia Metric togs frågan upp om de nödvändiga kunskaperna för att konstruera en dator.

Dessa kunskaper fanns då inte inom Luxor. Scandia Metric föreslog att kontakt skulle tas med Lasse Karlsson och hans företag DIAB. DIAB och Lasse Karlsson tog hand om utvecklingen. Under projektet gick utvecklingen av minnen så fort att DIAB på eget bevåg utrustade datorn med större minne.

För att få projektet i hamn beställde sedan Scandia Metric 3000 maskiner, utan att datorn var konstruerad! Denna beställningen kom att utgöra den ekonomiska förutsättningen för projektet. Vid midsommar var en prototyp klar. Då trodde dock alla på Luxor, utom utvecklingschefen Bengt Lönnqvist, att projektet skulle misslyckas.

Mikrodatorn, som nu hade döpts till ABC80, pressvisades i augusti 1978. Då var det en fullständig nyhet att ett svenskt företag hade utvecklat en mikrodator, nästan inga kände till projektet i förväg. Det blev succe direkt. Datorn var klar för tillverkning i november samma år. Dock kunde det inte ske eftersom TEXAS Instruments inte kunde leverera teckengeneratorn i tid. Datorer tillverkades ändå, för senare montering av teckengeneratorn.

En stor roll för tillkomsten av Sveriges första mikrodator har **Gunnar Markesjö/KTH, Lars Karlsson/DIAB, Bengt Lönnqvist/Luxor och Karl-Johan Börjesson/Scandia Metric** spelat. Scandia Metric har dock senare lämnat samarbetet kring ABC-datorerna.

I januari 1979 kom Luxorkraschen och Luxor inställde betalningarna. Detta förorsakade givetvis problem för mikrodatorutvecklingen. Trots allt kom sedan tillverkningen igång.

Luxor började senare att själv tillverka flexskiveminnen. En ny mikrodatorserie presenterades våren 1981 under namnet ABC800. Våren 1983 kom ABC802 och ABC806.

I januari 1984 meddelade industridepartementet att man hade sålt Luxor till Nokia i Finland. ABC-klubben ser med spänning fram emot vad detta kan innebära för utvecklingen av Luxor mikrodatorer. Nu under 1984 väntar vi på nya kraftfulla 16 bitars mikrodatorer från det svenska datorföretaget i Motala.

ABC-klubben

Diskussioner om att bilda en användarklubb kom fram i början på 1980. Detta understöddes av Luxor och Scandia Metric, vilket ledde till några möten i början på 1980. ABC-klubben bildades den 24 januari 1980 vid ett möte på KTH, där cirka 150 personer av de dåvarande 500 medlemmarna var närvarade. Sedan dess har verksamheten vuxit enormt.

Nu finns det över 4000 medlemmar! Medlemsutvecklingen har varit explosionsartad. Detta har inneburit att ABC-klubben numera måste anlita betald kontorshjälp för att få administrationen att fungera. Klubben är helt enkelt för stor för att den helt skall kunna drivas med ideelt arbete.

Bo Kullmar <1789>

Kort och Brett om

DEC-10.

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

Skriften kommer att publiceras efter hand i mån om plats.

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

ABC80 SOM RAMMASKIN.

När man har ABC80 ombyggd som rammaskin alltså med 64K adresserbart minne, så får man en rad nya möjligheter. Dessa möjligheter har alla samband med att man på olika sätt kan gå in och ändra i BASIC-tolken. En av de mer intressanta möjligheterna är att man kan ändra tidskonstanterna i kassettrutinen. För den som inte har floppy är denna möjlighet extra intressant eftersom man kan få snabbare inläsning av program. Tidskonstanterna för kassettrutinen ligger på de decimala adresserna 1144, 1153 och 1439 om man har check-summa 11273, annars ligger de på adresserna 1153, 1162 och 1439.

Med en av följande små snuttar kan du lekvämt ändra tidskonstanterna:

```
1 REM för check-summa 11273
10 INPUT A
20 POKE 1144,A
30 POKE 1153,A
40 POKE 1439,A*.8

1 REM för checksummorna 10042 & 9913
10 INPUT A
20 POKE 1153,A
30 POKE 1162,A/2
40 POKE 1439,A*.8
```

Tidskonstanterna är i hårdvaruversionen av ABC80 för checksumma 11273 150,150 och 120 det vill säga de två första har samma värde medan den tredje är 4/5 av detta värde. Tidskonstanterna är för de övriga checksummorna 150, 75 och 120. Man har helt enkelt tagit bort en skiftinstruktion, det vill säga en division med 2, när man ändrat i Basic-tolken.

Att ändra inspelningshastighet

Den normala överföringshastigheten till kassett är för ABC80 700 Baud och den hastigheten uppnås när parametrarna har standardvärdena 150, 150 resp 120. Det finns alltså utrymme för både långsammare och snabbare kassettrutin. Jag har gjort försök med båda delarna. Min utrustning är varit en ABC80 med checksumma 11273 och den gamla ABC80-bandspelaren och en ordinär kassett av samma typ som ABC-kassetterna. Vilken ramombyggnad man har spelar sannolikt ingen roll. Jag har en utan ram-floppy.

Läsning och skrivning i hastigheter lägre än 700 Baud

Jag ska börja att berätta om mina försök med att sänka hastigheten på kassettrutinen. Många ställer sig antagligen frågan: Vad ska det vara bra för? Frågan är högst begriplig, kassettrutinen går tillräckligt långsamt redan nu. Det finns knappast någon stark opinion som kräver långsammare kassetter. Det var ren och skär nyfikenhet som fick mig att försöka sänka kassettrutinen genom att mata in värdet 255 i programsnutten ovan. När jag gjort det läste jag in ett program från disk, utan flexskiveenhet hade det blivit besvärligt, sedan jag ändrat parametrarna kan jag ju inte läsa in standardkassetter.

Programmet sparades på kassett och när det var klart backade jag och laddade in programmet med load. Det lät dovt och ovant, men hast du mir gesehen FOUND .BAC det hade lyckats! Överföringshastigheten borde vara så där en 412 Baud. Det var emellertid nu som de egentliga experimenten började. Jag ville alltså ta reda på hur känslig kassettrutinen är för variationer i hastighet hos bandspelaren. Eftersom det är svårt att variera hastigheten hos bandspelaren på ett kontrollerat sätt är det bättre att variera hastigheten på kassettrutinen.

Kassettfilen som jag skulle försöka läsa in var alltså inspelad med 412 Baud och tidskonstanterna var vid inspelningen i ordning 255, 255 och 204. Hur mycket skulle jag våga ändra? Jag började försiktigt med ett par procent och matade in 250 i programmet för ändrad skriv- och läshastighet på kassett. Det gick bra och jag gjorde ett nytt försök med 240, vilket också gick bra. Nu blev jag djärvare och provade med 230, 210, 200, 190, 170 och 165. Det blev inte stopp förän vid 165, allt annat gick perfekt. Det innebär att kassettrutinen gick drygt 30% snabbare vid det sista lyckade försöket än den gjorde då jag skrev in test-filen på kassetten.

Vi som arbetat med ABC-kassetterna inom klubben vet att det så kallade fönstret är relativt stort, det vill säga kassettrutinen är relativ okänslig för variationer i hastighet hos bandspelarna. Det är tur annars skulle det gå mycket sämre att distribuera program på kassett. För att få en uppfattning om fönstrets storlek bör man variera hastigheten vid inläsning så att man prövar med såväl snabbare som långsammare kassettrutin. Den första test-filen var inspelad med långsammast tänkbara kassettrutin och någon möjlighet att pröva inläsning med en ändå långsammare förelåg alltså inte. Jag tog då en ABC-kassett nr 10 och satte kassettrutinen på en något högre hastighet med hjälp av ändringsprogrammet i vilket värdet 140 matades in.

ABC-kassetterna är inspelade med 700 Baud och alltså standardvärden på tidskonstanterna dvs 150, 150 och 120. Jag började alltså med att snabba upp rutinen i förhållande till testfilen.

Det första försöket gick bra, jag fortsatte med värdena 130, 120, 110 och 105. Det sista, 105, gick inte alls, de övriga gick perfekt. Nu var det alltså dags för den andra delen av försöket, att försöka läsa med en rutin som går långsammare än den rutin som användes vid inspelning. De värden jag provade var 160, 165 och 170. Det gick bra till och med 165 sedan var det stopp. Det betyder att jag kunde läsa med värden mellan 110 och 165, vilket faktiskt innebär en hastighetsskillnad om ca 50%.

Att skriva och läsa i mer än 700 Baud

Det är framför allt möjligheten av en snabbare kassettrutin som hägrar för de som hängt med ända hit, eller hur? Jag tror inte ni ska bli besvikna. Enligt en artikel i Mikrodatorn skulle det vara möjligt att snabba upp kassettrutinen till det dubbla med denna metod, jag har lyckats att få den att gå med nästan fyrdubbla hastigheten. Dubbla hastigheten får man genom att mata in 75 i ändringsprogrammet och det är inga större problem med att läsa tillbaka från kassetten. Svårigheterna kommer med högre skriv- och läshastigheter. Jag var alltså benägen att ge upp efter flera timmars experimenterande med skrivhastigheter över 1500 Baud. Det var först sedan jag funderat över resultaten från de tidigare försöken som jag kom på en möjlighet. Det hade ju visat sig att det inte var någon nackdel om kassettrutinen vid läsning var snabbare än den varit vid skrivning på kassetten. Varför inte göra ett försök? Tanke och handling äro som bekant ett hos data-amatören, varför jag genast grep mig verket an. Jag ändrade kassettrutinen genom att mata in 60 och skrev sedan in en fil på kassetten. Därefter laddade jag på nytt in ändringsprogrammet och gav 50 som invärde. Det var nu mycket spännande att se om programmet skulle gå in. Jag gav kommandot load cas: och fick FOUND efter en stund. Det var hastighetsrekord. Nu fortsatte jag att systematiskt pröva skrivning och läsning i höga hastigheter. Hela tiden gjorde jag så att jag lät kassettrutinen vid läsning vara ca 20% snabbare än vid skrivning på kassett. Sista lyckade försöket gjordes vid skrivning med kassettrutin-parametrarna 40, 40 och 32, följt av läsning med motsvarande parametrar satta till 33, 33 och 26. Skrivhastigheten bör då ha varit ca 2600 Baud. Det är naturligtvis inget tvång att gå på marginalerna och en lagom hastighet är 2400 Baud. Jag rekommenderar att man håller sig till denna hastighet, vilket man gör om man matar in 44 i ändringsprogrammet före skrivning och 37 före läsning. Dessa värden är naturligtvis bara förslag, men på sikt kan det finnas skäl att komma fram till en standard så att vi som har ram-maskiner kan kommunicera med varandra på kassett.

Säkrare läsning och skrivning på kassett utan internklocka

Genom att stänga av klockan i ABC80 får man något färre fel vid läsning från kassett och kanske också snyggare fyrkantvåg på sina kassetter. Det är mycket enkelt att stänga av klockan, det räcker med att poka in en enda instruktion i Basic-tolken. På adress 102 (decimalt för alla checksummorna) lägger man in instruktionen RETN, return from nonmascable interupt. Detta sker med POKE 102, 237,69 varvid klockan stannar. Den kan startas på nytt med POKE 102,24,223. När klockan stängs av hoppar ABC80 över klockrutinen och processorn får mindre jobb.

Ingen kompatibilitet med ABC800

De kassetter som spelas in med förhöjd hastighet på det ovan beskrivna viset kan ej läsas in utan vidare av en ABC800. Det beror på att kassettrutinen i ABC800 arbetar med ett inledande block vars hastighet är 700 Baud. I detta block finns uppgift om hur fort kassettrutinen överför de följande blocken. När man snabbar upp kassetterna på det här beskrivna viset går överföringen med samma hastighet hela tiden. Det spelar naturligtvis ingen större roll för den som bara vill kunna spara och ladda in sina egna program snabbare. Problem uppstår först om man vill använda denna metod exempelvis för att få plats med fler program på en ABC-kassett.

Kjell-Åke Johansson

Programmering BASIC II.

BASIC II har några ingredienser som gör det mycket enkelt att skriva strukturerad kod. Långa namn på variabler och funktioner, WHILE loopar och flerradiga funktioner. Om man använder en programutvecklingsmetod som JSP så förenklas själva kodningen av BASIC II. Dock vill jag för att undvika missförstånd påpeka att JSP kan tillämpas oberoende av programspråk.

Namn på variabler och funktioner.

I BASIC II kan man använda långa namn på variabler och funktioner. Just långa namn på variabler och funktioner gör att BASIC II-programmen blir mycket enklare att förstå och skriva än program i BASIC-dialekter med enbart korta variabelnamn.

Det finns egentligen ingen begränsning på längden av ett variabelnamn. Dock begränsas det ändå eftersom man inte kan mata in mer än 160 tecken på en rad. Alla långa namn på variabler och funktioner finns i en lista längst bak i programmet. Detta gör att man enbart får obetydligt längre kod med långa variabelnamn än med korta dito.

WHILE-satsen

Normalt använder man sig flitigt av "read ahead"-tekniken om man utnyttar JSP. "Read ahead" står för "läs i förväg" och kan användas även om man inte tillämpar JSP. Läs i förväg innebär att man alltid läser en post i förväg. Detta gör att man t ex alltid klarar av att behandla tomma filer utan att speciellt ta hänsyn till det. En enkel tillämpning av "read ahead"-tekniken visar jag här i ett exempel där jag läser tal från tangetbordet och adderar dem tills man matar in 0 (=noll). Jag bortser från radnummer.

```
Summa = 0
INPUT Tal ! Read ahead
WHILE Tal<>0
  Summa = Summa + Tal
  ; Summa
  INPUT Tal ! Read ahead
WEND
```

Detta innebär att man måste skriva lässatsen på två ställen, men det gör ju inget. En evighetsloop kan utryckas med en WHILE-loop på ett snyggt sätt. Villkorert -1 är ju alltid sant!

```
WHILE -1
  Z=FNDemo
WEND
```

Flerradiga funktioner

GOSUB-satsen i BASIC är inte särskilt bra eftersom den är bunden till ett radnummer som inte avsöjar särskilt mycket av vad som skall utföras. Det är också lite knöligt att skicka med parametrar tillsammans med ett anrop av en subrutin med GOSUB. Dessutom är anrop till flerradiga funktioner snabbare än motsvarande anrop med GOSUB.

Flerradiga funktioner i BASIC II är så bra att man normalt använder dem i stället för anrop med GOSUB. Även om det inte är frågan om riktiga funktioner, dvs man har inget värde och skicka med och inget värde returneras så använder man fleradiga funktioner.

Observera att man inte behöver låta programmet genomlöpa koden i en funktion innan man kan använda den. Man behöver alltså inte som i Pascal först skriva sina funktioner och sedan huvudprogrammet.

En annan fördel med flerradiga funktioner är att man kan använda lokala variabler i en flerradig funktion. Det innebär att de lokala variablerna enbart finns vid själva anropet till funktionen. Motsatsen kallas för globala variabler som gäller i hela programmet. En global variabel t ex variabeln Summa är alltså inte samma sak som den lokala variabeln Summa.

Typen på namnet för en flerradig funktion måste vara samma som returvariabeln, dvs skall funktionen returnera en sträng så måste funktionen t ex ha namnet FNTest$. En flyttalsfunktion kan dock returnera ett heltal och tvärt om.

Man får inte lov att tilldela en strängvariabel, som används som inparmeter, något värde i funktionen, dvs "insträngen" får ej stå till vänster om ett "="-tecken. Detta resulterar i felmeddelande nr 191, ej tilldelningsbar i funktionen:

```
; FNAdd$('ABC')
DEF FNAdd$(Sträng$)
  Sträng$ = Sträng$ + '-Klubben'
  RETURN Sträng$
FNEND
```

I stället får man skriva funktinen så här:

```
DEF FNAdd$(In$) LOCAL Sträng$=80
  Sträng$ = In$
  Sträng$ = Sträng$ + '-klubben'
  RETURN Sträng$
FNEND
```

I det senare och bugfria (felfria) exemplet används en lokal variabel. Det behövs inte för variabeln In$ och i det felaktiga exemplet eftersom inparametern alltid är lokal. Notera också att man måste deklarera längden på lokala strängar (här har jag tagit till kraftigt, mer än vad som behövs för exemplet).

Man kan med fördel använda flerradiga funktioner när man egentligen behöver en procedur som inte skall ha några parametrar varken in eller ut. Då anropar man funktionen utan parameter och returnerar en tom sträng eller, om ett tal skall returneras, så skickar man tillbaka 0 (=noll).

Två exempel på detta:

```
DEF FNSkriv
  ; 'Detta är ett exempel på en
  funktion'
  ; 'som inte returnerar något värde'
  RETURN 0
FNEND
```

```
DEF FNSkriv$
  ; 'Detta är ett exempel på en
  funktion'
  ; 'som inte returnerar något värde'
  RETURN "
FNEND
```

Egentligen spelar det ingen roll vilken typ ovanstående funktion är, men skall den anropas med ;FNSkriv så spelar det roll. I det första exemplet skrivs då noll eftersom funktionen returnerar detta. Om man anropar funktionen FNSkriv$ så skrivs givetvis inget alls. Egentligen skrivs det en tom sträng, men den syns ju inte.

Man kan låta en funktion returnera ett booleskt värde, dvs sant eller falskt (-1 eller 0). Sedan kan man anropa funktionen i en IF-sats.

```
INTEGER
IF FNUdda(Tal) THEN ; 'Talet är udda'
ELSE ; 'Talet är jämt'
DEF FNUdda(Tal)
  IF (Tal AND 1) = 1 THEN RETURN
  -1 ELSE RETURN 0
FNEND
```

Notera också att om man använder JSP-teknik så gör det absolut inget om man har flera uthopp ur en funktion. Att man tidigare inte ville ha det berodde det nog mest på att vanliga flödesscheman då blir lite grötiga.

I de första versionerna av BASIC II-tolkarna kunde man hoppa ut ur flerradiga funktioner med GOTO, GOSUB och ON ERROR GOTO satser. Detta kunde dock medföra att datorn dök i vissa fall. Därför kan man inte på senare versioner av BASIC II hoppa ut ur en flerradig funktion med någon form av GOTO-hopp. Felhanteraren är alltid lokal i en funktion.

En loopvariabel för en FOR-NEXT-loop kan dock inte vara en lokal variabel eftersom FOR-NEXT-loopen är optimerad så att den skall vara snabb. För att en loopvariabel skall var lokal så kan man i stället använda sig av en WHILE loop.

```
DEF FNLoop LOCAL I
  FOR I=1 TO 10
    ; I
  NEXT I
  RETURN I
FNEND
```

Detta är alltså en felaktig funktion och felmeddelande nr 186. Man kan i stället skriva så här:

```
DEF FNLoop LOCAL I
  WHILE I < 10
    I = I + 1
    ; I
  WEND
  RETURN I
FNEND
```

Denna loop blir dock obetydligt långsammare än det första exemplet. Det första exemplet är körbart om I inte är en lokal variabel. Man kan alltså behålla FOR-NEXT-loopen om man gör I till en global variabel. Notera att I ej behöver nollställas i exemplet med WHILE-loop eftersom det är en lokal variabel som alltid nollställes vid anropet. Variabellistan för lokala variabler finns enligt uppgift på stacken och då enbart under själva anropet.

Glöm slutligen inte att använda radnummer om du själv vill testa några exempel på en dator. Eftersom det inte har någon betydelse så har jag utelämnat dem, men jag vill också visa att man för logikens skull egentligen klara sig utan dem.

Bo Kullmar <1789>

och användardefinierade funktioner

I ett par tidigare artiklar har jag behandlat dels den s k Boole'ska algebran och dels behovet av bra rutiner för att ta emot svaret JA/NEJ. Det finns en inte tidigare redovisad kombination av dessa båda där man utnyttjar funktioner.

Användardefinierade funktioner

I ABC-Basic finns åtskilliga funktioner färdiga. Det är bl a olika matematiska uttryck och metoder för att hantera teckensträngar:

Y=SQR(X) beräknar kvadratroten ur X och lägger resultatet i Y.

B$=LEFT$(A$,3) tar A$:s 3 första tecknen och lägger i B$.

Särskilt när man arbetar med beräkningar kan man ha god nytta av att själv definiera någon speciell funktion. Man skriver då DEF (för definiera) FN (för funktion) och lägger till funktionsnamnet, på vanligt sätt en bokstav, ev följd av en siffra, ev följd av ett % för att beteckna heltal. Även indexerade funktioner kan användas: D(X,Y) (I ABC80 kan man däremot inte definiera strängfunktioner.)

Basictolken skriver sedan ihop delarna: DEF FN A blir DEFFNA.

Några exempel:

DEFFNA=B+C - man definierar FNA (funktionen A) som summan av B och C

DEFFND(X)=X*X - kvadraten av X läggs i FND(X)

DEFFNE(Y,Z)=SIN(Y)+COS(Z) - sinus för y plus cosinus för z läggs i FNE(Y,Z)

Dessa definitioner måste skrivas någonstans i programmet. Liksom DATA-satser kan de placeras var som helst. Somliga sätter dem sist, andra vill ha dem först. Man kan också sätta dem strax före den plats där de första gången används.

Vid kommandot RUN lägger datorn upp alla definitioner någonstans och sedan är det bara att använda dem:

Z=FND(3)+5*FNA

Här beräknas kvadraten av 3, sedan 5 gånger FNA som är summan av B och C. Alltsammans läggs ihop och placeras i variabeln Z.

Allt detta är inte så märkvärdigt när man väl vant sig vid de användardefinierade funktionernas något invecklade utseende.

JA eller NEJ

Men nu kommer det intressanta. Om man definierar en heltalsfunktion, t ex FNJ%, kan man använda den som en logisk variabel, dvs en som har värdet 0 (falskt) eller -1 (sant). Som funktionsdefinition kan man skriva vad som helst som datorn kan tolka som sant/falskt.

DEFFNJ%=(W$='J')

Vad betyder detta? Jo, om strängen W$ har innehållet J, då är det som står inom parentes sant, alltså FNJ%=-1%. Annars är det falskt dvs FNJ%=0%.

Man behöver inte ens ha parentes, men jag har med den för tydlighetens skull.

Vad använder man det här till?

Låt oss ta som exempel ett program där en lång rad frågor skall besvaras med JA eller NEJ:

```
---
100 ;'Skall printer användas (J)';:INPUT W$
---
110 ;'Stopp vid sidbyte (N)';:INPUT W$
---
300 ;'En omgång till (J)';:INPUT W$
---
```

Som visades i en tidigare artikel bör man gardera sig mot att användaren skriver små bokstäver i stället för stora. Vidare kan man snabba upp användingen genom att låta enbart RETURN betyda att man accepterar det förval som visas inom parentes.

För att slippa skriva den långa raden:

IF w$='J' OR W$='j' OR W$='' then --- med olika variationer tre gånger i ovanstående programbit kan man göra så här:

```
80 DEFFNJ%=(W$='J' OR W$='j' OR W$='')
90 DEFFNN%=(W$='N' OR W$='n' OR W$='')
```

Då har man definierat ett JA- och ett NEJ-alternativ. Resten fixas t ex så här:

```
110 IF NOT FNJ% THEN 200
120 REM här kommer rutin för att koppla in printern
---
210 IF NOT FNN% THEN 300
220 REM här kommer rutin för att stoppa
efter varje sida
---
320 IF FNJ% THEN 10 :REM från början
330 END
```

Rad 100 förutsätter att man vill ha printer inkopplad och väntar svaret J. Om man svarar med ett J, ett j, eller bara RETURN (som ju ger en tom sträng), blir parentesraden i 80 sann, dvs FNJ%=-1%. I så fall blir NOT FNJ% falskt och hoppet till rad 200 kommer inte att utföras.

Med andra ord: Om man inte svarar JA på rad 100, sker hopp till rad 200, annars utförs printerrutinen i rad 130.

På samma sätt förutsätter rad 200 ett NEJ-svar. Om den får något annat blir NOT FNN% sant och ger hopp till 300.

Rad 300, slutligen, förväntar ett JA. Om detta stämmer blir FNJ% på 320 sant och återhopp sker till början av programmet. I annat fall fortsätter rad 330, och man avslutar.

Program för andra

Att skriva program bara för eget bruk ar mycket lättare än att skriva för andra. Man vet ju vad progrmmet uträttar, vad frågorna betyder. Man vet vad den egna printern heter, och man kanske är nöjd med vissa fasta rubriker på utskrifterna.

Så fort progrmmen skall köras av andra människor och på andra utrustningar uppenbarar sig behovet av att kunna göra små förändringar och anpassningar.

För att fortsätta programexemplet ovan:

```
130 P$='PR:VSA30A72.5':P%=1%
140 ;'Vad heter printern (' P$ ')';:INPUT W$
150 IF W$<>'' THEN P$=W$
---
200 OPEN P$ ASFILE P%
```

Här definieras printernkoden på rad 130, vilket visas upp som förval på rad 140. Om man svarar RETURN är allt OK. Villkoret på rad 150 är ej uppfyllt, och programmet går vidare till nästa rad.

Gillar man inte printerkoden utan skall ha t ex PR: i stället, så skriver man den och saken är fortfarande OK. Då är villkoret uppfyllt, och W$ blir den nya printerkoden som överförs till P$.

Idiotsäker?

Man vill gärna göra sitt program idiotsäkert, men det finns gränser för vad maskinen klarar. När programmet kommer till rad 140 och visar upp förvalsvärdet på printerkoden, kanske operatören tänker: JA den är rätt, och skriver J i stället för att bara trycka RETURN. Följden blir programavbrott när datorn förgäves försöker öppna en fil som heter 'J'...

Sånt händer, men det kan botas med

```
150 IF NOT FNJ% THEN P$=W$
```

Nu godtas både den tomma strängen och J som sant i FNJ% och därmed passeras rad 150 utan åtgärd, eftersom NOT FNJ% då blir falskt.

(En liten komplikation är att ABC80 faktiskt tolkar både mellanslag, enkel-blipp (') och dubbel-blipp (") som tom sträng. Men det gör kanske mindre i detta fall eftersom printerkoden inte kan vara något av dessa tecken.)

Värre är att operatören, vid val av annan printerkod, kan skriva fel: t ex "PT;" i stället för "PR:". Man behöver en extra omgång som visar upp den nya koden för kontroll:

```
150 IF NOT FNJ% THEN P$=W$:GOTO 140
```

Nu vandrar programmet runt tills man godkänner koden med J eller bara RETURN.

Varför P%?

Varför öppnar jag inte printern som fil 1%, utan går omvägen över P%? Jo, det är enkelt. Då kan jag skriva de rader som ger utskrift så här:

```
;§P% 'Här kommer en utskriftsrad'
```

Om Nu printerrutinen har körts är P%=1% (eller något annat lämpligt tal från 1 till 255) och utskriften sker på den valda kanalen, i detta fall skrivaren.

Men om man valt bort skrivaren är P%=0%. Kanal noll är skärmen, varför utskriften i vanlig ordning hamnar på den. Vi kan alltså med smma programrad få utskriften på skärm eller skrivare.

Jaså, du vill alltid ha utskrift på skärmen också? Det var värre. Då måste det varje gång finnas en rad

```
; 'Här kommer en utskriftsrad'
If P% then ;§P% ' Här kommer en utskriftsrad'
```

Nu blir det alltid skärmutskrift enligt första raden. Om P% inte är noll blir villkoret P% sant och andra raden skriver på printern.

ABC80 tolkar nämligen alla värden skilda från noll som "sanna". Men det är bara värdet -1% som man kan ändra till noll med (P%=NOT P%).

Sven Wickberg

Skyddad programvara.

Juridik

Idag finns det inget riktigt skydd för programvara i Sverige. Klart är att patentlagstiftningen inte kan tillämpas på programvara. Jurister hävdar dock att man bör kunna tillämpa lagen om upphovsmannarätt (copyright). Copyright hindrar dock inte kopiering för privat bruk enligt andra jurister. Jämför rätten att för privat bruk kopiera t ex en sida ur en bok. Några rättsfall finns dock inte i Sverige som kan ge ledning. Detta oklara rättsläge gör att jurister rekommenderar programmakare att skydda sina program tekniskt.

Olika typer av tekniska skydd

Det finns främst två typer av skydd, nämligen listskydd och kopieringskydd. Ett 800:a program som är behandlat med Luxors program Squeeze får effekten av listskydd, men är rent tekniskt inte det.

Kopieringsskydd

Ett kopieringsskydd förhindrar inte att man kopierar en flexskiva. Däremot gör skyddet att man inte kan köra ett program från en kopia! Man kan alltså inte ens själv ta en backup-kopia. Det går inte heller att köra programmet från en winchesterdisk, eftersom ett program som har kopierats över till den fasta winchesterdisken blir okörbart. Kopieringskydd kan göras genom att man skriver information på skivan mellan sektorer och eller spår. Om man tar en vanlig helkopia av en disk så får man inte med denna speciella informationen eftersom helkopian tas enbart av normala sektorer/spår med hjälp av DOS:et. För att kontrollera kopieringskyddet läses den speciella informationen på skivan. En annan skyddsmetod är att märka skivan fysiskt genom att t ex sticka hål i den.

ABC-klubbens policy

ABC-klubben medverkar inte för att sprida metoder för att lösa upp kopieringsskydd. ABC-klubben är dock emot att kopieringsskydd används eftersom detta försvårar för våra medlemmar att använda sådan programvara.

Kopieringsskydd eller inte?

Måga använder sig inte av något kopieringsskydd. Detta har aldrig använts inom stordatorbranchen. Anledningen till att man använder kopieringsskydd är givetvis att man är rädd för att förlora betydande intäker i utebliven försäljning av programvara. Om man inte använder kopieringskydd innebär detta, såvitt jag förstår, att man gör bedömningen att en ev piratkopiering inte kan påverka intäkterna vid försäljning av programvaran.

Företag piratkopierar sannolikt inte program. En piratkopiering innebär ju att man inte får tillgång till dokumenation och nya versioner av programmet. Alla standardprogram som säljs till ABC800-datorer är normalt kopieringskyddade. Detta beror, tror jag, på historiska skäl. Luxor började med den numera hemdatorklassade datorn ABC80. För privatpersoner finns det mycket större intresse för att få programvara gratis. Trots att Luxor numera säljer flertalet program till företag så håller man fast vid de gamla principerna.

Hur många exemplar skall ett företag köpa?

I ett fleranvändarsystem använder flera användare samma program. ABC800/80 är ju inte några fleranvändarsystem. Frågan ställer sig då hur många exemplar av t ex ORD800 skall storföretaget X köpa in? För varje köp så får man en backup-kopia, vilket alltså räcker till två maskiner. Är man dessutom snabbfotad och maskinerna är väl samlade så kan kanske dessa två kopior räcka till fler än två maskiner. Dagens kopieringsskydd löser ju frågan, eftersom man inte får mer än två ex per program. För framtida mikrodatorer kan problemet dock bli akut eftersom dessa är fleranvändarsystem som normalt använder sig av en fast winchesterdisk.

Knyt programvaran till datorn?

En bra lösning på problemet med kopieringskydd är att strunta helt i det. Däremot kan man om man vill införa ett hemligt identifikationsnummer i hårdvaran (datorn). Då kan man göra så att man kan kopiera programvaran och även lyfta över den till en winchesterdisk. Man kan då förhindra att programmet körs på en annan dator. Klart är naturligtvis att detta skyddet såväl som alla andra skydd kan kringgås. Denna typ av skydd tillämpas för programvara som är speciellt anpassad för CAT-NET.

Listskydd

Genom listskydd förhindrar man att programmet kan listas. Listskydd kan kombineras med kopieringsskydd, men i den del fall är programmen inte listskyddade utan bara kopieringskyddade. Om programmet inte är listskyddat, men kopieringskyddat så kan man normalt ändra det om man lagrar det på orgi- nalskivan.

Ett BASIC-program lagras i en ABC-dator i en semikompilerad. form. Detta är en kod som kan sägas vara dett slags objektkod. Ett program som lagras i semikompilerad form kallas också för BAC-program (BAsic Code). För att lista ett BAC-program sker en tolkning av koden för att komma fram till den text som ursprungligen skrevs in. Denna rutin brukar man kalla för tillbakalistning. Vill man undvika att ett program skall listas så kan man antingen förhindra att en sådan tillbakalistning kan ske eller se till så att tillbakalistningen spårar ur när den görs.

Listskydd för ABC800-datorer

Om man ett-ställer bit nr 1 i adress 65318 så kommer 800:ans operativsystem att förhindra listning. De enda kommandon som man då kan ge är LOAD, RUN och NEW. Alltså fungerar inte POKE och PEEK. Man kan inte heller göra RESET och ha kvar programmet, eftersom datorn vid RESET skriver nollor i hela minnet. Vill man listskydda ett program så bör man ett-ställa ovan nämnda bit på fil. Den finns då i programblockets 3:e byte (PRSTAT).

Tro nu inte att den ovan nämnda enkla metoden för att låsa upp listskydd fungerar på mera kvalificierade listskydd. En annan metod är att se till så att listningen går snett när den utföres. Man kan t ex genom att skriva direkt i programmet/variabelslistan ändra namn på en variabel till t ex ASCII-tecknet 12. Detta får alltid den effekten att skärmen blankas när den variabeln listas.

Squeeze för ABC800 datorer

Luxor säljer ett programpaket som heter Squeeze. Detta paket är till för att komprimera BAC-koden genom att ta bort onödig information som t ex radnummer och variabelnamn. Detta gör att koden tar mindre plats både i minnet och på fil. Samtidigt som ett program behandlas med Squeeze får man dessutom ett mycket effektivt listskydd på köpet.

Tar man bort alla radnummer och variabelnamn så är det helt enkelt omöjligt att lista programmet. Squeeze är faktiskt ett ganska effektivt listskydd. Naturligtvis går det att knäcka ett sådant program rent teoretiskt,

Med en ny version av Squeeze kan man länka ihop tidigare semikompilerade program till ett nytt program.

Bo Kullmar <1789>

Om 80 till 800 och tvärtom

Mycket har skrivits om BASIC II till ABC80, men aldrig något om BASIC I till ABC800-familjen. Många som "byter upp sig" saknar sina gamla program från ABC80. Det kan gälla spel eller andra tillämpningar.

Skulle det inte vara möjligt för ABC-klubben att **ta fram en BASIC I-tolk för** ABC802 att laddas in ungefär som CP/M i RAM-minnet?

Om inte detta går att realisera vore det önskvärt med fler översättningar av t ex spelprogram till 800-format.

Lennart Eliasson, <1914>

En medlem har gjort ett program som gör det möjligt att ladda in ABC80:s BASIC I-tolk i ABC800C. Programmet heter X800X80.800 och finns på klubbens monitor. Med det programmet kan man dock enbart använda bildskärmen och tangentbordet. DOS:et är inte ändrat, för övrigt så kan man inte ändra det, eftersom grafikminnet i ABC800C enbart är 16 kbyte stort. Bildminnet är också organiserat på samma sätt i ABC800C som i ABC80, vilket såvitt jag förstår underlättar det hela.

Om någon medlem vill försöka enligt Lennart Eliassons förslag att göra en rutin så att man kan köra BASIC I i 800:or så kan det tänka sig att ABC-klubben kan lämna ekonomiskt stöd i form av projektstöd. Allra bäst är om man kan köra BASIC I på alla versioner av 800:or. Då krävs dock expanderat minne för att detta skall vara möjligt för ABC800C- och M-maskinerna.

Att göra ABC800-versioner av ABC80-program kan vara svårt. En del 800:or saknar t ex lågupplösnings- och ASCII-grafik. Andra medlemmar saknar en ABC800 och kan därför inte göra 800:a versioner av sina program. Jag kommer att föreslå att klubben köper en ABC802:a för att låna ut den till medlemmar som vill göra konvertera ABC80 program till ABC800.

Bo Kullmar

När jag de senaste gångerna har pratat med Bo Kullmar så har han berättat han han gått en kurs i Unix och hur bra det är. Han rättade bland annat om en rutin som heter LEAVE och som påminner användaren en valfri tid att det är dags att lämna terminalen "för nu börjar det bli sent!".

Det verkade ju bra! Varför skulle det inte gå att implementera det på ABC800? Hmm... Det där satte sig i huvudet på mig så jag kunde ju inte somna den natten förrän 4-tiden. Äntligen! Det fungerade! Trött som en jag-vet-inte-vad gick jag och lade mig, ganska nöjd med mig själv.

Rutinen läggs fr o m adress 64256, d v s dosbuf 6 & 7. Rutinen markerar själv att dosbuf 6 & 7 är upptagna i dosfilbeskrivningarna. Det gör att man endast kan öppna 6 filer samtidigt i stället för 8. Programmet går lika bra att starta från ett basicprogram som från en ABS-fil.

När man laddat in programmet i minnet kan från basic (ABS-filer startar av sig själva) kan man starta LEAVE genom att skriva "; CALL(64256)". Man får då till svar: "When do you want to leave (HHMM) ?". Då matar man in den timme och minut man vill bli påmind. Om man matar in en felaktig tid så får man ett felmeddelane "Time error!!".

Antag att vi matat in tiden 2055. När klockan är 2055 så får vi på skärmen meddelandet "It's time to leave now!!!". Det meddelandet får vi varje ny minut, d v s 2056, 2057 o s v.

Om vi vill stänga av rutinen, antingen under väntan på "väckning", eller under tiden mellan två "väckningar", så kan vi skriva "; CALL(64259)". Vi kommer då inte att besväras av några nya meddelanden.

Jag har testat och fått fram att internklockan i ABC800 tyvärr drar sig en aning när man har rutinen inlagd. Jag har experimenterat lite grand men inte fått till den rätta ruckningsrutinen. Däörför så finns det ingen sådan med.

Det kan tyckas att assemblerlistningen inte ser normal ut men det beror på att jag använder en assemblator som heter ASMZ. Den första raden står det ZPROG LEAVE på och den kan uteslutas i andra assemblatorer. I ASMZ så behöver man inte ha semikolon (;) framför kommentarer vilket man måste i alla andra assemblatorer.

Hoppas att allt går i bra

Kalle Lindström

```
LEAVE   ZPROG Command 'LEAVE'
;
        ORG   64256             Start address
;
OUTLI   EQU   11                Subroutine to write text on
                                screen
INLINE  EQU   5                 Get text from keyboard
TIME    EQU   65522             Position to hours
INTVECT EQU   65494             Address to interuptvector
;
START   JP    INIT
        JP    LEAVEIT
;
INIT    LD    HL,WHEN           Initialation text
        LD    BC,WHENSZ-WHEN    Text lenght
        CALL  OUTLI             Write text on screen
        LD    BC,10             Maximal string lenght
        LD    HL,BUFFER         Buffer for time to leave
        PUSH  HL                Save begin of buffer
        CALL  INLINE            Get time from keyboard
;
        POP   HL                Get begin of buffer
        LD    DE,BUF1           Buffer to save time in
        CALL  ASC2I             Konvert ASCII to integer
        CP    24                Check if hour alright
        JR    NC,WRONG          No, it was wrong
        LD    (DE),A            Save hours
        INC   DE                Increment buffertpointer
        CALL  ASC2I             Konvert minute
        CP    60                Check if minutes alright
        JR    NC,WRONG          No, minutes was wrong
        LD    (DE),A            Save minutes
        LD    HL,OK             Text to prompt
        LD    BC,OKSZ-OK        Size of text
        CALL  OUTLI             Write text on screen
        LD    A,255
        LD    (ALARM),A
        LD    (OLDMIN),A        Reset minute for alarm
        LD    HL,(INTVECT)      Get old interuptvector
        LD    (OLDINT),HL       Save old interuptvector
        LD    HL,NEWINT         Adress to new interupt
        LD    (INTVECT),HL      Save new interuptvector
        LD    HL,64967          Address dosdescription 6
        INC   A                 A: is now = 0
        LD    (HL),A            Mark dosbuf busy
        LD    HL,64945          Adress dosdescription 7
        LD    (HL),A            Mark dosbuf busy
        RET
;
WRONG   LD    HL,WRTXT          Error msg
        LD    BC,WRTXTSZ-WRTXT  Lenght of text
        CALL  OUTLI             Write text on screen
        RET
;
ASC2I   LD    A,(HL)            Get 10-number
        SUB   48                Make integer of ASCII
        LD    B,A               Number of tens
        XOR   A                 Clear A:
ADD10   ADD   A,10              Multiply 10 times B:
        DJNZ  ADD10
        LD    C,A               Save result in C:
        INC   HL                Increment bufferpointer
        LD    A,(HL)            Get 1-number
        SUB   48                Make integer of ASCII
        ADD   A,C               A:=Hour/minute
        INC   HL                Increment pointer
        RET
;
WRTXT   DEFB  13,10
        DEFM  "Time error!"
        DEFB  13,10
WRTXTSZ EQU   *
;
WHEN    DEFB  13,10,10
        DEFM  "LEAVE   ver 1.00    1984-05-07    (C) TCP"
        DEFB  13,10,10
        DEFM  "When do you want to leave (HHMM) ? "
WHENSZ  EQU   *
;
OK      DEFB  13,10,10
        DEFM  "Ready!"
OKSZ    EQU   *
;
ALARM   DEFS  1
OLDMIN  DEFS  1
OLDINT  DEFS  2
BUFFER  DEFS  10
BUF1    DEFS  2
;
NEWINT  PUSH  HL                Save registers
        PUSH  DE
        PUSH  BC
        PUSH  AF
        LD    HL,TIME           Position actually hour
        LD    DE,BUF1           Position leave hour
        LD    A,(ALARM)         Check if alarm already
                                activited
        AND   A
        JR    Z,WAKE            Wake'm
        LD    B,(HL)            Get actually hour
        LD    A,(DE)            Get leave hour
        CP    B                 Are they the same?
        JR    NZ,NXTINT         No, jump to old interupt
        INC   HL                Increment pointers
        INC   DE
        LD    B,(HL)            Get actually minute
        LD    A,(DE)            Get leave minute
        CP    B                 Are they the same too?
        JR    NZ,NXTINT
        XOR   A                 Clear A:
        LD    (ALARM),A         Set alarm flag
        JR    WAKEIT
WAKE    INC   HL
        INC   DE
        LD    B,(HL)            Get actuall minute
WAKEIT  LD    A,(OLDMIN)        Get last minute
        CP    B                 Are they same?
        JR    Z,NXTINT          Don't write anything
        LD    A,B               Shift registers
        LD    (OLDMIN),A        Save minute
        LD    HL,MESS           Text for message
        LD    BC,MESSSZ-MESS    Size of text
        CALL  OUTLI             Write text on screen
;
NXTINT  POP   AF                Get registers
        POP   BC
        POP   DE
        LD    HL,(OLDINT)       Here is address to
                                ordinary int.
        EX    (SP),HL           Switch HL: and returnaddress
        RET                     Jump to ordinary interupt
;
MESS    DEFB  13,10
        DEFM  "It's time to leave now!!!"
        DEFB  13,10,7
MESSSZ  EQU   *
;
LEAVEIT LD    HL,(OLDINT)       Get old interupt
        LD    (INTVECT),HL      And put it back
        LD    HL,BYETXT
        LD    BC,BYETXTSZ-BYETXT
        JP    OUTLI             Write text and return
;
BYETXT  DEFB  13,10
        DEFM  "No LEAVE activated!"
        DEFB  13,10
BYETXTSZ EQU  *
;
        END   START
```

Vilken rubrik skall jag välja.

1. Vad rymmer raden 10 i TV-MAIN?
2. Hur arbeta bakom kompilatorn.
3. Inledning till hur använder man två program i parallell.
4. Att puffa för Disassembler.
5. Introduktion till continue.radnr.

Efter att ha studerat TV-MAIN en hel del, har även jag modifierat den i avsikt att spara programutrymme, således mer för texten. Jag uppmärksammade förslaget att eroende av checksumma använda någon .v två något olika rader 10.

Det är möjligt att förändra TVMAIN:s rad 10 så att den kan ge samma verkan oavsett vilken checksumma ABC-80:n har, som programmet för tillfället användes på. Det viktiga är, att man känner dess uppbyggnad och funktion.

Här är ett förslag.

```
 a  b  c  d  e  f   g   h  i  j
36 10 0 135 31 133 203 64 22 58 42 254

167 32 3 195 85 13 58 99 13 254 14 202

                          x  y z å  ä  ö
97 13 195 95 13 0 0 203 64 0 187 13 .1
```

a = längdbyte för raden (inklusive a självt och t.o.m. ä)
b = ental för radnummer.
c = 256-tal för radnummer.
d = satskod för skip.
e = antalet byte att överhoppa i RUN-mod. (f.o.m. f - t.o.m. ä)
f = Internkod, listas ; ASCII 59
g = kod för uttryck(sträng).
h = enl. Arne Stockman strängdelimiter (avgränsare).
i = antalet byte i teckenraden (exklusive bytet i)
' = teckenraden (maskinkoden) här innalles 22 byte.
x = som g (uttryck)
y = som h strängavgränsare.
z = som i (här = 0 endast utryckets förekomst har betydelse)
å = satskodavslutning
ä = radslut.

4 st längbyte ingår i raden. z är längden av andra uttrycket vilket lämpligen alltid är 0. De 3 övriga är beroende av antalet byte i maskinkoden. i är lika med byteantalet i koden, e (byte att skippa) är 9 talenheter större än i, a (radlängdbytet) är ytterligare 5 enheter större än e.

Basic:ens listrutin utförs, för varje rad, i 2 steg. Först tas radnummer, satskoder mm. ut, och omvandlas till ASCII-tecken, vilka mellanlagras på radbufferten = (65088-65207). Därefter görs återhopp, vilket i normalfallet innebär att dessa ASCII-tecken skrivs ut till fil eller skärmen.

När ett uttryck ska föras till radbufferten håller registret DE adressen in på raden. DE läggs, (PUSH:as) på stacken. Så sker även i icke normalfallet (vid andra uttrycket). När uttryck-:en har förts till radbufferten, höjs stacken med POP en gång (2 steg) och därefter sker återhoppet.

Om en rad som denna innehåller 2 uttryck kommer återhoppet att ske till den adress dit där DE höll då första uttrycket skulle föras till radbufferten.

Återhoppet sker till den adress, som avgränsaren h= ASCII 64 står på. 40H är instruktionen LD B,B vilken är harmlös, nästa är icke längdbytet i (22 i raden), därför att listrutinen, av naturliga skäl, inte för det till radbufferten, (man vill ju, på skärmen kunna läsa vad som står i strängen, inte hur många tecken den består av). Nästa blir då j 58 42 254 = LD A(254:42), från adr. 65066 som är höga delen av variabelroten, det följande innehållet i raden ger hopp till bestämda platser i Basic-tolken.

Denna rad, (10 i TV-MAIN) måste, som jag ser det, eftersom den inte går att lista, och därför inte går att editera, vara konstruerad så, att kompilatorn inte har fått tillfälle att granska den.

Var radens andlige fader är att söka vet jag ej, kanske hos Microsoft eller grabbarna 3 x Anders på LiTH i Linköping.

Ett stor tack vill jag passa på att framföra till Gunnar Tidner för att ABC-klubben har fått tillgång till detta utmärkta hjälpmedel, som TV-editorn utgör.

Hur tillverkar man så en rad som denna?

Jag hade maskinkoden = j i raden ovan, jag har ingen floppy och kan således ej råka ut för att skada diskar, min ABC-80 har endast 16K arbetsminne (grundutförande).

Return, som vanligen skrivs <cr> förkortar jag till cr. Med 2 tomma 'Ljudbrev' går jag till verket.

Inom parentes sagt har bandkvaliteten betydelse endast i avseende drop outs, och sådana åstadkommer man mestadels själv genom att göra veck på tejpen därför att man kör mot band/slut/början med snabbspolning, eller att man utför bandlägesjustering med avspelningstangenten nedtryckt. Denna fara, att åstadkomma drop outs, (bandveck = err 37,35) ökas till viss del med tilltagande speltid på kassetten.

Jag gör reset och skriver in följande

```
10 REM 1234567890123456789012345678 90cr
```

tar ED10cr kollar och tar åter cr.

Tar sedan ;peek(49152)cr och kontrollerar om svaret=36, som är basic-radens längd. Matar in SAVE REMRAD, antecknar REM-RAD.36 på kassetten och lägger in den för inspelning och tar cr. Återspolar och tar MERGECAS:cr därmed har jag en Basic REM-rad 36 byte lång. Jag ger följande kommandon POKE 65053,193cr, och NEWcr, nu pekar BOFA till 49408.

I delomgångar åstadkommer jag följande rad

```
20 FOR I = 49150 TO 49190 STEP 10 : ; I
   : FOR J = 0 TO 9 : ;PEEK(I+J); : NEXT
   J : ; : NEXT I
```

och ger följande kommando ; PEEK-(49408)cr och avläser radlängden till 114 byte, antecknar LISTRAD.114 på kassett nr.2, och sparar under samma namn, spolar tillbaka och tar MERGECAS:cr och kollar ånyo radlängden å 49408. Därmed är det klart så långt.

Adr. 49408 håller således värdet 114, men det gör också EOFA låg = Adr. 65054 därför att jag endast har just denna rad inlagd, EOFA är således 193:114.

Jag tar RUNcr och kan nu studera vilka byte som kompilatorn har lagt i REM-rad 10, ser då att de 2 första byten är 255, alltså inte ram där. Därefter står 36= längd basicrad, sedan radnummer 10*1+256 *0 = 10 och satskoden 134, som behöver tillägget 132 för REM.

I Radens 6:te byte adr. 49157 börjar texten med 49 = ascii för 1 och slutar i adr. 49186 med 48 för 0.

Nu vidtar arbetet med att härifrån BOFA 49408, i omgångar, förändra REM-raden på 49152. Med BOFA på 49152 'SAVE:as' raden , och med MERGECAS:cr kollar jag att raden är läsbar, därefter hoppar jag tillbaka hit, och med endast RUNcr avläser hur raden nu ser ut.

Jag gör detta i omgångar för att belysa hur man kan ha ett körklart program i ABC80:n på ett ställe under det att man förändrar och spar undan ett annat program på annan plats i ABC-80:n.

Inledningen på LISTRUNraden ska enligt beskrivningen vara 36 135 31 133 203 64 22. Jag räknar fram att teckenantalet 22 ska stå på adr. 49160, och att maskinkoden börjar på 49161, jag ger kommandot POKE 49161,58,42 o.s.v. t.o.m. första NOPinstruktionen, utan att hoppa över något byte eller något kommatecken och slutar inte med kommatecken när cr tas.

Med RUNcr kollas att det blev riktigt så långt, och pokar så in radslutet, bevakar att sista 13 förblir på exakt samma plats som förut.

Så är det tid att lägga, den halvfärdiga raden på kassetten. Jag ger kommandot POKE 65053,192cr tar inte NEW, då skulle 36 på 49152 ändras till 1 och 114,193 i EOFA på 65054,55 ersättas med 0,192. Avsikten är att endast använda kommandona SAVE och MERGECAS, vilka ej fordrar att EOFA är rätt ställd, den ökas inte, då denna rad är kortare än LISTRAD.

Gör man något misstag finns ju LISTRAD att tillgå. Kassetten med REMRAD spolar jag till slut, vänder den om och ställer bandspelaren på inspelning, ger kommandot SAVE UTRAD10.1cr. Jag backar bandet och ställer på avspelning samt tar MERGECAS:cr (inte LOAD, påverkar EOFA). Om allt gick enligt beräkning antecknar jag på kassetten UTRAD10.1, tar POKE 65053,193cr (ej heller nu NEW), tar sedan RUNcr, och radens byte skrivs ut. Nu är det tid att prova maskinspråkrutinen, jag tar alltså ;CALL(49161)cr, den står ju där eller hur?

LISTRAD står på BOFA=49408 och det är ju den som ska komma i RUN-mod och därigenom återigen visa UTRAD10.1.

Går det till pepparodlingarna så återstår endast att ta reset, med loadcas:cr inhämta UTRAD10.1 med POKE65053,193cr och NEW cr flytta BOFA, samt att med RUNCAS:cr ta in och köra LISTRAD.

Oberoende om det slutade väl eller ej när man arbetar med maskinkod bör man ha tillgång till någon form av disassembeler, själv använder jag CODLIZ80 kostar 60kr. + porto, kassett och inspelning =15:- Sa. 75:- till postgiro 447 58 42-3 S.Johansson, ange avs.-adr.

Byter man ut LISTRAD genom att skriva RUNCAS:cr och därvid har CODLIZ80 i bandspelaren, samt i formuläret för in startvärdet 49167, slutvärdet 49190,tiden ca 20 sek. och startorder, kan man i lugn och ro studera den disassemblerade maskinkoden.

Tillbaka till raden igen, den saknar ju ännu byten för skip mm. Radens inledande byte 36 10 0 står där ju redan. BOFA har jag på 49408 och ånyo är LISTRAD körklar för att visa nuvarande UTRAD 10.1-s ingående byte. Med POKE-kommandon passar jag in byten 135 31 133 203 64 22. Jag flyttar BOFA att peka på 49152 men utan att göra NEW och spar så med SAVE UTRAD10cr (utan .1). Därefter kontrollerar jag kassetten med MERGECAS:cr. Nu tar jag NEWcr och LOADCAS:cr, därmed har jag min UTRAD10 klar.

Genom att ta MERGECAS:cr och därvid ha LISTRAD i bandspelaren får jag ett

ABC STOCKHOLM

Så har åter varit årsmöte i ABC-Stockholm och så här ser den nya styrelsen ut:

Ordförande: Bo Hjulström < 557>
Sekreterare: Kalle Lindström < 837>
Kassör: Kjell Järbin < 700>
Materielförvaltare: Gunnar Forsell <1631>
Mötesansvarig: Joe Johnsson <1489>
Övr. ledamot: Kurt Minnberg <1522>
Övr. ledamot: Christer Lerin < 797>
Suppleant: Lars-Göran Göransson < 495>
Suppleant: Stig Löfgren < 872>

Naturligtvis kommer vi att fortsätta med öppet hus på Tisdagskvällarna som vanligt, Tid 18.30 - ca. 22.00, Plats klubblokalen Vidängsvägen 1 Alvik. Här finns möjlighet att träffa andra medlemmar, diskutera problem, köpa ABC-Klubbens publikationer, kassettband och i mån av tillgång disketter

---

Inför hösten har vi planerat 3 st informationsmöten (i september, oktober och november) men vi måste ha tips om vad vi skall ta upp på dessa möten. Har du några ideer eller känner du nån annan med sådana, så ta kontakt med någon av oss i styrelsen antingen per telefon eller brev eller i KOM till mötet ABC-STOCKHOLM FORUM.

Om du inte redan är medlem i ABC-Stockholm så blir du det enklast genom att sätta in 40 kronor på postgiro 15 33 36 - 3 och glöm inte att ange namn, medlemsnummer och att avgiften gäller ABC- Stockholm

Med Vänlig Hälsning
Styrelsen i ABC-Stockholm

---

program bestånde av 2 rader, rad-numren10 och 20.

Jag kollar med ;PEEK(65053)cr att talet där är summan av 36 + 114 och tar då RUNcr, vilket ger samma som tidigare. I rad 10 skip-as 31 byte d.v.s. till rad 10:s slut, men rad 20 utförs. Nu tar jag LISTcr, som resulterar i att programmet stöter på masinkoden i rad 10 och överförs till RUN-mod, och skeendet blir detsamma som i föregående mening.

Jag editerar rad 20 (listraden) så att den pekar in i radbufferten 65085-65125 för att konstatera hur där ser ut.

Med en kommenterad disassemblerlista är sedan steget inte långt för att kunna tillverka en bandstump, som i radluckorna läser in maskinkod, så att man genom att, i kommando-mod, ta ;CALL(31864,- RADNR.) cr och får en verklig CONTINUE.

UTRAD10:s kod är nu disassemblerad och granskad, dess funktion vid LIST-och RUN-mod är nu kontrollerade, så det bör ej vara något hindrer att sammanslå den med TV-MAIN. Med programmet TV-MAIN inlagt i minnet, och UTRAD10 i bandspelaren kan rad 10 bytas ut genom att ta MERGE-CAS:cr.

Därmed skulle denna beskrivning vara genomgången, för detaljerat och virrigt om en struntsak, anser de flesta. För svårt tycker någon. Har jag inte alltför fel i sak, kan det kanske för några leda till nya idéer.

Sigvard Johansson <3466>
Lindeborgsgatan 47
214 81 Malmö

ATN og ASIN.

Når man anvender ATN(x)-funksjonen må man huske på at den bare gjelder mellan 0 og 90 grader. Forsök:

TAN(45*PI/180)=1 og TAN(225*PI/180)=1, d v s at ATN(1)*180/PI=45 eller 225 grader!

I bl a amatörradio blir ATN anvendt ved beregning av satellitbaner etc., og det kompliserer at man har den nevnte begrensningen.

M Min lösning på problemet er åomgjöre argumentet x til x/y, hvor x og y er fordelt slikt:

Nord-Öst=0-90 :+x, +y
Öst-Syd=90-180 :+x, -y
Syd-Vest=180-270 :-x, -y
Vest-Nord=270-360 :-x, +y

Medet lite program som vises her, beregnes vilken som helst vinkel. For sikkerhet skyld er lagt inn noen IF-rader (192,193,242) for det fall at x og/eller y blir null.

Man får ofte også bruk for ASIN (og ACOS), men de finnes ikke bland ABC-80s funksjoner. Normalt beregnes ASIN=ATN(x/SQR(-x*x+1)) men det blir komplisert å erstatte x med x/y her. Problemet er löst med en enkel iterasjonsrutine, som er vist her. Det er odså lagt inn 'sikring' for 0-verdier av x og y.

Iterasjonen består i at man först gjetter på at vinkeln (v) er null (200). Så regner man med den (220) ogser hvor stor feil man får (differens=D), korrigerer (250) og regner om igjen med korrigert verdi (220-260). Når differrensen er mindre enn 0.00001 er man nöyd og 'hopper ut'. Vinkeln (i radianer) blir omgjort til grader (270), men dette er ikke nödvendig dersom raderne brukes som subrutine.

Tilsvarende metode anvendes også for ACOS o s v, men se til at SGN-funksjonen blir rett anvendt.

Halvard Torgersen <3342>
Ref.:
ABC-bl. 1980-no2-s15
ABC-bl. 1983-no3-s9

```
100 REM .      NORD A+Y 0 grader
110 REM .           I
120 REM .    -X     I    +X
130 REM . VEST<----+----->ÖST
140 REM .  270      I      90
150 REM .           I-Y
160 REM .       SYD V 180
170 REM *** ATN *** H.T. *******
180 ; "x="; : INPUT X : S=SGN(X)
190 ; "y="; : INPUT Y : T=SGN(Y)
192 IF X=0 AND Y=0 THEN ; "UDEFIN."
    : GOTO 180
193 IF Y=0 THEN 242
200 V=ATN(X/Y)*180/PI
210 ON T+2 GOTO 220,230,230
220 V=180+V : GOTO 250
230 ON S+2 GOTO 240,250,250
240 V=360+V : GOTO 250
242 IF X>0 THEN V=0 ELSE V=270
250 ; "Vinkel="V : GOTO 180
```

```
100 REM *** ASIN *** h.t. ***
110 REM SIN(v)=x/r, v=ASIN(x/r)
120 REM R=SQR(x*x+y*y)
130 REM
140 ; "x="; : INPUT X : S=SGN(X)
150 ; "y="; : INPUT Y : T=SGN(Y)
160 IF X=0 AND Y=0 THEN ; "UDEFIN."
    : GOTO 140
170 IF X=0 OR Y=0 THEN 320
180 R=SQR(X*X+Y*Y)
190 Z=X/R
200 V=0
210 P=SGN(Z-V)
220 W=SIN(V)
230 D=W-Z : Q=SGN(D)
240 IF ABS(D)<.00001 THEN 270
250 V=V+(P*Q*D)
260 GOTO 220
270 V=V*180/PI
280 ON T+2 GOTO 290,300,300
290 V=180-V : GOTO 310
300 IF S=-1 THEN V=360+V
310 ; "Vinkeln="V : GOTO 140
320 REM --- x eller y = 0
330 IF S=0 THEN 360
340 IF S=1 THEN V=90 ELSE V=270
350 GOTO 310
360 IF T=1 THEN V=0 ELSE V=180
370 GOTO 310
```

Radannonser.

KÖPES

Flexskivemine FD2 eller liknande passande till ABC80.
Erik Mårtensson <936>
Banjovägen 6
961 46 BODEN

Tel: 0921-656 50

SÄLJES

8K extra minne för ABC80. Passar Databoard diskstation. Märke K8848/E4680/A55
Conny Ohlsson <1010>
Sundsvall

Tel: 060-202 00

PROGRAM BYTES

Jag vill byta program med jämnårig.
Stefan Varga <2482>
Hallongatan 4
267 00 BJUV

Födelseår: 1968

SÄLJES

ABC-80 32K RAM. Kassettminne och bildskärm. Originalförpackning. Böcker ABC om Basic och

Avancerad programmering på ABC-80.

Massor av program bl a ALLA ABC-kasetter totalt över 20 kasetter.

Hela klabbet inkl frakt 6500.-
Ring Janne Pettersson <1090>
Tel: 0278-512 18

NYA MONITORN.

Tyvärr har det tagit längre tid än vad jag räknade med att utveckla den nya monitorn. När detta läses hoppas jag verkligen att den är igång.

Själva monitorn har skrivits av Lars-Göran Göransson och Benny Löfgren. Meddelandesystemet, MSG, har jag skrivit. När detta skrives så är som sagt inte allt klart ännu. Kommando och annat är enbart preliminära, jag redovisar läget som det är nu när jag skriver detta (840813). Bli alltså inte förvånade om det när systemet kommer igång finns saker som inte stämmer riktigt med denna artikel!

Hårdvaran:

Datorerna består av två stycken ABC802-or, som är kopplade till telefonnummer 08-80 15 23 och 08-80 15 26. Till dessa skall man kunna ringa med både 300/300 bps V21-modem och 75/1200 V23-modem, split speed. Vi har inte löst funktionen med split speed när detta skrivs, men vi räknar med att göra det.

Jag har preliminärt beställt ett 1200/1200 V22-modem som kommer att kopplas till en ny telefonlinje. Den dator som vi troligen kommer att använda är en ABC802:a som normalt finns i klubblokalen och som inte är avsedd för monitorn utan för lokal verksamhet i klubblokalen. Det är inte så ofta möte i klubblokalen så denna linje kan vara öppen alla dagar utom tisdagskvällar och övriga tider när det finns folk i klubblokalen som vill använda 802:an.

Tyvärr har televerket en lång väntelista på V22-modem. De har beställningar från mars månad som inte är levererade. Vi kan tidigast räkna med leverans i oktober. Detta innebär att det kan dröja en tid innan denna linje kommer igång. Telefonnumret kommer när det blir aktuellt att annonseras på telefonsvararen, 08-80 15 22.

ABC802:orna är anslutna till en MICA 40 Mbytes winchesterdisk via ett CAT-NET. CAT-NET:et har vi fått låna av CAT Ingenjörsbyrå AB. Winchestern, som även innehåller en backupenhet, har vi köpt.

Mjukvaran:

Inloggning: Normalt ekas alla tecken från klubbens monitor, men man kan genom ett kommando stänga av ekningen. Ställ därför in terminalprogramet för full duplex (egentligen ekning av tecken från värddatorn).

Användaren möts av en textfil, som dock INTE föregås av något "Tryck <RETURN> utan som läggs ut direkt. Sedan frågar systemet efter medlemsnummer och lösenord. Som lösenord skall man ange sitt personliga lösenord som stod längst upp till vänster på avien för medlemskortet. Det inloggningsnummer som står på själva medlemskortet skall INTE användas.

Lyckas inloggningen får man en välkomsthälsning. Därefter tar det lite tid för systemet att uppdatera en del ISAM-filer. Efter detta svarar systemet med en ">" och är då beredd att ta emot komando.

Kommandointerpretatorn:

ABC-Klubbens nya monitor använder samma typ av kommando som ABC-maskinerna har om man kör under CMDINT, dvs efter att ha skrivit BYE i BASIC:en.

Normalt anger man kommando enligt följande:

Kommando, switch, parameter.

Vill man få reda på vilka kommando som finns när man kör systemet så gör man HELP * och då får man se alla kommandon. Man kan också göra HELP S* och då får man se alla kommando som börjar på S. Ett "?" kan också användas och ersätter då en bokstav. Till varje kommando finns det en hjälpfil som kan läsas med man anger "HELP <kommando>".

Systemet använder en inbuffert på 400 tecken så man kan använda "type ahead", dvs skriva flera kommando i en följd med return emellan. Dessutom kan man skriva flera kommando till monitorsystemet genom att skriva ";" mellan varje kommando. Detta gäller dock inte MSG systemet.

Här följer en kort sammanfattning av alla kommandon. För utförligare anvisningar hänvisar jag till hjälpfilerna i systemet.

LIB Med kommandot får man se vilka filer som det finns på det AKTUELLA biblioteket i programbanken. Observera alltså att om man gör lib när man har loggat in så får man inte se några filer eftersom det inte finns några på huvudnivån.

Till LIB-kommandot finns switchar som gör att man kan få se datum och bestämma i vilken ordning filerna visas.

Ett FILELIB-kommando finns för närvarande inte. En lista över alla filerna publiceras dock här i ABC-Bladet på annan plats.

Att ta fram en lista över alla filer på programbanken bör ej ske av varje medlem eftersom detta kommer att belasta den gemensamma winchesterdisken för mycket.

ECHO Kommandot används för att stänga av och sätta på ekning av tecken från monitorn. Ekningen stängs av med "ECHO OFF" och sättes på med "ECHO ON".

GET och SEND Dessa kommandon används för att skicka och hämta filer från monitorn. När detta skrivs så utvecklas en förbättrad version av FILTRANS som innebär att man skall kunna dels använda de gamla programmen lokalt och dels om man använder nya program lokalt skall man kunna använda ett bättre filöverföringsprotokoll med checksummekontroll.

TYPE och WRITE Dessa kommando används för att skriva in textfiler och läsa sådana.

NEWS Nyhetsfilen läses med detta kommando.

DIR Kommandot används för att sätta ett aktuellt bibliotek eller se vilket bibliotek som är aktuellt. Enbart "DIR" talar om vilket bibliotek som är aktuellt och vilka ev underbibliotek som finns. Skall man se ABC80 biblioteket för spelprogram skriver man "DIR ABC80/SPEL". Man kan också först skriva "DIR ABC80" och sedan "DIR SPEL".

LOGIN Med detta kommando kan man avsluta körningen och en annan medlem kan påbörja den utan att telefonförbindelsen bryts. Man anger bara "LOGIN" utan parameter.

WIDTH Kommandot används för att sätta antal tecken per rad. Uppgiften sparas vid utloggningen. Normalt är den satt till 80 tecken. Man anger "WIDTH 40" om man vill ha 40 tecken per rad.

LOG Kommandot används för att se loggen över tidigare körningar. Kommandot kommer dock att byta namn eftersom det kolliderar med LOGin.

KILL KILL används för att ta bort eller markera filer för borttagning. Om de tas bort direkt beror på om det är man själv som har skickat in filen eller vilka privilegier som man har.

BYE Kommandot används för att avsluta körningen. Man bör komma ihåg att alltid avsluta körningen med BYE. Detta skall man göra även om man har använt "sluta" kommandot i MSG-systemet.

MSG Kommandot används för att köra meddelandesystemet, som beskrivs på nästa sida.

MSG-systemet.

MSG-systemet är ett meddelandesystem med flera möten och möjligheter att skriva brev till medlemmar. Systemet håller reda på vilka inlägg som man har läst, i de möten som man är medlem i.

När man loggar på systemet kollas brevlådan. Finns det ett brev så MÅSTE man läsa det och sedan frågar systemet om man vill spara brevet. Om man svarar ja på den frågan sparas brevet på ett speciellt bibliotek för sparade brev. Det är meningen att man skall kunna hämta hem sparade brev med hjälp av GET i monitorn.

Kommandon kan alltid förkortas. Man skall INTE ange den del av kommandot som står inom parantes. Inläggen i varje möte numreras i en serie var för sig. Vill man utföra det kommando som datorn föreslår så trycker man return. Normalt kan man läsa alla nya inlägg genom att bara trycka return.

(Bli) medlem (i) — används för att bli medlem i ett möte. Man kan normalt inte själv bli medlem i ett slutet möte.

_ista alla) möten — visar en lista över alla möten, med uppgift om antalet inlägg i de olika mötena.

(Gå till) nästa (möte) — används för att gå till nästa möte som man är medlem i.

I vissa lägen så kan man få upp "(Gå till) nästa (möte)" och när man sedan gör det genom att trycka return så får man reda på att man har ett antal olästa texter. Sedan kan man direkt få upp "(Gå till) nästa möte" igen. Detta är ingen bug, utan beror på att systemet har upptäckt att de inlägg som var olästa antingen var raderade eller var skrivna av dej själv. Man kan givetvis läsa texter som man själv har skrivit, men då får man själv skriva "läsa <nr>".

Gå (till möte) — används för att gå till ett möte.

(Lista alla) nyheter — används för att lista antalet nya/olästa inlägg i respektive möte.

Status (för mig själv) — ger status på sig själv. Man får se vilka möten man är med i och hur många olästa texter man har i dessa.

Var (är jag) — talar om i vilket möte som man är närvarande i.

Vilka (är närvarande) — talar om vilka som just nu kör systemet. Om någon loggar ut ur MSG-systemet utan att använd "sluta" så uppdateras denna uppgiften inte förrän nästa medlem loggar in på samma nod.

Läsa (inlägg) — används får att läsa inlägg. Det kan man göra genom att bara trycka return när datorn skriver ut t ex "Läsa (inlägg) 1 - ". På detta sätt så kan man läsa alla olästa inlägg. Vill man läsa andra inlägg än vad systemet föreslår så får man själv ange "Läsa" plus ett nummer. Om man själv anger vilket inlägg man vill läsa så uppdateras INTE uppgiften om att man har läst texten. "Återse (inlägg)" och "Läsa (inlägg)" är synonymer.

Om man inte har WIDTH 80 satt utan något annat så formateras själva textmassan om så att alla text mellan tomrader fylls ut på alla rader och brytning sker på ordgränser.

(Läsa) igen — används för att igen läsa den senast lästa eller skrivna texten.

Endast (läsa senaste) — används för att endast läsa ett visst antal av de sista OLÄSTA inläggen. Man anger hur många man vill läsa.

Hoppa (över inlägget) — används för att hoppa över det inlägg som datorn vill att man skall läsa.

Skumma (inlägg) — används för att bara läsa ett visst antal rader i varje inlägg som man läser.

(Läsa) resten — används för att läsa resten när man skummar inlägg.

(Läsa) hela — används för att läsa hela inlägget när man skummar inlägg.

(Skriva eget) inlägg — används för att skriva inlägg i ett möte. Nedan finns en förteckning av de kommandon som man har tillgång till vid inläggsskrivning.

Skriva (inlägg) — är samma sak som "(Skriva eget) inlägg".

Kommentera (inlägget) — används för att kommentera ett inlägg. Vill man kommentera det senast lästa eller skrivna inlägget skriver man bara "kommentera". Annars måste man skriva vilket inläggsnummer som man vill kommentera. Kommandot kan lämpligen förkortas till "kom".

(Skicka) brev (till) — används för att skriva ett brev till en medlem. Man kan ange medlemsnummer eller namn. Om namnet inte är unikt så får man reda på vilka som finns med samma namn och vilka medlemsnummer som de har. I så fall måste man ange medlemsnumret för att skriva brevet.

Återse (inlägg) — används för att återse inlägg. Kommandot påverkar inte systemets uppgift om lästa inlägg. Man skall ange vilket kommando som man vill återse.

(Visa) menyn — ger uppgift om vilka kommando som finns i systemet. Om man har WIDTH 80 satt så visas kommandona i full 80-teckens bredd i två spalter annars får man se den i en spalt.

(Ta) bort (den) — tar bort inlägget. Man kan normalt bara ta bort egna inlägg.

Flytta (inlägget) — flyttar inlägget till ett annat möte som man är medlem i. Man kan normalt bara flytta egna inlägg och bara EN gång.

Prioritera (möte) — innebär att mötet som man anger hamnar först i sin personliga lista. Detta påverkar ordningen som man hamnar i möten med olästa texter, om man kör systemet genom att bara trycka return när datorn anger "(gå till) nästa (möte)".

(Terminal) sidstorlek — sparar uppgiften om hur många rader man skall se av en text innan systemet frågar efter "tryck return". Anger man 0 så kommer aldrig frågan. Normalt är det 20 rader. Observera att antalet rader är lika med det antal som finns i ursprungstexten. Har man inte WIDTH 80 så formateras texten om så att man får FLER rader på skärmen, men systemt räknar bara de ursprungliga raderna.

Utträda (ur möte) — används för att utträda ur ett möte

Hjälp — används för att läsa hjälpfilen för ett kommando. Man anger kommandot som parameter.

Help	är samma sak som "hjälp".
?	ger menyn.
Sluta	används för att avsluta körningen. Man MÅSTE alltid avsluta med "sluta" för annars uppdateras inte systemets uppgifter över lästa texter. Efter "sluta" kommer man tillbaka till monitorn. Glöm inte att logga ut från monitorn med BYE.

Editorkommandon för MSG-systemet:
==========

Kommandotext inom parantes skall ej anges!

Kommando:	Funktion:
Hjälp	Denna text visas.
Help	Denna text visas.
?	Editorkommandon visas.
Läsa (den skrivna texten)	Den skrivna texten visas.
(Läsa) hela (den skrivna texten)	Den skrivna texten visas.
Läsa (den) kommenterade (texten)	Den text som man kommenterar läses
Ändra (rad)	En rad ändras. Anger man inte radnummer ändras den föregående raden. Man editerar fram texten med -->.
Ändra ärende	Rad nr 1, dvs ärenderaden ändras Detta kan också ske med "ändra 1".
Radera (rad)	En rad raderas. Anges inte radnr tas föregående rad bort.
Spara (texten)	Inlägget avslutas och sparas.
Lägga (inlägget)	Inlägget avslutas och sparas. Man kan också avsluta ett inlägg genom att först på en rad ange CTRL-Z.
(Ta) bort (texten)	Avbryt inlägget utan att spara det.
Avbryt (inlägget)	Avbryt inlägget utan att spara det. Inläggskrivningen kan även avbrytas med CTRL-C/CTRL-D först på en rad.
Ny (rad)	Sätt in en ny rad.

Man kan alltid ta bort det senast skrivna ordet med CTRL-W.

Bo Kullmar

ABC-Klubbens programbank

1984-08-10 21.39.47

Denna lista omfattar de program som fanns på klubbens programbank vid nämnda tidpunkt. Observera att listan enbart omfattar program som har granskats av programgruppen.

Filstorlekarna är storleken i CAT-NET format. CAT-NET använder ett speciellt diskformat som inte är lika med det vanliga DOS:en för ABC-maskiner. Den skillnad som finns jämfört med ABC-DOS (UFD eller gammalt DOS) är främst att under CAT-NET:s tar inte filer upp större utrymme än de verkligen behöver. Det är alltså inte som på gamla monitorn att varje fil upptar ett utrymme som är jämt delbart med 4.

De rubriker som finns i listan utgör biblioteksnamn i nya Monitorsystemet. ABC80 och ABC800 är överbibliotek till de övriga. Alla 800 program har normalt ".BAS" som extension eftersom biblioteket anger typ av dator.

Dessutom finns alla program som har givits ut på kassett på kassettbibliotek. Program som har givits ut på kassett nr 12 finns t ex på biblioteket "kassett/k12".

Bo Kullmar

ABC80/SPEL
==========

ALIEN .BAS	14
AMAZING .BAS	27
ASTROZIP.BAS	16
BIL16K .BAS	32
BIL32K .BAS	48
BILRACE .BAS	14
BLACKJAK.BAS	47
BLOCKAD .BAC	20
BOMB .BAS	24
BRIDGE .BAS	23
BYRÅKRAT.BAS	15
CHEMIST .BAS	9
CHOPPER .BAS	45
DOMINO .BAS	15
FARTYG .BAS	14
FEMKAMP .BAS	50
FLUGAN .BAS	30
FMG .BAS	41
GROTTAN .BAS	44
HIQ .BAS	24
INSTÄNGD.BAS	16
INVADER .BAC	40
JAKT .BAC	24
JAKTPLAN.BAS	22
JULGRAN .BAS	15
KATTRÄTT.BAS	14
KORT .BAS	19
LABJAKT .BAS	15
MASKLIPP.BAS	21
MEMORERA.BAS	9
MINDLOAD.BAS	12
MINDPLAY.BAS	35
MONSTER .BAS	33
NORRTRAV.BAS	28
NORSK .BAS	27
PADDEL .BAS	20
PLAYTILL.BAS	9
QUEST .BAS	43
QWERTY .BAS	25
REAKTION.BAS	10
RIVAMUR .BAS	17
ROADRACE.BAS	19
ROBOT .BAS	31
ROULETTE.BAS	46
RUIN .BAS	15
RYMDBUS .2	60
RYMDBUS .BAC	9
RYMDKRIG.BAC	39
SOLVALLA.BAS	27
SQUASH .BAS	19
STARTREK.BAS	46
STARTRK2.BAS	75
STARTRK2.REM	5
STUDS .BAS	45
SURROUND.BAS	28
SÅSER .2	41
SÅSER .BAC	16
SÅSER16K.BAC	54
TUBEN .BAS	13
TURNABLK.BAS	27
TVSPEL .BAS	7
TYPETEST.BAS	12
UBÅT1281.BAS	25
UPPSJÖ .BAS	44
UPPSJÖ .INF	9
UPPSJÖ1 .PRO	3
UPPSJÖ2 .PRO	3
UPPSJÖ3 .PRO	3
UPPSJÖ4 .PRO	3
UPPSJÖ5 .PRO	3
UPPSJÖ6 .PRO	3
WARFISH .BAS	19
WORLDPOW.BAS	27

Antal filer: 73

ABC80/GRAFIK
============

ANSIKTE .BAS	9
BASPLOT .BAS	13
BESÖK .BAS	13
DONALD .BAS	16
FLAGGA .BAC	18
GRAFIK .BAS	5
HÄLSNING.BAS	4
IGELKOTT.BAC	18
KENNEDY .BAS	22
KLOCKA .BAS	13
KULGRAF .BAS	7
MADONNA .BAS	31
MÖNSTER .BAC	20
MÖNSTER1.BAS	7
RITA .BAS	14
SNOBBEN .BAS	13
SNOOPY .BAS	19
SYMETRI .BAS	6
SYNKRON .BAC	8
TAKDROPP.BAS	13
TRIANGEL.BAS	8

Antal filer: 22

ABC80/MUSIK
===========

PIANO .BAC	14
SYNTIZER.BAS	17
TICOTICO.BAC	12

Antal filer: 4

ABC80/UTILITY
=============

CONCAT .BAS	6
CSUMTEST.BAS	5
DELETE2 .BAS	12
DELREN .BAS	12
DEVELOP .BAS	8
DISKREG .BAS	27
EXTDEL .BAS	10
FILSUM .BAS	5
GRAFPRIN.BAS	6
GRAFRUB .BAS	6
HJÄLP16K.BAS	33
HJÄLP32K.BAS	29
HJÄLPARE.REM	9
HUVUD .BAS	11
LAYOUT .BAS	30
LIB .BAS	14
LIBLIST .BAS	11
LISTBART.BAS	6
LISTVARI.BAC	22
LOOK .BAS	14
LVARIRAD.BAS	26
LÄSDISK2.BAS	44
LÄSDISK2.REM	12
MEMCHECK.BAS	9
MEMSTAT .BAS	10
MX80PR .BAS	8
NOCTRLC .BAS	6
NOTANGNT.BAS	5
NOTONLY .BAS	6
NYCHANGE.BAS	12
NYVISAPR.BAS	13
REMHODE .BAS	14
RESETFIX.BAS	7
RUNONLY .BAS	7
SKÄRM .BAS	11
SÖKAREN .BAS	7
VISA . .BAS	11

Antal filer: 38

ABC80/KOMMUNIK
==============

ABCMINI .BAS	7
ABCTRAN4.BAS	30

EXTEND800.

CAT Ingenjörsbyrå har utvecklat ett mycket användbart hjälpprogram till ABC800, ABC802, ABC806 och DTC1. Systemet är avsett för att användas både tillsammans CAT-NET och utan CAT-NET.

Maskinkodsrutinerna finns i REL-filer. Systemet tillhandahålles såväl på normalt diskformat (det som CAT kallar lokalt format) som på det speciella CAT-NET formatet.

Nya kommadon:

Change	Byter variablers och funktioners namn
Cross	Söker efter variabler, funktioner, text eller radnummer
Dir	Ger biblioteksl istning
pe	Skriver ut textfiler
ar	Listar variabler, funktioner eller radnummer

Extra instruktioner:

Key	Definierar och dimensionerar tangenter
>	Ger FLOAT, NO EXTEND
<	Ger INTEGER, EXTEND

Extra funktioner:

Bin$(<tal>)	Ger argumentet i binär form
Dec(<sträng>)	Ger värdet på hexsträngen
Dump$(<sträng>)	Ger ASCII-värdena för tecken i strängen i decimal form
Dump$(<sträng>)	Ger ASCII-värdena för tecken i strängen i hexadecimal form

Tangetbordsbuffert, tangentdefinitioner och kommandofiler:

Tangetbordsbufferten buffrar cirka 80 cken. Definition av makrotangenter kan ske, dels så att definitionerna laddas in från fil och dels att definitionen läggs upp i minnet tillfälligt till CLEAR eller NEW görs.

Kommandofiler kan framställas, så att datorn hämtar alla kommandon från en fil d extension .HAS.

För att använda kommadofiler skapar man en fil enligt följande:

```
PREPARE "SUB.HAS" AS FILE 1
PUT §1, "PRINT 'Hej'"+CHR$(13)
PUT §1, CHR$(255)
CLOSE 1
```

För att köra den skapade filen skriver man OPEN 'HAS:SUB' AS FILE 1 : CLOSE 1 eller SAVE HAS:SUB. När detta har gjorts läser datorn tecken från filen SUB.HAS precis som om de "knackats in" på tangetbordet.

När filen SUB.HAS är slut kommer datorn åter igen att läsa tecken från tangentbordet som vanligt. Dock kan jag inte få kommandofiler att fungera på min ABC806 med UFD-DOS ver 19.

Jag har under utveckligen av MSG-systemet till ABC-klubben använt mig av EXTEND800 på min ABC806, utan anslutning till CAT-NET.

EXTEND800 tar upp 2 Kbytes av tillgängligt minne i datorn. Detta är den enda nackdel som jag har upplevt med systemet. Jag kan varmt rekommendera systemet, trots att vissa fel finns i som jag redovisar här. Det är väl anpassat till BASIC II. Speciellt så uppskattar jag funktionerna VAR fn och CROSS <FNNamn>.

När man programmerar i BASIC II så använder man flitigt flerradiga funktioner. Om man har en funktion som t ex har namnet FNCat ger kommandot CROSS FNCat en listning av funktionen i sin helhet och alla anrop till funktionen. Också VAR fn, som ger en listning av namnen på funktionerna är mycket användbar.

Skall man se bibliotket på en lokal RAM-floppy så måste man tyvärr skriva DIR RMI:. DIR ger normalt filnamn med gemena bokstäver, vilket naturligtvis upplevs lite underligt. Kanske har man gjort detta för att det skall se annorlunda ut i jämförelse med bibliotekslistning över centrala filer i CAT-NET (RUN CAT:). Man kan även få utlistning med versaler om man anger en option. Storlekar på filer kan man också få, samt styra ut utskriften till en fil.

Type kan användas för att omdirigera utskriften till en fil. Dock ger detta en extra radframmating för varje rad. Utskrift på lokal skrivare utan anslutning till CAT-NET har jag inte fått att fungera.

Kollisioner tycks uppstå med Key om man använder program som skickar text med Pf-tangenter i terminalmode.

Variabelnamn som kolliderar med de nya funktionerna kan ej användas. Detta gäller t ex variabeln Var. Vid vissa lägen sker syntaxkontroll av programmet innan kommandot utföres och vid syntaxfel avbryts kommanot.

EXTEND800 ger några funktioner som jag är van att arbeta med på Burroughs stordator. EXTEND800 kostar cirka 2000 kronor. Systemet verkar vara bättre än SMARTAID800 som Owoco visade på mikrodatormässan. Fördelen med Owocos system är att det inte tar upp plats i det normala minnet. När dett detta skrivs så har Owoco inte släppt ut sin SMARTAID800.

Bo Kullmar <1789>

Rekursiva funktioner

BASIC II.

Ett rekursivt anrop sker när en procedur /funktion anropar sig själv. Då skall nya dataareor läggas upp efter det nya anropet. När proceduren/funktionen återvänder, återvänder den till "sig själv". Detta innebär att man skall "få tillbaka" de gamla dataareorna.

I BASIC II kan man utföra rekursiva anrop i funktioner. Normalt sker det rekursiva anropet genom att koden mångfaldigar sig själv samtidigt som nya dataareor tilldelas. I BASIC II återanvänds samma kod, men nya dataareor tilldelas. Detta innebär att man vid rekursion snabbt kan få minnet fyllt. (Dataareorna lagras på stacken.)

För att man skall kunna använda nya dataareor vid varje rekursivt anrop måste variablerna vara lokala. Inparametrar är alltid lokala i BASIC II funktioner. Rekursion kan vara lite svårt att förstå, det förutsätter ett nytt tankesätt! För att underlätta förståelsen så har jag gjort ett litet exempel nedan. Kör exemplet på en ABC800/802/806 :a!

Första delen av funktionen genomlöps tills det rekursiva anropet kommer. 29 gånger anropar funktioen sig själv och går uppåt i nivå. När I inte längre är mindre än 94 så börjar funktionen att backa ut ur sig själv. Detta blir förhoppningsvis lite enklare att förstå om man kör exemplet på en dator och studerar resultatet. Observera att variabeln Text$ förekommer i 29 olika "versioner" (lokala), när funktionen har genomlöpts.

```
10 ! RECURSION
20 INTEGER : EXTEND
30 ; FNRec$(65)
40 DEF FNRec$(I) LOCAL Text$=10
50   Text$=CHR$(I)
60   I=I+1
70   ; Text$ '  ';
80   IF I<94 THEN ; FNRec$(I)
90   Text$=Text$+'+++'
100   RETURN Text$
110 FNEND
```

För er som inte har tillgång till en dator med BASIC II har jag gjort en utskrift som visar vad som händer när ovanstående exempel körs. Observera dock att av redigeringstekniska skäl så kan exemplet bli lite förvanskat!

```
A  B  C  D  E  F  G  H  I  J  K  L  M
N  O  P  Q  R  S  T  U  V  W  X  Y  Z
Ä  Ö  Å  Å+++   Ö+++   Ä+++   Z+++
Y+++   X+++   W+++   V+++   U+++   T+
++   S+++  R+++   Q+++   P+++   O+++
   N+++   M+++  L+++   K+++   J+++
I+++   H+++   G+++   F+++   E+++   D+
++   C+++   B+++   A+++
```

Vill man studera vad som händer om Text$ inte är global så kan man ändra rad 40 och införa en ny rad nr 25 enligt nedan. I det här fallet skulle man kunna uppnå samma sak med iteration (upprepning).

```
25 DIM Text$=200 ! Global variabel
40 DEF FNRec$(I)
```

Här är början av ett kraftigt redigerat exempel på när Text$ är global.
(Bara avkortat)

```
A  B  C  D  E  F  G  H  I  J  K  L  M
N  O  P  Q  R  S  T  U  V  W  X  Y  Z
Ä  Ö  Å  Å+++   Å++++++   Å++++++++++
  Å++++++++++++   Å++++++++++++++++
Å++++++++++++++++++    Å++++++++++++++
++++++++
```

Strängen "Å++++++++++++++++" ökar alltså med tre "+" varje gång.

För ytterligare exempel hänvisas till BASIC II manualen och läroböcker i programmering. Man bör notera att rekursion normalt inte finns till språk som BASIC, FORTRAN och COBOL utan man bör i så fall titta i läroböcker för moderna språk typ Pascal, LOGO eller ADA. Enligt ett rykte kommer rekursivitet att införas i en ny version av FORTRAN.

Bo Kullmar

SUBRUTINER!

Subrutiner eller flerradiga funktioner.

Under den här rubriken har ABC-bladets redaktion tänkt att läsarna skulle kunna skicka in användbara subrutiner för att dela med sig av sitt kunnande.

Vi hade tänkt att subrutinerna skulle ha en viss form för att dels underlätta publiingen dels för att underlätta användandet. exempel här bredvid.

Subrutinen skall alltid börja på rad 200. Alla radnummer ovanför skall kunna tas bort utan att funktionen av rutinen ändras. Raderna ovanför 200 skall bestå av ett testprogram som använder subrutinen.

På rad 200 som är början på subrutinen hade vi tänkt sätta in subrutinens namn (VECKODAG.BAS el dyl). Om vi får många subrutiner till oss kommer vi att sammanställa ett register som vi publicerar i bladet. Rutinerna kommer då naturligtvis att finnas att hämta på den nya monitorn.

Efter rad 200 kommer en del beskrivningar av rutinen. I exemplet står det bara "Datum ger veckodag" som med några få ord förklarar vad rutinen gör. Lite längre ner står vem som har gjort rutinen eller vem som har skickat in den. I det här fallet är rutinen plockad ur ett program som var insänt till programbanken.

Man skall också ange typ av dator, t ex ABC80, ABC800M, ABC800C ABC802 eller ABC806. Ange i förekommande fall typ av disk och DOS.

Efter en rad med "++++" kommer en förklaring "gäller i första hand ABC80" på ingående variabler i rutinen. Efter ytterligare en rad "+++" börjar subrutinen.

Redaktionen vill gärna ha förklarande text vid sidan av den aktuella subrutinen. Desto utförligare desto bättre.

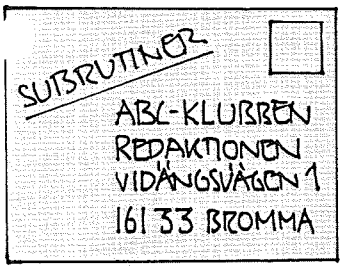
Om det går är det bra om programmet skrivs med korta rader för att det skall vara lättare att publicera. Titta t ex på rad 440 och 450 som har skrivits på två rader i stället för en.

All insänt material till redaktionen skall vara i maskinläsbar form typ Kassettband eller flexskiva.

Redaktionen återsänder kasett och flexskivor så snart som redaktionen har hunnit kopiera dem.

Rune Mattsson <904>

Subrutiner och text sändes till:




```
10 REM ====================================
20 REM TESTPROGRAM------------------------
30 M%=7% : D%=25% : Y%=84%
40 GOSUB 200
50 ; V$
60 END
70 REM SLUT TESTPROGRAM-------------------
80 REM ====================================
190 REM
200 REM VECKODAG.BAS ++++++++++++++++++++
210 REM Datum ger veckodag.
220 REM Typ av dator = ABC80
230 REM Testad på ABC80
240 REM
250 REM Inskickad av Rune Mattsson <904>
260 REM ++++++++++++++++++++++++++++++++++
270 REM M%=MÅNAD
280 REM D%=DAG
290 REM Y%=ÅR t ex 1984 anges som 84
300 REM V$=Resultat dag
310 REM I%=Slaskvariabel
320 REM ++++++++++++++++++++++++++++++++++
330 D%=D%+1%
340 IF M%<3% M%=M%+12% : Y%=Y%-1%
350 Y%=Y%+Y%/4%-Y%/100%+Y%/400%
360 M%=M%+M%+.61*(M%+1%)
370 D%=D%+M%+Y% : D%=D%-D%/7%*7%
380 RESTORE 440
390 FOR I%=0% TO D%
400 READ V$
410 NEXT I%
420 RETURN : REM Åter till huvudprogram
430 REM
440 DATA MÅNDAG,TISDAG,ONSDAG,TORSDAG,FREDAG
450 DATA LÖRDAG,SÖNDAG
```

BREV.

Brev från Per Nilstad , bror till <4036>

Hej redaktionen
Här är tre problem som ni kan sätta in i ABC-bladet om ni vill. Jag är inte själv med men min bror är det.

Per Nilstad
Georg Nyströmsgatan 23 E
644 00 TORSHÄLLA
Min bor är medlem <4036>

Här får Ni tre små problem att lösa

1. Varför ger inte ABC80 felmeddelande då man skriver följande rader direkt på ABC80:ns skärm utan att använda som av citationstecken?

```
; LANDSORT
; KAN DU BORRA
; NORDKOREA
```

2. När man trycker på Ctrl-E (eshappe) får man ASCII-värdet 0 (noll). Man kan erhålla samma ASCII-värde genom att använda en annan tangent. Vilken?

3. Har Du löst problem 1 inser Du enkelt att man också kan skriva följande rader

```
; BANDA
; LANDA
; SANDA
```

Men varför blir det felmeddelande då man försöker skriva

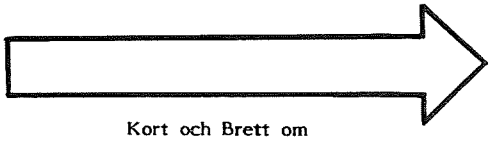
```
;TANDA
```

SVAR

1. Datorn tolkar texten som en logisk funktion, d v s L AND S OR T eller t ex NORGE som blir N OR G E.

2. Om man trycker på SHIFT+CTRL+P så erhåller man ASCII-värdet noll. Liksså finns det en annan tangent för Ctrl-Å, Ctrl-Ä, Ctrl-Ö och Ctrl-ü.

3. T AND A är visserligen en logisk funktion men datorn tolkar det som om man skall använda den matematiska funktionen TAN (=tangens) och då inser man att det måste bli ett ERROR-meddelande.



Kort och Brett om

DEC-10.

Här kommer nu dom sista 16 sidorna av skriften Kort och Brett om DEC-10.

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

Joystick interface

ABC80

för ABC80,s V24-kontakt.

I många program har man ibland saknat någon enkel variant av cursorstyrning som inte använder div. kontroll-koder och likn. För att inte tala om alla spel som kan bli mer lättmanövrerade via en joystick.

Visserligen har det funnits joysticks till ABC80 tidigare men det har inte fått någon vidare spridning. Därför tänkte jag visa hur jag har gjort.

Det enklaste stället att ansluta någon yttre enhet borde ju ..a via V24-kontakten, men där finns ju bara 3 ingångar och 2 ..gångar. Alltså får man försöka med multiplexing.

Id'n till den här varianten fick jag av Christer Lerin. Principen är att man tittar på en ingång i taget och lägger ihop dem till en byte som då visar alla ingångar på en gång.

Om man tittar på kopplingsschemat nedan ser man hur IC2 & IC3, som är en '1-of-8 dataselector/multiplexer', adresseras av en ..nare IC1 som man nollställer med pulser på en pinne 'res' och ..gar med den andra pinnen 'clk'. Efter varje stegning läser man av de båda ingångarna och översätter vid behov till lämpligt bitmönster beroende på hurdan sticka man har. IC3, 4 & 6 är buffertkretsar för att få ett entydigt omslag. Man måste naturligtvis inte ansluta just en joystick utan det går bra med vilken strömbrytare som helst som ger en slutning till jord.

Kretsen byggs enklast på en bit Veroboard. Observera att man kan utesluta IC5 & 6 om man bara tänker använda en joystick. Anslutningarna sker via 9-poliga "Cannonkontakter en hona mot ABC80 och en eller två hanar mot stickan(orna). Kretsen bör lämpligast byggas in i en lagom låda med sladdar till ABC80 och till stickan. Kretsen får strömförsörjning via en enkel stabiliseringskrets som matas från +12v i V24-kontakten.

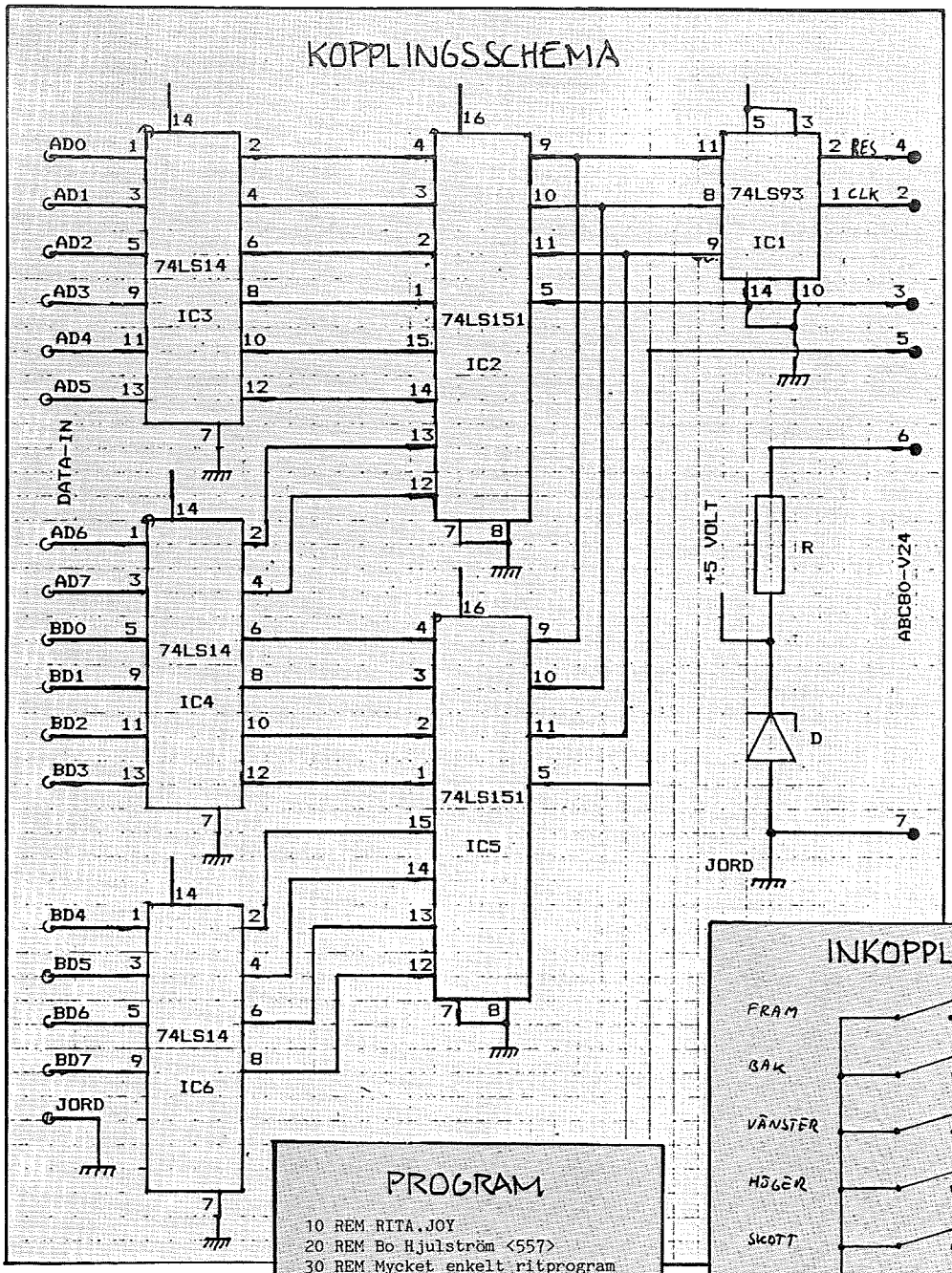
Programmet är ganska enkelt men bör skrivas i assembler för hastighetens skull. Assemblerlistningen är gjord för två stycken ATARI-kompatibla joysticks. Om någon har en annan typ av sticka så kan jag visa hur man ändrar programmet. Se flödesschema och programlista.

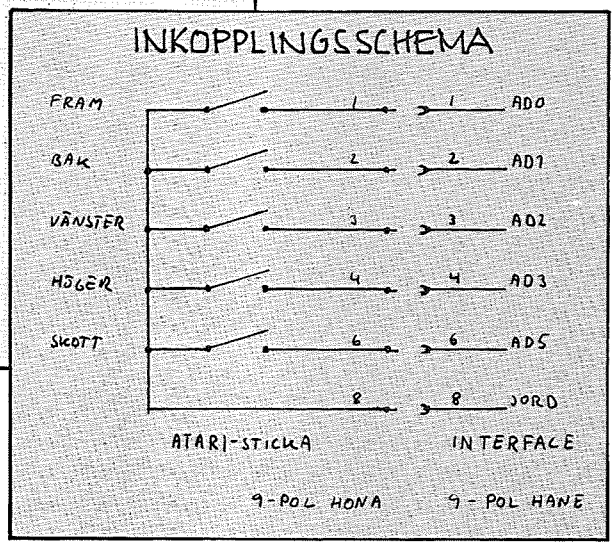
Hur använder man då drivrutinen ? Jo, rutinen anropas t.ex. ' Z%=CALL(64512%) ' och sedan 'maskar' man fram vilka bitar som är satta t.ex ' IF Z% AND 1% THEN ... ' för att se om stickan är i främre läget och ' IF Z% AND 16% THEN ... ' om eld-knappen är tryckt. Tänk på att flera bitar kan vara satta samtidigt t.ex. framåt,vänster & eld. Drivrutinen bör inte ligga i programmet ..m använder den utan läses in först eftersom man annars blir ..t till en viss joystick. Programmet bör kolla att rutinen finns innan man anropar den, annars dyker ABC80. Se programexempel.

Om ni har synpunkter på bygget får ni gärna höra av er till mig. Jag ska försöka visa anpassningar till några av klubbens program i nästa nummer av ABC-bladet.

VH Bo Hjulström <557>

```
        TITLE   'JOYDRV'
        ;
        ;
        ; Vers. 2.0
        ; 22 Mar. -84
        ; av (c) Bo Hjulström
        ;        Håkan Järbin
        ;
        ; Rutin för att läsa av ett
        ; interface på V24-kontakten.
        ; Denna version läser av
        ; två stycken ATARI-kompatibla joysticks.
        ;
V24     EQU     58
        ;
        ;
        ORG     0FDB1H          ; Beskrivning för dosbuffert
        ;                       ; §7 (64945).
        DEFB 00                 ; Markera den upptagen.
        ;
        ;
        ORG     0FC00H          ; Start för dosbuf §7 (64512).
        ;
        ;
        ; Nollställ räknaren på interfacet.
        LD      A,11100111B
        OUT     (V24),A
        ;
        LD      A,11111111B
        OUT     (V24),A
        ;
        ; Initiera räknare och pekare.
        LD      DE,0000         ; Resultat.
        LD      B,8             ; Bit räknare.
        LD      HL,INORDN       ; Tabell över bitvärden.
        ;
        ; Läs av sticka 1 (V24-pinne 3).
LAES1   IN      A,(V24)
        AND     00000001B
        JR      NZ,LAES2        ; Kontakt sluten ?
        ;
        LD      A,(HL)          ; Ja, sätt bit
        OR      E
        LD      E,A
        ;
        ; Läs av sticka 2 (V24-pinne 5).
LAES2   INC     HL
        ;
        IN      A,(V24)
        AND     00000010B
        JR      NZ,LAES3        ; Kontakt sluten ?
        ;
        LD      A,(HL)          ; Ja, sätt bit
        OR      D
        LD      D,A
        ;
LAES3   INC     HL
        ;
        ; Stega interface-räknare (V24-pinne 2).
        LD      A,11110111B
        OUT     (V24),A
        LD      A,11111111B
        OUT     (V24),A
        ;
        ;
        PUSH    BC              ;
        LD      B,8             ; Kort väntan för
DELY    DJNZ    DELY            ; säkrare avläsning.
        POP     BC              ;
        ;
        ; Läs av alla 8 ingångarna på interfacet.
        ;
        DJNZ    LAES1           ; Läst av 8 ggr ?
        ;
        ; Klart. Ta med resultat till basic.
        ;
        LD      H,D             ; Resultat joystick 1.
        LD      L,E             ;    -"-   joystick 2.
        ;
        ;      Höga byten        Låga byten
        ;      0 0 0 S V H B F   0 0 0 S V H B F
        ;           k ä ö a r         k ä ö a r
        ;           o n g k a         o n g k a
        ;           t s e å m         t s e å m
        ;           t t r t å         t t r t å
        ;             e   t             e   t
        ;             r                 r
        ;
        RET
        ; --------------------------------
        ; Konverterings tabell bitar in -> bitar ut
        ; Utbit 76543210        Sticka,Ingång
INORDN  DEFB    00000001B   ; 1       1  (Framåt)
        DEFB    00000001B   ; 2       1
        DEFB    00000010B   ; 1       2  (Bakåt)
        DEFB    00000010B   ; 2       2
        DEFB    00001000B   ; 1       3  (Vänster)
        DEFB    00001000B   ; 2       3
        DEFB    00000100B   ; 1       4  (Höger)
        DEFB    00000100B   ; 2       4
        DEFB    00000000B   ; 1       5
        DEFB    00000000B   ; 2       5
        DEFB    00010000B   ; 1       6  (Skott)
        DEFB    00010000B   ; 2       6
        DEFB    00000000B   ; 1       7
        DEFB    00000000B   ; 2       7
        DEFB    00000000B   ; 1       8
        DEFB    00000000B   ; 2       8
        ;
        END
        ; ------------------------------
```





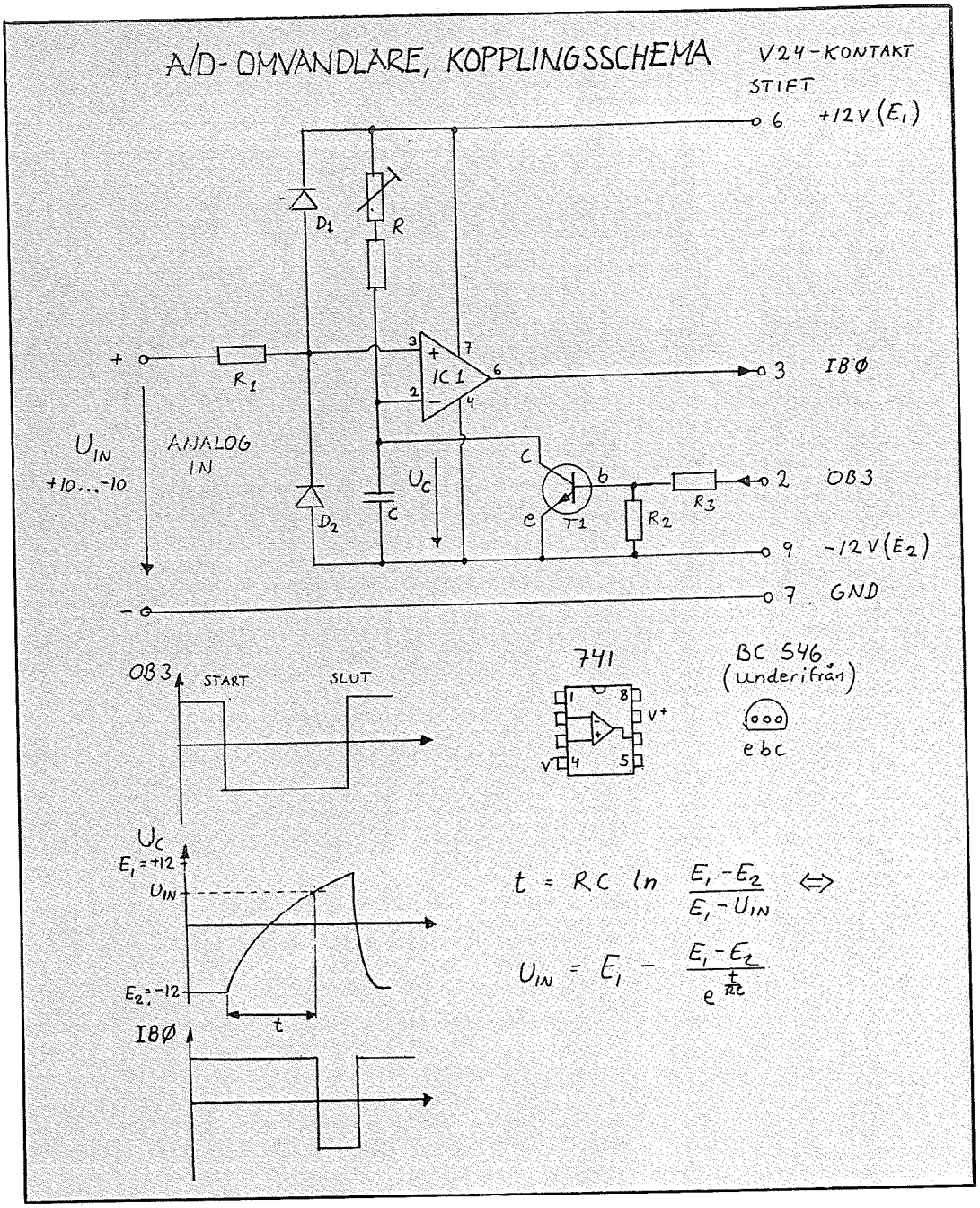
PROGRAM

```
10 REM RITA.JOY
20 REM Bo Hjulström <557>
30 REM Mycket enkelt ritprogram
40 REM
50 REM Erfodrar drivrutin ,
60 REM samt joystick-interface
70 REM 16k minne, 40 tkn skärm
80 REM
90 REM
100 IF PEEK(64512%)=62% GOTO 120
110 ; 'Drivrutin saknas.' : STOP
120 X%=30 : Y%=30
130 REM GRAFIK
140 ; CHR$(12%)
150 FOR I=0 TO 23
160 ; CUR(I,0);CHR$(151%);
170 NEXT I
180 REM FLYTTA
190 Z%=CALL(64512%)
200 IF Z% AND 1% X%=X%-1
210 IF Z% AND 2% X%=X%+1
220 IF Z% AND 4% Y%=Y%+1
230 IF Z% AND 8% Y%=Y%-1
240 IF Z% AND 16% GOTO 130
250 SETDOT X%,Y%
260 FOR I=1 TO 100 : NEXT I
270 GOTO 190
```

DRIVRUTIN

```
1 REM  Drivrutin för interface via V24.
       För två ATARI-stickor.
2 POKE 64512%,62%,231%,211%,58%,62%,
       255%,211%,58%,17%,0%,0%,6%,8%,
       33%,49%,252%
3 POKE 64528%,219%,58%,230%,1%,32%,3%,
       126%,179%,95%,35%,219%,58%,230%,
       2%,32%,3%
4 POKE 64544%,126%,178%,87%,35%,62%,
       247%,211%,58%,62%,255%,211%,
       58%,16%,226%,98%,107%
5 POKE 64560%,201%,1%,1%,2%,2%,8%,8%,
       4%,4%,0%,0%,16%,16%,0%,0%,0%
6 POKE 64576%,0%
```

A/D omvandlare.



I boken "Avancerad programmering på ABC 80" finner man en Analog/Digital-omvandlare, som kan avkänna en potentiometer. Med samma princip, men med en något annorlunda konstruktion kan man mäta likspänningar.

Omvandlaren ansluts via V24-snittet och styrs av en drivrutin i maskinkod.

Prestanda:

Mätområde	+-	10 Volt
Onogrannhet	<	2% +- 30 mV
Upplösning	ca.	10 mV
Mättid	ca.	0.1 s
Inimpedans		1 Mohm

Funktion: En kondensator laddas upp genom ett motstånd. Spänningen över kondensatorn jämförs med spänningen som ska mätas. När dessa blir lika skickas en puls till datorn, som då stoppar tidmätningen den har gjort under uppladdningen.

Sedan återstår lite räknande med EXP-funktionen för att få fram spänningen. Datorns 12 Volt-matningar fungerar som referenser. Tyvärr ändras de något då datorn värms upp, och därför kommer mätresultatet att driva ca. 30 mV

Intrimning:

Värm upp datorn 30 min.
Starta testprogrammet.
Kortslut A/D-omvandlarens ingång.
Justera trimpoten tills datorn skriver noll volt eller däromkring.

Om möjligt: mät upp plus/minus 12V-matningarna med digitalvoltmeter, och lägg in värdena i rad 120 (testproget). Trimma därefter enligt ovanstående.

För att få omvandlaren snabbare kan RC-produkten minskas. Upplösningen blir dock sämre.

Komponentlista:

IC1	Operationsförstärkare 741
T1	Transistor BC546
C	Kondensator. 0.47 mikrofarad (polyester)
R	Trimpot 100 kohm + motstånd 47 k
R1	Motstånd 10 kohm
R2	" 330 ohm
R3	" 4.7 kohm
D1,2	Diod 1N4148
	V24-kontakt

D1,D2 och R1 är skyddskomponenter - kan tas bort på egen risk!

D1,D2 och R1 är skyddskomponenter - kan tas bort på egen risk!

Montering sker lämpligen på en bit Vero-Board.

Hälsningar från <1423>
Erik Åström
Köpmang. 7
960 40 Jokkmokk

```
; Drivrutin för enkel A/D-omvandlare
; Erik Åström 840609
;
          ORG 65408
START     LD HL,0   ;Nollställ tidmät
          IN A,(58)
          SET 3,A
          OUT (58),A ;Starta mätning
          LD DE,(65008) ;Avläs klocka
;
;Mätrutin - mäter pulsens längd
; Ett varv tar 20 mikrosek. HL räknar!
;
LISTEN    IN A,(58)  ;Avläs inpuls
          BIT 0,A    ;Puls slut??
          JR NZ,RESET
;
          INC HL     ;Öka tidmätare
          BIT 6,H    ;Oändlig tid?
          JR NZ,RESET ;Ja-bryt
;
          JR LISTEN  ;Puls kvar.
;
RESET     RES 3,A    ;Ladda ur kond
          OUT (58),A
;
; Hur många NMI har gjorts under
; pulsmätningen? (Ett NMI tar 35 us)
; Ta klockförändringen * 2 och addera
; den till HL så blir det bra!
;
          LD A,(65008) ;Läs klocka igen
          LD D,A
          LD A,E
          SUB D        ;Klockändring
          RL A         ;*2
          LD E,A
          LD D,0
;
          ADD HL,DE    ;Kompensera NMI!
          RET
;
          END
```

```
10 REM Demoprogram för enkel A/D-
20 REM omvandling. Källkod AD.ASM
30 REM Erik Åström <1423> 840609
40 REM
50 POKE -128%,33%,0%,0%,219%,58%,203%,223%,211%,
        58%,237%,91%,240%,253%,219%,58%
60 POKE -113%,203%,71%,32%,7%,35%,203%,116%,32%,
        2%,24%,243%,203%,159%,211%,58%
70 POKE -98%,58%,240%,253%,87%,123%,146%,203%,
        23%,95%,22%,0%,25%,201%
80 REM
90 ; CHR$(12);CUR(6,7)'A/D Omvandling';CUR(8,6)
   'Inspänning (volt):'
100 REM ----- Beräkningsdel--------
110 REM
120 E1=12.25 : E2=-12.2
130 R=1E+5 : C=4.7E-7
140 REM ---------------------
150 T=CALL(65408)/50766
160 U=E1-(E1-E2)/EXP(T/(R*C))
170 REM ---------------------
180 ; CUR(10,10)U'
190 FOR I=1 TO 1000 : NEXT I : GOTO 150
```

DIV:

Lokalavdelningar

På de kort som följde med inbetalningskortet för medlemsavgiften fanns det en lapp som man kunde skicka in och ange om man ville vara med i en lokalavdelning. Ett hundratal sådana kom in.

De lappar som angav att en medlem ville vara med i en lokalavdelning har vidarefordrats till den lokala lokalavdelningen det finns en sådan inom området.

En del ville ha uppgifter om vilka lokalavdelningar som finns. Tyvärr finns det just nu bara tre stycken. Intresset för en ny lokalavdelning verkar vara störst i Skåne/Malmö. Är det någon i Skåne eller i någon annan del av Sverige som vill bilda en lokalavdelning så ta kontakt med ABC-klubbens styrelse.

Lokalavdelningar måste alltså bildas på lokalt initativ, men ekonomsikt stöd från ABC-klubben kan erhållas.

För närvarande finns det följande lokalavdelningar:

ABC-Stockholm Bo Hjulström	Stockholm 08-7772753
ABC-Väst Anders Lundberg	Göteborg 031-117572
ABC-Öst Nils Larsson	Linköping 013-134565

För närmare information om dessa lokalavdelningar hänvisar jag till artiklar i ABC-Bladet.

Bo Kullmar

ASSEMBLER, vad är det?

Ordet ASSEMBLERPROGRAM kan tolkas (eller misstolkas) som två saker:

1. Ett program som konverterar källkod (med mnemonic mm) till objectkod (i binärt, skinläsbart format).
2. Källkod (innehållande mnemonic mm) som ska assembleras.

Ett sätt att unvika missförstånd är att i stället för ASSEMBLER eller ASSEMBLERPROGRAM i stället använda orden:

1. ASSEMBLATOR: Assemblerar; Omvandla källkod till objectkod i binärformat. T.ex. "Använd en Z80 assemblator".
2. ASSEMBLY: Ett format för källkod, 'språket' om man så vill T.ex. "Programmet är skrivet i assembly" eller "Titta ett assemblyprogram".

Motsvarande förvirring kan även orsakas med ordet DISASSEMBLER.

I en artikel i ABC-bladet nr 1.1984 med rubriken "DISASSEMBLERN, VAD ÄR DET?" borde dokumentet som beskrivs benämnas "DISSASEMBLERINGEN". De som inte vet vad artikeln syftar på kan lätt tro att den handlar om ett fantastiskt program (DISASSEMBLATOR) som kan förklara hur ABC80 fungerar.

David Andersson

RESET ABC80

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

Jag hade i alla fall lyckats med detta konststycke. Men som sagt var nöden är uppfinningarnas moder.

Jag hade ingen större lust att medelst PEEK leta mig fram till radslutet och räkna bytes därimellan. Utan jag satte mig ner och tänkte (det var ett ganska stort program jag hade förlorat). Jag kom att tänka på BASIC - tolkens kompileringsrutin. Den skulle få göra jobbet åt mig.

Jag skrev POKE <BOFA>,0

Detta lurar datorn att tro att det finns ett program i minnet. Efter RESET står det 1 där.

Sedan LIST <FILNAMN>

Detta för BASIC-tolken behöver ju bara Kompilera .BAS - filer. Och jag hoppades också att han på samma gång skulle räkna ut den riktiga CHECK-Summan.

Detta gjorde han också.

Sedan skrev jag SAVE <FILNAMN>

Detta för att spara den kompilerade verisionen.

Och det var med glädje jag kunde skriva RUN - och programmet fungerade.

Lars-Börje Cid <2839>
Bøndegatan 24 B
641 45 Katrineholm

Ytterligare förslag om JA eller NEJ.

I förra ABC-bladet fanns två artiklar om hur man enklast läser in svar av typen JA eller NEJ. Här är en variant till :

```
DEFFNJ% = ( INSTR (1,'JjYy'+CHR$(13%),
X$) ) <> 0% DEFFNN% = ( INSTR (1,'Nn'+
CHR$(13%),X$) ) <> 0%
```

Används t.ex :

```
100 PRINT ' .....? (J) '; : GET X$ : ; X$
110 IF FNJ% THEN sant ELSE falskt
```

Funktionen godtar stora och små bokstäver samt enbart return som svar.

Bo Hjulstöm

Två små BASIC II-tips

Vill man använda oändliga loopar i BASIC-II så kan man skriva så här:

```
WHILE -1
! Gör något
WEND
```

WHILE -1 är nämligen alltid sant.

Vill man använda en struktuerad IF-sats så saknas den i BASIC II. Om den hade funnits så hade man skrivit:

```
IF <villkor> THEN
! Gör något
IFEND
```

Nu går inte detta, men troligen så går det i BASIC III, dvs BASIC:en till DS90. Man kan ändå fixa det i BASIC II med en WHILE-sats. Skriv så här:

```
WHILE <villkor>
! Gör något
IF 0 WEND
```

Sista satsen är alltid falsk, så WHILE-loopen genomlöps alltid bara en gång.

Bo Kullmar

Rättelse av LÖKAREN ABC80

På kassett nr 12 fanns programmet LÖKAREN, men tyvärr i en version som hade några buggar. Författarna har rättat dessa.

En fil har lagts in på ABC-monitorn i Alvik men för en större spridning är ABC-bladet bättre.

Problemet yttrade sig i att program som lagrats med SAVE ej gick att åter läsa in i ABC80:n. För att Du skall kunna använda SAVE på Din dator gör följande:

1. Tryck RESET.
2. Ladda in LÖKAREN i den form som den fanns på kassetten.
3. Gör sedan dessa små POKE-satser:

```
POKE <BOFA+1493>, 13
POKE <BOFA+1562>, 13
POKE <BOFA+1722>, 13
POKE <BOFA+1808>, 13
```

där <BOFA+nn> betyder att adressen skall vara BOFA:n plus respektive adress. BOFA:n är ju beroende på hur mycket minne Din ABC80 har och var programmet hamnar i internminnet. Fullt utbyggt är den 32768.

4. Gör sedan SAVE LÖKAREN <RETURN> och Du har en kraftfull hjälp vid kodning och redigering av program.
   Programmet behöver laddas in vid varje programmeringssessions början. Det finns kvar i maskinen så länge Du inte använder <RESET>.

Med hälsningar från kassettredaktionen

V24 Channel B för ABC800-serien

Jag har gjort en del tester för att komma underfund med hur man bär sig åt för att dels läsa av modemsignaler, dels sätta vissa modemsignaler. Här är vad jag har kommit på genom tålmodigt prövande:

IN-signaler:

CTS på stift 5
DCD på stift 8

För att kunna avläsa förändring av dessa signaler måste man först sända kommandot OUT 65,16. Sedan kan man läsa av porten med INP(65). CTS motsvarar då bit 5 och DCD bit 3.

UT-signaler:

DTR på stift 1
RTS på stift 4

För att sätta dessa signaler måste man först ge kommandot OUT 65,5. Därefter sättes DTR med OUT 65,128 (bit 7) och RTS med OUT 65,2 (bit 1). Bägge sättes samtidigt med OUT 65, 130. Vill man sedan slå av dem ger man kommandot OUT 65,24.

Jag har inte kommit underfund med hur man gör om man endast vill slå av en av signalerna.

På liknande sätt kan man styra kassettutgången. Porten är då 67. För att slå på bandspelarens motor sätter man DTR genom OUT 67,5 och OUT 67,128. Stoppar den gör man med OUT 67,24.

Gunnar Tidner

MIN HJÄLP AV

SUPERSMARTAID.

ABC80

Jag gör i mitt dagliga arbete mycket programmeringsarbete. Dock lämnar ABC80 dock övrigt att önska i många avseenden. Jag har tidigare haft en hjälprutin som tyvärr måste läsas in i minnet varje gång. tta har ju fungerat, men nu finns något ttre.

SUPERSMARTAID från OWOCO AB som alltid finns till hands och som dessutom har vissa kommandon som är unika i sitt slag.

JOB-strömmar

Genom ett visst kommando kan Smartaiden tolka en textfil som en ström av instruktioner. Detta utföres med >>JOB textfiln.amn (RETURN)<< som tar text från en textfil och utför det som står där.

Vid start eller reset av ABC80 söker Supersmartaiden efter en fil med namnet SUPER.JOB. Om denna hittas utförs de kommandon som står däri.

Tack vare detta kan jag stoppa in en skiva och trycka på resetknappen och genast så laddas terminalrutinen in i datorn med rätt parametrar. Därefter körs terminalprogrammet igång och jag behöver bara ringa upp ABC-klubbens monitor.

På så sätt erhålls även automatisk uppstart av ABC80 om datorn används till t ex övervakning el dyl. Efter strömavbrott körs programmen automatiskt igång!

Dessa jobfiler kan lämna indata till de program som körs. På detta sätt kan de flesta program köras i en förutbestämd följd.

T ex om jag har ett program som skriver ut brev på skrivaren och programmet frågar er filens namn som ska skrivas ut så anger jag detta i en fil.

```
>> RUN SKRIVUT _M BREV1.DAT _M
RUN _M BREV2.DAT _M (RETURN)<<
```

Denna rad sparar jag som en textfil med namnet SKRIVUT.JOB.

När jag ger kommandot >>JOB SKRIVUT.JOB (RETURN)<< görs följande: RUN SKRIVUT (_M = CTRL-M = RETURN för Supersmartaid) kör igång programmet som frågar efter ett filnamn. BREV1.DAT hämtas så in till programmet och utskrift sker. När programmet är klart ges ett nytt RUN och filen BREV2.DAT skrivs ut.

På detta sätt kan man utföra hur många saker som helst så länge som inga fel uppstår, då spårar det hela ur.

Allt detta bygger i sin tur på att ABC80s inmatningsbuffert är utökad till flera hundra tecken. Om vi t ex tar ett stort program så tar det ju ca 10 sekunder innan något händer efter RUN. Under denna tid kan man ge de data som programmet senare kommer att fråga efter. Inget syns än på skärmen men när programmet börja fråga så skrivs det in automatiskt till programmet!

Detta är toppen vid program som t ex läser något från disken. Då slipper man sitta och vänta utan kan lugnt fortsätta skriva in nya data.

Text-editorn

De textfiler som utför dessa JOB kan direkt skrivas in i minnet via smartaiden. I denna finns nämligen en editor som tillåter att vanlig text matas in istället för program.

Allt man behöver göra är att efter NEW börja första raden med en asterix, >> 10 * (RETURN)<<. Genom detta ställer Smartaiden om sig och inget av det man skriver in kompileras.

Den rad som innehåller en asterix kan senare tas bort. Texten som skrivits in sparas enkelt på skivan med >>TSAVE namn (RETURN)<< och hämtas in med TLOAD namn. Genom TMERGE läses en ny textfil in i minnet och skarvas på den som redan ligger där.

I övrigt kan alla kommandon utföras på texten såsom att ta bort rader, lägga till, omnumrera osv.

Denna artikel är föresten skriven på detta sätt. Den upptar ca 9 kbyte text. Även i en oexpanderad ABC80 finns det plats för ca 5000 tecken till!

Tack vare dessa kommandon kan vilken sekventiell datafil som helst läsas direkt in i ABC80 och beskådas. Vill man inte radera det program som finns i minnet används >>DISP filn.amn (RETURN)<< som visar alla typer av sekventiella filer direkt på skärmen.

Genom detta kan t ex en rutin från ett annat program listas upp på skärmen. Därefter används skärmeditorn och de aktuella raderna läses in i det program som finns i datorn. En rutin som en gång skrivits kan läggas till andra med ett minimum av arbete!

Vid programskrivning vill i alla fall jag ofta se vad som skrivits in, och då används LIST. Genom att istället skriva >>^ (RETURN)<< listas programmet också, men det går mycket fortare att skriva.

Listfunktionen visar förövrigt inte mer än vad som rymms på en skärmbild åt gången. Varje gång man listar sker detta från det radnummer där man sist slutade, vilket är mycket praktiskt! Genom att skriva LIST- startas dock listningen från början.

Listningen kan scrollas mot högre radnummer precis som vanligt, men med <-- pilen kan denna listning backas mot lägre radnummer!

Ofta när jag skriver program kommer jag på att en viss funktion redan skrivits. Tyvärr så ligger denna inte som en egen subrutin utan inklämd mitt i någonstans. Genom utökningen av kommandot REN klaras detta upp snabbt. Om jag på raderna 610 - 730 har en rutin som jag vill bryta ut och lägga som en egen subrutin behövs bara >>REN 2000,10 FROM 610-730 (RETURN)<<.

Nu har raderna 610-730 flyttats till 2000 och framåt. De ställen i programmet som hoppar hit har också ändrats till 2000. Detta kommando är ovärderligt!

Ändra utseendet på tangentbordet

Genom JOB-kommandot växer ABC80 avsevärt. De gränser som finns är väl snarast fantasins. Nu finns det faktiskt ytterligare en sak med smartaiden som kittlar fantasin, nämligen KEY-funktionerna.

En keyfunktion är detsamma som att omdefiniera tangentbordet. Jag kan t ex göra så att skärmen töms och listningen startar från början. Allt detta kräver endast en tangentnedtryckning.

Genom raden >>KEY _^="_L Q_L_Ä^-_M" (RETURN)<< kommer smartaiden ihåg att CTRL-^ (_^) skall lista programmet. Det behövs inget return efter. Utan att gå alltför djupt betyder raden ovan följande: > Ersätt CTRL-^ med "töm skärmen + LIST + RETURN" > De CTRL kommandon som ovan används ingår i den skärmeditor som finns i Supersmartaiden. _L betyder CTRL-L = CHR$(12)!

För de som har 80-teckenstillsats kan följande göras: KEY _O="IF PEEK(828)=40 ;INP(4)CHR$(12) ELSE ;INP(3)CHR$(12) _M" Varje gång (CTRL-O) (utan RETURN) trycks ned byts radlängden på skärmen och densamma töms!

Alla tangenter, med eller utan CTRL, kan göras om. Dessa KEY-funktioner kan tom lagras på en fil och läsas in vid behov.

När man programmerar läser man t ex in en fil som gör om alla små tecken till basic-kommandon. Man får då en typ av entangentsbasic som man själv definierar. (t ex i=INPUT, g=GOTO, r=REM, t=THEN o.s.v)

Genom följande erhålls oanade möjligheter: >>KEY _Å="JOB INITPR.JOB _M" (RETURN)<< En enda tangentnedtryckning kan alltså starta ett JOB från en textfil som i sin tur kan starta nya JOB som i sin tur och så vidare.

Felsökning

När CTRL-C trycks ned avbryts inte programmet helt utan det stannar bara. Därefter kan följande göras:

(CTRL-C) som stoppar programmet helt,

(mellanslag) som gör att programmet fortsätter,

(CTRL-S) som kör nästa programrad och stannar igen,

(CTRL-T) gör samma sak men radnumret skrivs ut eller

(CTRL-D) som slutligen visar hela nästa programrad som ska utföras. Detta är otroligt värdefullt om fel skulle uppstå och ett program av okänd anledning inte fungerar som avsett.

Behövs en snabb papperskopia av vissa programrader eller lösningar som ett program direkt skrivet på skärmen, behöver inte hela programmet göras om för att dirigera texten till skrivaren.

Genom att endast trycka ned (SHIFT-CTRL-O) får man en avskrift av skärmen direkt på skrivaren. Smartaiden innehåller naturligtvis en printerrutin, och självklart klarar den av en ABC80 med 80-tecken.

Sammanfattning

Supersmartaiden växer man ihop med, lär sig uppskatta och endast fantasin sätter gränser för vad som kan göras. Det tog mig ca en månad innan allt gick naturligt. Bara detta att inte skriva LIST utan bara (CTRL-tyskt y) tog sin tid, men vilken enorm tidsbesparing, att endast behöva trycka ned en enda tangent!

Hela Supersmartaiden fungerar på liknande sätt. Jag har med flit utelämnat mycket därför att vissa saker måste upplevas för att man ska tro på det!

ABC80 utan supersmartaid är numera för mig en hemsk tanke. Finns det verkligen någon som på fullt allvar kan vara utan bilskärmseditorn, texteditorn, jobfilerna, keyfunktionerna, tracefunktionen, omnumreringen . . .

Peter Stahl <1943>

ABC16.

På Datakraft84 mässan i Malmö visade Myab sin tillsats som gör en dator ur ABC800-serien till en 16 bitars dator. Tillsats är kanske lite att ta i, eftersom det är frågan om en ny CPU eller en ny dator. ABC800:an blir enbart terminal till ABC16. Senare kommer man också att kunna använda ABC16 till ABC80. Priset beräknas till 15000 kronor exklusive moms. ABC16 beräknas vara klar kring den 1 november och försäljningen börjar kring årsskiftet.

ABC16 känner automatiskt av om man kör ett ABC-program eller ett 16 bitars program. Program för ABCenix eller UNIX går inte att köra på ABC16. Uppgifterna i denna artikel om ABC16 grundar sig enbart på vad jag såg och hörde på mässan, samt på reklambroschyrer.

Vad kan ABC16 då användas till? Räcker det inte med ABC800 som den är?

Många avancerade programvaror utvecklas i dag för 16 bitars datorer. Vill man till exempel köra det populära programmet Lotus 1 2 3 så får man antingen investera i en ny dator eller "bara" köpa ABC16. Det är alltså främst om man vill köra program som normalt finns till datorer typ IBM PC, som man bör tänka på att köpa ABC16. En stor fördel med ABC16 är att man samtidigt som ny 16-bitars programvara kan köra sina ABC program. Totalt, jämfört med att investera i en ny dator, blir nog ABC16 ganska billig.

Myab kommer att tillhandahålla en hel del programvara till ABC16. En förteckning över dessa finns i broschyren för ABC16. Myab har konstruerat ABC16 i sammarbete med Luxor. Nu över till en lite mer teknisk presentation.

Hårdvara:

ABC16 finns i en låda av samma typ som ABC806:ans dator. Den diskkontroller som man redan har sätter man i ABC16. Där finns det då två diskkontroller, en till ABC:en och en till ABC16. Disken ansluts direkt till ABC16. Anslutning till ABC:en sker via en busskabel.

I ABC16 sitter troligen en Intelprocessor, typ 8088. Minnet är på 256 Kbyte, men går att bygga ut till 1 Mbyte. Dock finns en praktisk begränsning i dagens programvara till cikra 640 Kbytes.

Vill man ha utökad snabbhet vid beräkningar kan man ansluta en beräkningsprocessor, Intel 8087. Dock förutsätter denna processor att applikationsprogramvaran är anpassad till den. Plats finns för två IBM-tilläggskort i ABC16.

Operativsystem:

ABC16 körs under Digital Researchs Concurrent PC DOS. Under detta operativsystem kan man köra program som är gjorda för MSDOS och CPM86 samt en del program till IBM:s version av MSDOS, PCDOS. Detta innebär i praktiken att program som inte "går förbi operativsystemet och in i datorn den vägen", bör gå att köra.

Som namnet "concurrent" (=jämnlöpande) antyder, kan man dessutom köra flera program samtidigt. Fyra program kan köras på en gång.

Bakgrunden till tillkomsten av Concurrent PC DOS är värd en kommentar. Föresten kallar Myab det för Concurrent DOS, men enligt amerikansk fackpress heter det Concurrent PC DOS! När IBM skulle välja operativsystem till sin PC, så trodde alla, inklusive Digital Research,att just DR skulle få kontraktet. DR var ju marknadledande på 8 bitars maskiner genom operativssystemet CPM. Dock valde IBM, något överanskande, MSDOS från Microsoft! IBM:s val gjorde att MSDOS snabbt blev marknadsledande, trots dess brister! Detta innebar alltså att CPM versionen för 16 bitars datorer, CPM68 blev en flopp!

Nu ansåg sig Digital Research tvingade att göra ett nytt operativsystem som skulle vara kompatibelt med IBM:s, dvs MSDOS eller PCDOS som IBM kallar det för. Dessutom så skulle man naturligtvis kunna köra program för CPM86 också. Resultatet blev alltså Concurrent PC DOS, som alltså klarar program både för MSDOS och CPM86.

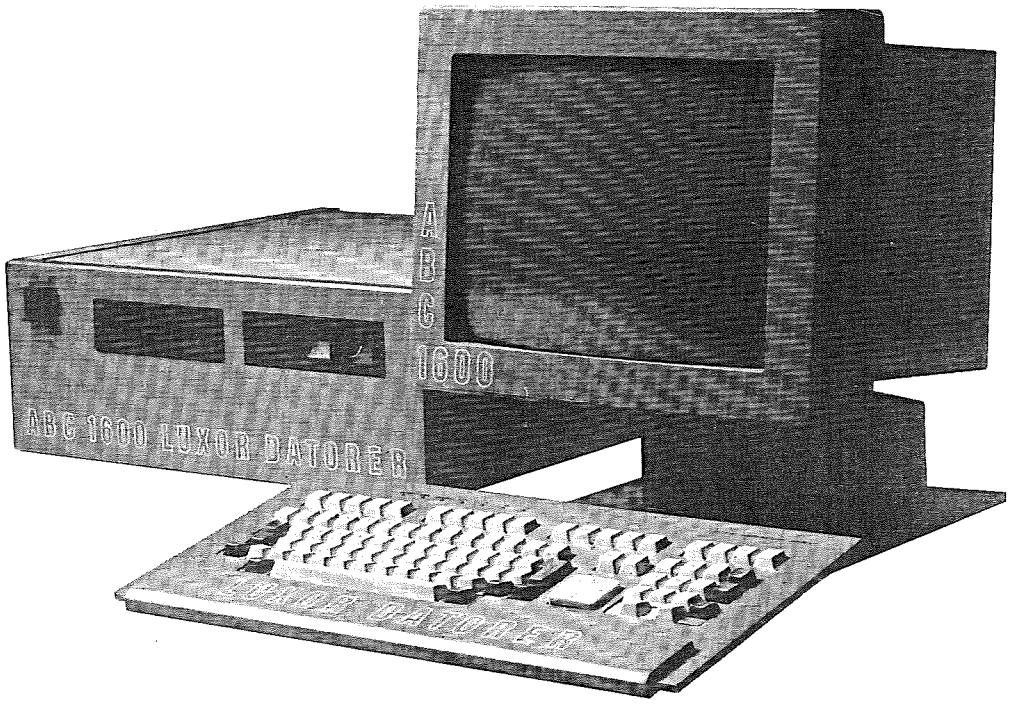
Jag tycker att Myab har gjort ett gott val. Dessutom så är det ju naturligt för Myab att satsa på produkter från Digital Research, eftersom man redan säljer deras CPM för 8 bitars maskiner till ABC-datorer.

Är ABC16 IBM PC kompatibel?

Nej, troligen inte i så stor utsträckning. Med IBM PC kompatibiltet menar man vanligen lite mer än att man skall kunna köra program för PCDOS. Allt bör efterlikna IBM PC. Kravet på fullständig IBM kompatibilitet uppfylles förresten av en trevlig svensk dator, Ericsson PC!

Bo Kullmar

ABC1600.



På DataKraft84, datormässan i Malmö visade Luxor för första gången sin nya supermikro ABC1600. Eftersom det är ett helt nytt system som ännu inte är helt klart för försäljning, när detta skrivs, så låter jag Luxor själv presentera det. Texten nedan är hämtad från broschyr som man lämade ut på mässan.

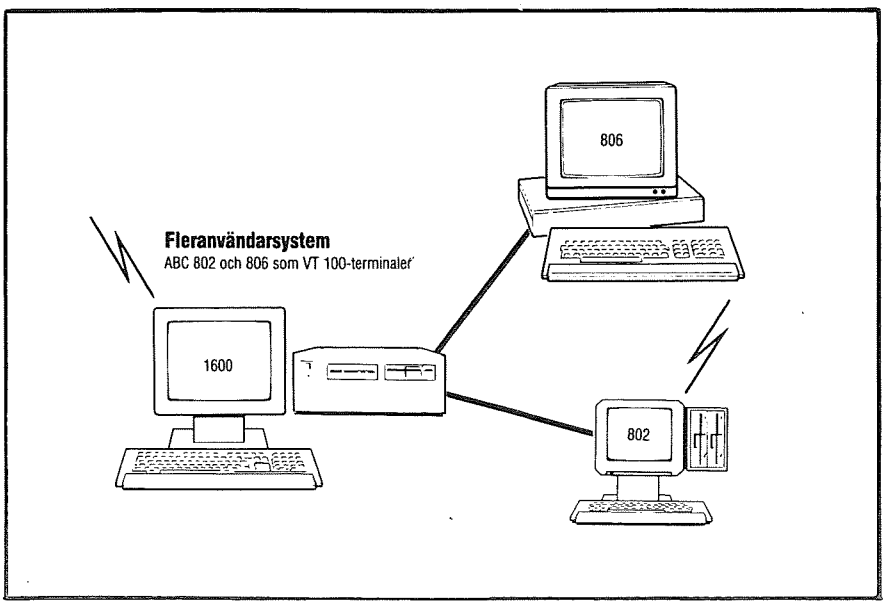
+ + +

System ABC1600.

ABC1600konceptet bygger på erfarenheter från ABC80 och ABC800serien. Kontinuitet, användarvänligt hård och mjukvara och ett starkt kommunikationskoncept är några begrepp, som vi har låtit gå i arv. Dessutom är ABC1600, precis som sin föregångare, ett helt öppet koncept att växa i. Ingen skall behöva betala dyrt för onödig kapacitet eller utrustning. Men ABC1600 representerar också ett helt nytt smådatortänkande. Den är kraftfullare, snabbare och mer mångsidigt än konventionella smådatorer. ABC1600 placerar sig därför i en helt ny nisch mellan micro och minidatorer, ABC1600 är främst en avancerad enarbetsplats med hög kapacitet. Men kan även med fördel användas i fleranvändarsystem. ABC1600 består i sitt grundutförande av tangentbordet ABC99, datordelen ABC1600 samt bildskärmen ABC1615. Datordelen innehåller förutom datorn en 13 Mbyte Winchester och en flexskiveenhet på 640 Kbyte. Operativsystemet är ABCenix. Som standard ingår även programspråket Basic III (inkl ISAM-hanterare) och MIMER relationsdatabashanterare. ABC1600 är väl lämpad för både administrativa och tekniska tillämpningar. ABC1600 är ett utmärkt hjälpmedel för krävande programmeringsarbeten och avancerade applikationstillämpningar. ABC800-serien starka kommunikationskoncept har blivit ännu starkare i ABC1600. Förutom kommunikationsmöjligheten med de flesta stordatormärken t ex IBM, DEC, UNIVAC, CDC m fl kan ABC1600 kommunicera med alla andra UNIXbaserade system. Dessutom kan du kommunicera lokalt i LUX-NET och genom Teletex 800 med samtliga telex och teletexapparater runt om i världen.

ABC1600 och två terminaler.

Du kan ansluta en eller två terminaler till ABC1600 (Fig 1). Inställningen av terminaltyp är mjukvarustyrd. Grundinställningen är VT 100, men du kan enkelt definiera önskad typ av terminal. De ABC800datorerna

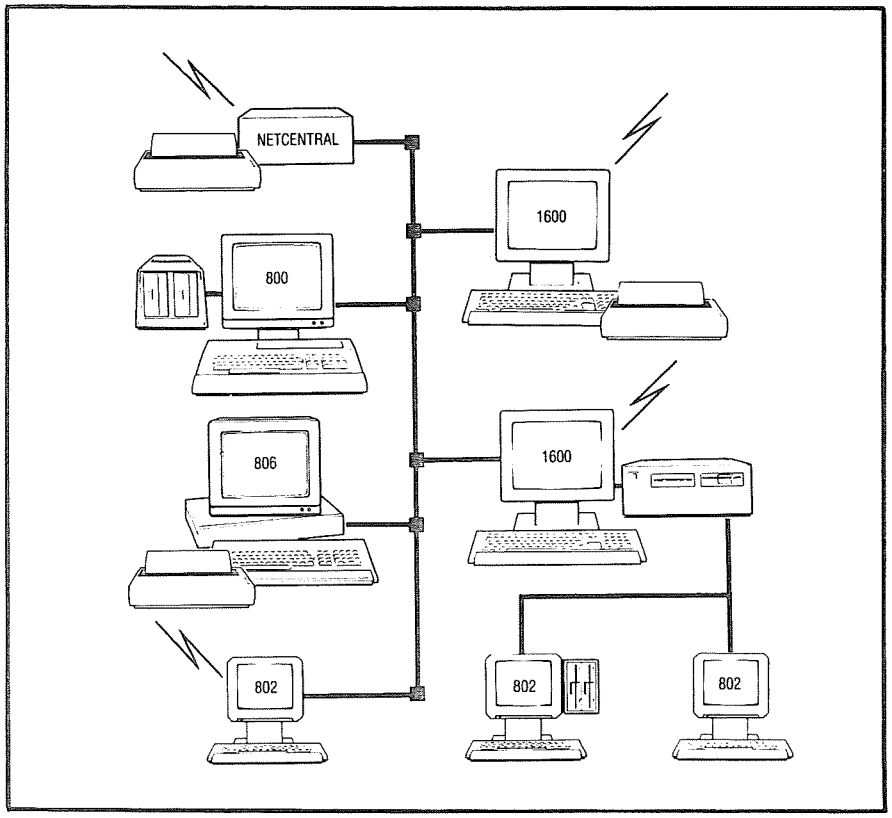


som du använder som VT 100-terminaler, emulerar du med ABC UTE.

ABC1600 i LUX-NET.
ABC1600 kan anslutas i LUX-NET antingen enbart ABC1600-datorer eller i kombination med andra ABC-datorer. Du kan fritt välja att köra ABCenix eller UFD-DOS från t ex en ABC802, ABCenix genom att logga in på någon av ABC1600-datorerna och UFD-DOS genom inloggning på NET-centralen.

ABCenix.
Operativsystemet ABCenix är utvecklat av Luxor Datorer och är kompatibelt med UNIX System III. UNIX har hög portabilitet dvs är enkelt att flytta mellan olika datorer. Dessutom är det ett väl beprövat operativsystem på väg att bli industristandard. En bra garanti för ett ständigt ökande utbud av UNIX-baserad programvara. Förutom alla delar som finns i UNIX, innehåller ABCenix en del förbättringar och nyheter t ex en liten, effektiv kärna för bättre prestanda, flera samtidiga filhanterare realtidsorientering, nya kommandon samt svenskt och engelsk mod.

Fönsterhantering.
Till ABCenix ingår också en fönsterhanterare, som implementeraras under operativsysteme. Den kan hantera upp till 16 fönster. Varje fönster är virtuellt, dvs en tittglugg in i den större skärmen. Dessutom kan varje fönster öppnas, tas bort, minskas, ökas eller flyttas. Teckenstorleken regleras automatiskt när fönstret varieras.



Relationsdatabas MIMER.
Den centrala momentet i alla datorbearbetning är hanterandet av data. Därför krävs en snabb och framförallt flexibel databashanterare i ett modernt datorsystem. Till ABC1600 har LUXOR valt MIMER, en avancerad relationsdatabashanterare. MIMER tillsammans med frågespråk, programgenerator, skärmhanterare och underhållsrutiner ger programmeraren ett "fjärde generationens" verktyg.

NECTAR
NECTAR är ett hjälpmedel för programutveckling, som gör att användare utan tidigare programmeringskunnskaper kan bygga egna program. Med NECTAR kan du koncentrera dig på applikationens yttre egenskaper, dvs vad du vill uppnå med programmet. Du beskriver i klartext menyer, formulär, tabeller och rapporter på skärmen och NECTAR översätter dem till programspråk. Med NECTAR klarar du de vanligast förekommande applikationerna, utan att behöva använda traditionell programmering. Skulle du köra fast i en applikation, som kräver traditionell programmering, är ändå inte ditt arbete ogjort. En programmeringskunnig kan ta vid där du slutade. Databashanteraren i NECTAR är MIMER.

Synkron och asynkron kommunikation.
ABC1600 följer upp ABC800-seriens breda kommunikationskoncept. T ex behövs ingen extra programvara för asynkron överföring. Den hanteras internt i ABC1600. För synkron kommunikation kan du välja mellan ett flertal protokoll t ex IBM 2780/3780, IBM 3770 SNA, UTS 400, IBM 3270 BCS och IBM 3270 SNA/SDLC.

Stor minneskapacitet.
Ju större arbetsminne i en dator desto större plats för och snabbare hantering av program och data. Arbetsminnet i ABC1600 är i standardutförandet hela 512 Kbytes och utbyggbart till 1 Mbyte. Externminnet är en 13 Mbytes winchester och en flexskiveenhet på 640 Kbytes. Båda dessa minnesenheter är inbyggda i datordelen. För att ytterligare öka snabbheten i hanteringen av data från externt anslutna minnen. har

DATORDEL
Dimensioner: 470*450*180 mm
Vikt: 17 kg
Processor:
Motorola 68008,16/32 bitars med klockfrekvens 8 MHz

Funktionsuppbyggnad:
- CPU - CIO - DART - SCC - DMA
- ROM/EPROM Bootstrap
- RAM (användarminne)
- Kommunikationsinterface (3 st)
- Interfacebuss (4 st)
- Flexskiveinterface

Minne:
Bootstrapminne ROM på 16 Kbytes. Arbetsminne RAM på 512 Kbytes utbyggbart till 1 Mbyte. Grafikminne RAM på 128 Kbyte utbyggbart till 512 Kbyte. Inbyggd flexskiveenhet på 640 Kbyte, 5 1/4" samt winchester på 13 Mbytes (formaterad)

In- och utmatning:
Fyra in- och utgångar. Två för kommunikationsmoduler V24 (RS 232). Dessa används för terminalanslutning. Överföringshastigheten 50-512 Kb/s asynkront och 50-512 Kb/s synkront. Modulerna är utbytbara till t ex V11 (RS422). En utgång används för skrivare, alternativt terminalanslutning. Överföringshastighet 50-19200 b/s. En utgång är för anslutning av tangentbord om systemet körs utan bildskärm.

Kalender: Batteridriven CMOS-kalender.
Nätspänning: 230/115 Volt (48/60 Hz).
Max uteffekt: 180 W.
Driftsäkerhet: 10000 timmar MTBF.
Säkerhet: S N D SI
Störsökerhet: VDE 0871 B.

BILDDEL
Dimensioner:
Porträtt 430*300*360 mm, landskap 390*370*360 mm.
Vikt: 15 kg,.

Bildrör:
Vridbart 90 grader och med ställbar lutning -5 till +15 grader. Den användbara bildytan är 210*280 mm. Bildrörtet är antireflexbehandlat, av forsfortyp (paperwhite) och 380 mm diagonalt. Röntgenstrålningen är mindre än från en normal TV, 0.5 mRem/h. Upplösningen är 1024*768 punkter och bildväxlingsfrekvensen är 60 Hz.

Nätspänning: 230/115 v (48/60 Hz).
Max uteffekt: 60 W.

TANGENTBORD
Funktionstangenter:
15 st programmerbara funktionstangenter med 60 olika koder. 8 stycken mjukvarustyrda indikeringslampor. Markörhanterare för styrning av markören i åtta (8) olika riktningar.

Alfanumerisk del:
SIS 662241.
Tangentkoder: Enligt svensk standard SES 850200 version 2.
Numerisk del:
Siffrorna 0-9, decimalpunkt, minustangent, backstegning hel rad samt RETURN-tangent.
Driftsäkerhet: 100.000.000 nedslag.
Programstyrda funktioner:
Ljudgenerator, klickljud, caps-lock och repetering vid tangentnedtryckning.
Anslutning: Anslutning av mus.

expansionsplatserna tillgång till direkt minnesaccess, DMA.

Grafikblock.
I standardutförandet är grafikblocket utrustat med 128 Kbytes RAM, expanderbart till 512 Kbytes. Grafikgeneratorn, som ingår i grafikblocket, understödjer mjukvarustyrt teckenformat och typsnitt samt automatiskt rullning av sida, soft scroll m m. Vid leverans medföljer ett antal lagrade standardtypsnitt (fonter). Du kan också skapa och lagra egna fonter.

Bildskärm.
Bildskärmen är vridbar till liggande eller stående A4, dvs landskaps- eller porträttformat. Texten är valfritt mörk mot ljus bakgrund eller ljus mot mörk bakgrund. Bildskärmen är pixelorienterad, dvs bilden byggs upp av individuellt tända och släckta punkter. Bildväxlingsfrekvensen är hög vilket ger en stabil och flimmerfri bild. Den höga upplösningen,1024*768 punkter, gör att kurvor, tecken mm blir lättavlästa.

Tangentbordet.
Tangentbordet ABC99 är av lågprofiltyp och ergonomiskt väl genomtänkt. För att användaren lättare skall kunna identifiera tangenterna, är 5:an försedd med en liten knopp och F och J extra skålade. Antalet funktionstangenter uppgår till 15 stycken och kan generera upp till sextio olika koder. Tangentbordet är också utrustat med en speciell markörhanterare. Med den kan du snabbt och enkelt flytta markören i åtta olika riktningar på skärmen.

Mus.
För att förenkla hanteringen av kommandon och data är ABC1600 utrustad med uttag för anslutning av mus, penna och digitaliseringsbord. Den mus som Luxor Datorer tillhandahåller är mekanisk, med tre olika funktionsknappar.

Kommunikationsanslutningar.
I ABC1600 finns två stycken utbytbara kommunikationsmoduler. I standardutföranden är det V24-moduler. Dessa kan bytas ut mot tex V11-moduler (RS 422).

Expansionsplatser.
För att ytterligare öka utbyggnadsmöjligheterna i ABC1600, finns en inbyggd expansionsplats, där du kan ansluta expansionskort av I/Otyp, fex kommmunikationskort, styr- och mätkort. Det finns plats för fyra kort, samtliga med tillgång till DMA. Vill du ansluta ytterligare en expansionslåda, använder du en av de ordinarie kortplatserna. Allt detta gör system ABC1600 synnerligen flexibelt. Det ger dej ett öppet koncept, där du fritt kan utforma systemets storlek och konfiguration.

\+ \+ \+

Genom den mjuk- och hårvara som Luxor har valt, har man valt en annan väg än många andra tillverkare. Luxor gör alltså inte någon IBM/PC-kompatibel mikrodator. Valet innebär att man kommer med en persondator som är mycket kraftfullare än IBM/PC G eller IBM/PC XT. Eventuellt kan IBM:s nya IBM/PC AT mäta sig med ABC1600.

Många andra, bland dem Ericsson, har valt att göra ganska enkla IBM/PC-kompatibla mikrodatoter. Ericsson och andra som gör IBM/PC-kompatibla datoter klarar sig på marknaden tack vare att man gör system som är något bättre än IBM:s.

I dag utvecklas mycket programvara till det operativsystem som IBM har valt, MSDOS. MSDOS har därmed blivit industristandard. Dock är MSDOS inget underverk, man kan t ex inte köra flera program samtidigt.

Luxor har valt ett operativsystem som är kompatibelt med UNIX System III. Även om Luxor kallar det för ABCenix, så kan man köra generell programvara avsedd för UNIX. Om man köper ett "programmerarpaket" till ABC1600 så får man tillgång till XENIX.

Luxor körde ett kalkylprogram på mässan som var skrivit för UNIX och PDP 11. Programmet hade Luxor bara fått i form av objektkod. Man kunde alltså köra programmet, utan att ens kompilera om det på ABC1600. IBM:s nya dator, IBM/PC AT skall också leveras med XENIX. IBM har dock ännu ej XENIX klart för leverans. XENIX är samma operativsystem som Luxor leverar till ABC1600!!!

I dag finns det alltså gott om programvara till MSDOS, men däremot mycket lite till XENIX och UNIX. Det faktum att IBM kommer att leverera XENIX till sin nya maskin gör dock att detta operativsystem kommer att vinna terräng.

Fördelen med UNIX kompatibla operativsystem är att operativsystemet är avancerat och detta gör bland annat att man kan köra flera program samtidigt. Nackdelen är att UNIX kärver en avancerad hårdvara och att det idag finns lite programvara till UNIX.

Eftersom Luxor räknar med att ABC806 och ABC802 skall finnas kvar på marknaden några år till så är ABC1600 ett bra alternativ tror jag. Att bara göra en IBM/PC-kompatibel dator skulle ge ett system som skulle rycka undan markanden för 800-serien. I stället så väljer man alltså ett större och mera avancerat system.

Luxor har, tror jag, tillräckligt stor marknad i Sverige för att man skall kunna klara sig utan att göra IBM-kopior. Någon har sagt att varför skall man göra kopior när man kan göra orginal! Vill man trots allt köra program för MSDOS så kan man ju köpa tillsatsen ABC16 till sin ABC800:a.

Mjukvaran:
Operativsystemet kallar Luxor för ABCenix. Det är samma sak som DIAB kallar för D-nix. Skillnaden skall bara vara att man kan köra ABCenix med "ÅÄÖ" och de UNIX funktioner som normalt ligger på dessa tecken ersätts av andra.

Vid leverans av systemet får man med ABCenix med de viktigaste hjälprutiner som normalt ingår i ett UNIX system samt en enkel skärmorienterad editorn, kallad SIV.

Vill man ha fler hjälprutiner, så får man köpa programmerpaket. Detta innehåller rutiner som gör ABCenix helt kompatibelt med XENIX. Detta gör också att man måste betala en dyr licens till XENIX:s upphovsman, Microsoft. Dessutom ingår programmeringsspråken C och Assembler i programmerarpaket. Priset för detta paket är ej offentliggjort när detta skrivs.

Själva operativsystemet tar upp cirka 120 Kbytes, filhanteraren 40 Kbytes. Skall man dessutom köra fönsterhanteraren så går det åt minst 60 Kbytes till. Då har man alltså redan förbrukat cirka hälften av de 512 Kbytes som finns vid leveransen. Detta innebär att det inte skadar om man expanderar minnet till 1 Mbyte. Detta kan för övrigt göras bara genom att trycka dit minneskapslarna (64 kbits kapslar).

Fönsterhanteraren:
Fönsterhanteraren är nog systemets pärla! Med den kan man hantera upp till 16 fönster samtidigt! I varje fönster kan man t ex köra ett program. Sedan väljer man vilka fönster som man vill se på skärmen. Om skärmen står i "porträttmode" så kan man se två fönster samtidigt.

Fönsterhanteraren påminner och Apples Lisa och Macintosh. Den stora skillnaden är att till ABC1600 kan man köra standard UNIX program och inte specialsydd programvara som till Apple-maskinerna. Fönsterhanterare kan jag inte beskriva i text, den måste upplevas med ögonen.

Man kan flytta cursorn med antingen en stor markörförflyttningstanget eller med en råtta som kopplas till tangentbordet.

Hårdvaran:
Processorn är Motorolas 68008. Denna har en 8 bitars databuss, vilket gör att man kan använda dagens diskar. Systemet hanterar alltså data med 8 bitar "i taget". Processorn kan adressera max 1 Mbyte minne linjärt. Utöver detta kan operativsystmet hantera virtuellt minne, dvs program kan vara större än vad som går in i minnet på en gång.

Utöver detta så är processorn jämförbar med vanliga 68000:an, såvitt jag förstår.

Vad skall man använda denna supermikro till?
Eftersom det när detta skrivs inte är klart vilka program som kommer att finnas till ABC1600 när den börjar säljas så kan man inte direkt säga när man skall satsa på ABC1600. Jag kan dock säga, att om man vill köra flera program samtidigt med hjälp av en fönsterhanterare, då kan ABC1600 vara ett alternativ. Jag är dock rädd för att ABC1600 systemet inte kan konkurera prismässigt med IBM/AT. Troligen är ABC1600 systemet bättre än IBM/AT, men trots allt så brukar inte tekniska prestanda sälja lika bra som förmånliga priser.

Luxor Datorer räknar med att sälja cirka 2000 ABC1600 system under 1985. Priset beräknades till 75 000 på mässan.

Bo Kullmar

ABC9000.

På mässan i Malmö visade Luxor sin största dator, ABC9000. ABC9000 är samma system som DS90. Den enda som Luxor har gjort med den är nog att klistra sitt varumärke på den! Enligt uppgfift kommer även Nokia att sälja den DS90.

Det är alltså frågan om samma system som jag har skrivit några rader om i ABC-bladet nr 3, 1984, sid 4 under rubriken "Från DIAB...".

ABC9000 använder samma operativsystem som ABC1600, nämligen ABCenix eller D-nix som DIAB kallar det. Systemet består av en centraldator, till vilken man kan ansluta terminaler.

Processorn är Motorola 68010 och minnet kan byggas ut till 8 Mbyte. Priset beräknas till 120 000 för grundenheten. För att få ett fungerade system med programvara för några terminaler tror jag att man kommer upp i en kostnad på cirka 200 000, om man inte räknar med kostnaden för terminaler. ABC806 kan t ex användas som terminal.

Skillnaden mellan ABC9000 och ABC1600 är att ABC1600 arbetar med en 8 bitars databuss och därmed vanliga diskar. ABC9000 har ingen grafikkonsol, dvs en huvdterminal till systemet med grafikegenskaper. Man kan däremot använda terminaler/mikrodatorer som grafikterminaler till ABC9000.

Bo Kullmar

ABC80

JOYSTICKINTERFACE

för ABC80 del 2

Hej igen. Den här gången skall jag visa hur man kan koppla ett tangentbord till datorn via joystick-interfacet och använda den som räknedosa.

Principen är densamma som för med en joystick förutom att man använder alla ingångarna och att programmet läser alla bitarna. Drivrutinen är nästan lika utom den lilla tabellen i slutet.

Se listning 1.

Tänk på att vi nu läser av alla knapparna. Om man trycker ned flera samtidigt, blir det inlästa värdet inte entydigt.

Det bifogade basicprogrammet är som synes mycket enkelt. Det klarar bara beräkningar med max två tal typ a + b = .

Se listning 2.

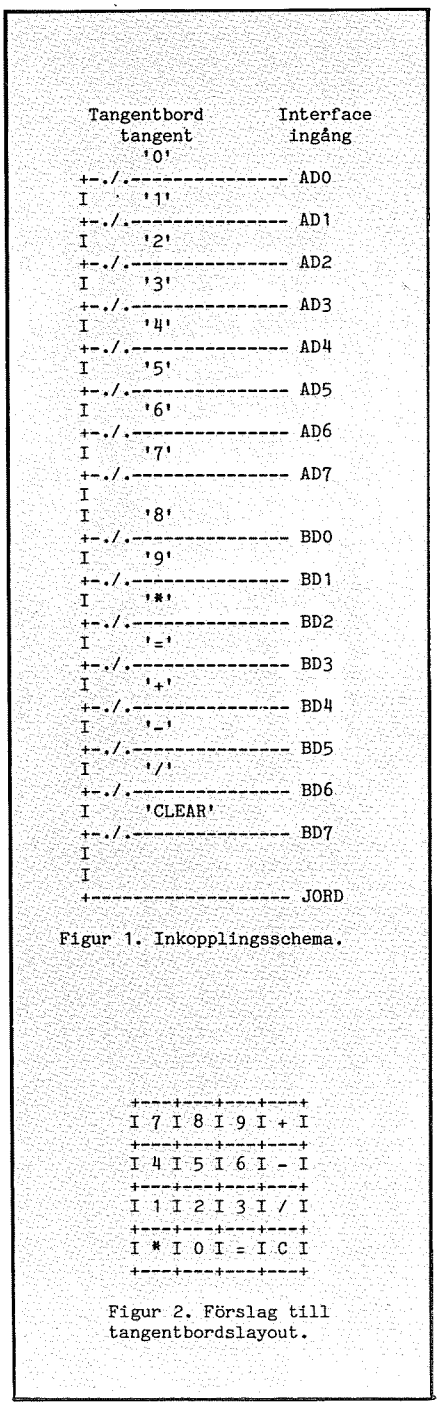
Värdet på zenerdioden Z i förra artikeln skall vara på 4.8 eller 5.1 V och motståndet R bör vara mellan 200 och 330 ohm 0.5 W.

Hur man kan använda joystickan till några av klubbens program återkommer jag till i nästa nummer av ABC-bladet.

Bo Hjulström <557>

Listning 1. POKE-satser.

```
1 REM  Drivrutin för interface via V24.
       För 16 knappars tangentbord.
2 POKE 64512%, 62%, 231%, 211%, 58%,
       62%, 255%, 211%, 58%, 17%, 0%,
       0%, 6%, 8%, 33%, 55%, 252%
3 POKE 64528%, 219%, 58%, 230%, 1%,
       32%, 3%, 126%, 179%, 95%, 35%,
       219%, 58%, 230%, 2%, 32%, 3%
4 POKE 64544%, 126%, 178%, 87%, 35%,
       62%, 247%, 211%, 58%, 62%, 255%,
       211%, 58%, 197%, 6%, 8%, 16%
5 POKE 64560%, 254%, 193%, 16%, 220%,
       98%, 107%, 201%, 1%, 1%, 2%,
       2%, 4%, 4%, 8%, 8%, 16%
6 POKE 64576%, 16%, 32%, 32%, 64%, 64%,
       128%, 128%
```

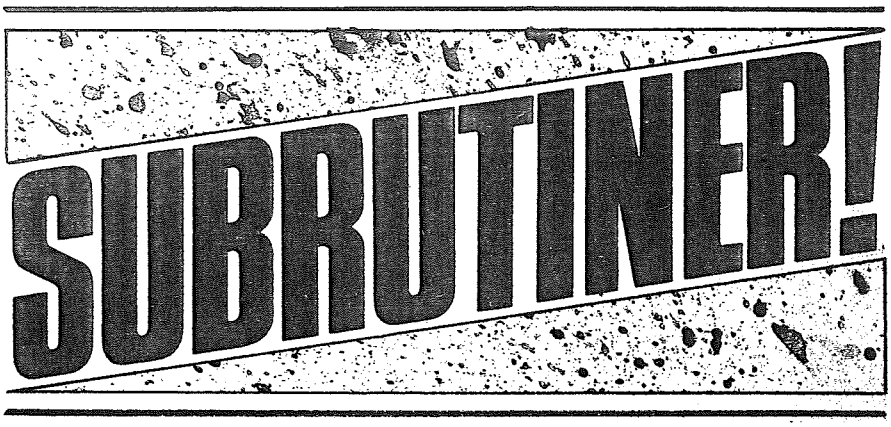


Figur 1. Inkopplingsschema.

Figur 2. Förslag till tangentbordslayout.

Listning 2. Programexempel.

```
 10 REM ----------------------------
 20 REM MINIRÄKNARE  MINIR.BAS
 22 REM (OBS MYCKET ENKEL.)
 30 REM 840326  BGH <557>
 40 REM ERFODRAR DRIVRUTIN I
 50 REM ASSEMBLER OCH INTERFACE
 60 REM FÖR TANGENTBORD VIA V24-
 70 REM KONTAKTEN.
 80 REM FÖR UPPLYSNINGAR KONTAKTA
 90 REM OVANSTÄENDE.
100 REM ----------------------------
102 S$='0' : S1$='0' : T=5
110 IF NOT PEEK(64512)=62% ; 'DRIVRUTIN
    SAKNAS.' : STOP
120 ; CHR$(12);'M I N I R Ä K N A
    R E !' : ; '>';
130 REM AVLÄSNING
140 Z%=CALL(64512%)
150 IF Z%=Z1% GOTO 140
160 Z1%=Z%
170 IF Z%=0% GOTO 140
180 REM OMKODNING
190 IF Z% AND 1% Z$='0'
200 IF Z% AND 2% Z$='1'
210 IF Z% AND 4% Z$='2'
220 IF Z% AND 8% Z$='3'
230 IF Z% AND 16% Z$='4'
240 IF Z% AND 32% Z$='5'
250 IF Z% AND 64% Z$='6'
260 IF Z% AND 128% Z$='7'
270 IF Z% AND 2^8% Z$='8'
280 IF Z% AND 2^9% Z$='9'
290 IF Z% AND 2^10% Z$='*'
300 IF Z% AND 2^11% Z$='§'
310 IF Z% AND 2^12% Z$='+'
320 IF Z% AND 2^13% Z$='-'
330 IF Z% AND 2^14% Z$='/'
340 REM IF Z% AND 2^15% Z$='.'
344 IF Z% AND 2^15% Z$='C'
350 REM KALKYLATORDEL
400 IF Z$>='0' AND Z$<='9' THEN S$=S$+Z$
    : ; Z$; : GOTO 130
410 IF Z$='' GOTO 130
440 IF Z$='+' THEN ; ' + '; : T=1 : S1$=
    S$ : S$='' : GOTO 130
460 IF Z$='-' THEN ; ' - '; : T=2 : S1$=
    S$ : S$='' : GOTO 130
480 IF Z$='*' THEN ; ' * '; : T=3 : S1$=
    S$ : S$='' : GOTO 130
500 IF Z$='/' THEN ; ' / '; : T=4 : S1$=
    S$ : S$='' : GOTO 130
520 IF Z$='C' THEN ; ' CLEAR ' : GOTO 700
560 GOTO 600
580 ; ' ERROR' : GOTO 700
600 IF NOT Z$='§' THEN GOTO 130
605 ; ' = ';
608 IF S$='' OR S1$='' GOTO 580
610 ON T GOTO 620,640,660,680,580
620 ; ADD$(S1$,S$,0) : GOTO 700
640 ; SUB$(S1$,S$,0) : GOTO 700
660 ; MUL$(S1$,S$,0) : GOTO 700
680 ; DIV$(S1$,S$,2) : GOTO 700
700 ; '>'; : S$='0' : S1$='0' : T=5 : GO
    TO 130
```



Subrutiner eller flerradiga funktioner.
I ABC-bladet nr 3,84 skrev jag och efterlyste subrutiner som medlemmar skrivit och var villiga att dela med sig av.
Nu har vi erhållit ett flertal bidrag. Vi tackar för dom.

Vi hoppas naturligtvis att flera skall ta sig tid att skicka in användbara subrutiner. Under tiden publicerar vi en del av det materiel vi erhållit.

Rune Mattsson <904>

Här är några filer som jag gärna vill dela med mig av.

ABC80

DOWNSCRL
En liten assemblerrutin som scrollar nedåt. Bra att ha ibland. Kanske någon vill göra en racertävling och utnyttja rutinen?

LEFTSCRL
Den här rutinen scrollar skärmen åt vänster. Detta kan man utnyttja när man plottar sinuskurvor t.ex.!

GRPHCHAR
En assemblerrutin som genererar en tabell över de grafiska tecknen. Den är kombinerad med en tangentbordsrutin som känner av CTRL-A. Om man håller på och gör program behöver man bara trycka CTRL-A så får man snabbt och bekvämt en lista.

STAND-BY
Med den här assemblerrutinen kan man tillfälligt "låsa" tangentbordet om man måste gå ifrån en stund t.ex. Under tiden kan ingen förstöra/ändra eller köra programmet man höll på med. Bara den som har rätt kod kan "låsa upp".

RUN-TIME
En assemblersnutt som förenklar inskrivningen av R-U-N.... Kan även användas som en sorts primitiv nödstopp om exempelvis printerutskrifter håller på att gå åt skogen...

FILE-NAME
En "ful" assemblersnutt som förenklar inskrivningen av OPEN "CAS:" ASFILE 1. Skönt för kasettfantomer som vill veta vad de har på bandet utan att radera.
De fyra filerna ovanför ska väl knappast användas var för sig. Om man med lite möda "kopplar ihop" dom till en liten hjälpare, får man nytta av dom tillsammans.

Filerna är gjorda och insändsa av
Johan Stuwe <3338>

DOWNSCRL

```
10 GOSUB 200 : REM Ladda maskinkod
20 REM RALLYTÄVLING
30 A%=10%
40 ; CHR$(12%)
50 IF A%<0% THEN A%=0%
60 IF A%>29% THEN A%=29%
70 ; CUR(0%,A%)"/          /"
90 FOR I%=1% TO 50% : NEXT I%
100 P%=CALL(65408%)
110 A1%=0% : IF RND<.5 THEN 130
120 IF RND<.5 THEN A1%=1% ELSE A1%=-1%
130 A%=A%+A1%
140 GOTO 50
200 REM DOWNSCROLL
210 REM Vid anrop scrollas skärmen ett
220 REM steg nedåt.
230 REM Anrop :  (Ex) P%=CALL(65408%)
240 REM Rutinen är relokerbar.
250 REM
260 DATA 245,197,213,229,221,229,221
261 DATA 33,160,3,62,24,1,40,0
262 DATA 221,110,0,221,102,1,61,40,14
263 DATA 221,94,2,221,86,3,237,176
264 DATA 221,43,221,43,24,230,33,0,124
265 DATA 17,1,124,1,39,0,54,32,237,176
266 DATA 221,225,225,209,193,241,201
270 RESTORE 260
280 FOR I%=65408% TO 65408%+57%
290 READ P% : POKE I%,P%
300 NEXT I%
310 RETURN
```

LEFTSCRL

```
10 GOSUB 200 : REM Ladda maskinkod
20 REM KURVPLOTTER
30 ; CHR$(12%)
40 FOR X=0 TO PI STEP .1
50 A%=11%-8%*SIN(2%*X)
60 ; CUR(A%,39%)".";
70 ; CUR(11%,39%)"-";
80 P%=CALL(65408%)
90 NEXT X
100 GOTO 50
200 REM LEFTSCROLL
210 REM Vid anrop scrollas skärmen ett
220 REM steg åt vänster.
230 REM Anrop :  (Ex) P%=CALL(65408%)
240 REM Rutinen är relokerbar men
250 REM gör ett CALL 2 : 147 till ROM.
260 REM Testat på checksumma 10042.
270 REM
280 DATA 245,197,213,175,50,244,253
281 DATA 50,243,253,245,205,147,2
282 DATA 6,39
283 DATA 19,26,27,18,19,16,249
284 DATA 62,32,18,241,60,254,24
285 DATA 32,231
286 DATA 209,193,241,201
290 RESTORE 280
300 FOR I%=65408% TO 65408%+35%
310 READ P% : POKE I%,P%
320 NEXT I%
330 RETURN
340 REM
350 REM Om följande ändringar görs,
360 REM så sparas kolumn 0 vid scroll.
370 REM Det är bra vid användning till-
380 REM sammans med grafik!
390 REM 282 DATA 6,38,19
400 REM 285 DATA 32,230
410 REM 300 FOR I%=65408% TO 65408%+36%
```

GRPHCHAR

```
200 REM GRAPHCHARACTER
210 REM Vid anrop genereras en ASCII-
220 REM tabell med de grafiska tecken.
230 REM Anrop :  P%=CALL(65408%) eller
240 REM om rutinen kombineras
250 REM med tangentbordsrutinen
260 REM på rad 370-490 så sker
270 REM anrop med CTRL-A.
280 REM Rutinen är relokerbar men gör
290 REM CALL 2 : 118,CALL 0 : 11 i ROM.
300 REM Testad på checksumma 10042.
310 REM
320 DATA 245,229,197,33,0,0,205,118,2
321 DATA 6,32,34,243,253,120,50,64,254
322 DATA 50,66,254,62,23,50,65,254
323 DATA 62,135,50,67,254,229,197
324 DATA 33,64,254,1,4,0,205,11,0
325 DATA 193,225,44,44,125,254,24
326 DATA 32,7,36,36,36,36,36,46,0
327 DATA 4,120,254,128,32,-53,33,22,0
328 DATA 34,243,253,193,225,241,201
330 RESTORE 320
340 FOR I%=65408% TO 65408%+73%
350 READ P% : POKE I%,P%
360 NEXT I%
370 REM
380 REM Tangentsbordsrutin som känner
390 REM av CTRL-A och gör isåfall hopp
400 REM till GRAPHCHARACTER ovan.
410 REM
420 I=250 : REM I-registret
421 REM Kolla takets höga byte :
422 IF PEEK(65064)=I THEN 430
424 REM Flytta taket och gör run igen :
425 POKE 65063,44,I : P%=CALL(3411%)
427 REM Nu pokas rutinen ovanför taket :
430 POKE I*256+45,245,62,I,237,71,241,201
440 POKE I*256+52,56,I,148,5
450 POKE I*256+56,245,219,56,254,129,40,4,
    241,195,30,3
460 POKE I*256+67,33,73,I,229,237,77,205
470 POKE I*256+74,65408%,SWAP%(65408%)
480 POKE I*256+76,195,219,0
490 P%=CALL(I*256+45) : REM Initiera!
500 REM
510 REM Eventuellt kan följande
520 REM ändringar göras i GRAPHCHAR. :
530 REM 325 DATA 193,225,44,125,254,12
540 REM 327 DATA 4,120,254,128,32,-52,33,11,0
550 REM 340 FOR I%=65408% TO 65408%+72%
```

STANDBY

```
10 REM Startprogram för
20 REM stand-by funktion.
30 ; CHR$(12%)
40 ; "Mata in önskad kod"
50 INPUTLINE A$
60 A$=LEFT$(A$,LEN(A$)-2%)
70 FOR I%=65408% TO 65407%+LEN(A$)
80 P%=ASC(MID$(A$,I%-65407%,1%))
90 POKE I%,P%
100 NEXT I%
110 GOSUB 200
120 POKE 64071%,LEN(A$)
130 END
200 REM STAND-BY
210 REM I bland har man nytta av att
220 REM kunna gå ifrån datorn medan
230 REM man håller på med ngt viktigt.
240 REM Då kan man med detta program
250 REM låta datorn stå "stand-by".
```

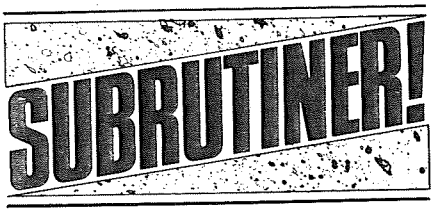
```
260 REM Nu har ingen möjlighet, att
270 REM använda/förstöra programmet.
280 REM Själv kan du trycka en kod
290 REM och på så sätt "låsa upp" !!
300 REM Kodsträngen lagras 65408-65535
310 REM med valfri längd. (Se nedan)
320 REM Rutinen är relokerbar men
330 REM pokas här ovanför taket.
340 REM Anrop :  CTRL-SHIFT-o
350 REM
360 DATA 245,219,56,254,159,194,31,3
370 DATA 243,229,197,33,128,255,6,5
380 DATA 219,56,254,128,48,-6
390 DATA 219,56,254,128,56,-6,230,127
400 DATA 190,32,-22,35,16,-20
410 DATA 193,225,241,251,237,77
420 REM Testa var taket finns :
430 IF PEEK(65064)=250 THEN 460
440 REM Flytta taket och gör "run"
450 POKE 65063,52,250 : P%=CALL(3411%)
460 REM Hoppvektorer :
470 POKE 64052,56,250 : REM Tangentbord
480 POKE 64054,148,5 : REM V24/kassett
490 FOR I%=64056% TO 64056%+41%
500 READ P% : POKE I%,P%
510 NEXT I%
520 REM Initiering :
530 POKE 64098,245,62,250,237,71,241,201
540 P%=CALL(64098%)
550 RETURN
560 REM Adressen och längden till
570 REM kodsträngen finns på rad 370.
580 REM Ändra raden så här om man
590 REM önskar adressen 55000 resp
600 REM längden 3 :
610 REM DATA 243,229,197,33,216,214,6,3
```

RUNTIME

```
200 REM RUN-TIME
210 REM En liten tangentbordsrutin
220 REM som om man trycker CTRL-R
230 REM startar BASIC-program,
240 REM snabbt och lätt!!!
250 REM Rutinen är relokerbar men
260 REM gör JP (!?) 13 : 80 i ROM.
270 REM Testad på checksumma 10042.
280 REM
290 REM Undersök var taket finns :
300 IF PEEK(65064%)=250% THEN 330
310 REM Flytta taket och gör "run"
320 POKE 65063,52,250 : P%=CALL(3411%)
330 REM Hoppvektorer :
340 POKE 64052,56,250 : REM Tangentbord
350 POKE 64054,148,5 : REM V24/kassett
360 REM Maskinkod :
370 POKE 64056,245,219,56,254,146,40,3
380 POKE 64063,195,31,3,33,81,250
390 POKE 64069,1,3,0,205,11,0
400 POKE 64075,33,80,13,229,237,77
410 POKE 64081,82,85,78
420 REM Initiering :
430 POKE 64084,245,62,250,237,71,241,201
440 P%=CALL(64084%)
```

FILENAME

```
200 REM FILE-NAME
210 REM En liten tangentbordsrutin
220 REM som om man trycker CTRL-F
230 REM startar bandspelaren och
240 REM skriver ut namnet på filen
250 REM den förhoppningsvis hittar.
260 REM Rutinen är relokerbar men
270 REM gör JP 13 : 144, JP 0 : 201 i ROM.
280 REM Testad på checksumma 10042.
290 REM
300 REM Undersök var taket finns :
310 IF PEEK(65064%)=250% THEN 340
320 REM Flytta taket och gör "run"
330 POKE 65063,52,250 : P%=CALL(3411%)
340 REM Hoppvektorer :
350 POKE 64052,56,250 : REM Tangentbord
360 POKE 64054,148,5 : REM V24/kassett
370 REM Maskinkod :
380 POKE 64056,245,219,56,254,134,40,3
390 POKE 64063,195,31,3,33,72,250
400 POKE 64069,229,237,77,33,106,250
410 POKE 64075,1,9,0,205,11,0,17,90,250
420 POKE 64084,205,144,13,195,201,0
430 POKE 64090,134,133,203,34,4,67,65
440 POKE 64097,83,58,187,184,199,1,0
450 POKE 64104,186,13
460 POKE 64106,83,69,65,82,67,72,73,78,71
470 REM Initiering :
480 POKE 64115,245,62,250,237,71,241,201
490 P%=CALL(64115%)
```



SUBRUTIN INPUT

Hur långt sträcker sig upphovsrätt till programrader? Precis som när det gäller schack torde man kunna hävda att det inte finns några hemligheter i allmänt publicerade programspråk som BASIC. Allting finns där till beskådande. Men olika människor är olika snillrika när det gäller att sätta ihop delarna.

Det kan också hända att olika människor kommer på ungefär samma sammansättning, helt enkelt därför att den är logisk. Kan man då kräva upphovsrätt eller copyright eller vad det kan heta?

Nedanstående subrutin gör ungefär detsamma som ABC-datorernas normala inmatningsrutin, men den gör det öppet så att man programmässigt kan ingripa, t ex för att räkna radlängden och säga ifrån när den blir för stor (ex rad 2080).

Visserligen har jag hederligt och med en del besvär skrivit den själv, men det kan inte vara så stora avvikelser hos de versioner som finns på annat håll.

Alla får ju också ide'er av varandra, och det kan vara svårt att i efterhand reda ut vem som påverkade vem eller vem som sa det först.

Det är knepigt för den som vill hävda upphovrätt. För små programsnuttar av subrutintyp torde det vara ganska omöjligt. Å andra sidan är det väl knappast sådana småjobb som någon programmakare lever på? Det vore intressant med en diskussion om den saken i bladets spalter. ---

Nu de vanliga orden på vägen.

Huvudslingan ligger i 1000-2000 och utformas litet olika beroende på aktuell tillämpning. I stället för INPUTLINE I$ skriver man alltså GOSUB 2000 och får då "svaret" i strängen I$.

Här låter jag programmet skriva I$ på rad 1015 för kontroll. Därför syns det skrivna två gånger - en gång under själva skapelseprocessen och en gång när det hela är färdigt. Därigenom kan jag också pröva om vänster- och högerpil fungerar som de skall.

Subrutinen som sådan börjar med rad 2005 som spar en eventuell gammal rad (t ex vid editering i ett färdigt register) i J$ och nollställer målsträngen I$.

På 2010 kommer så tangenttrycket, som analyseras i 2020. Här har man möjlighet att bestämma åtgärder vid icke skrivbara tecken (t ex CTRL-A som är ascii 1, CTRL-L ascii 12). I den här subrutinen ignoreras alla sådana ascii-koder utom 8 (vänsterpil), 9 (högerpil) och 13 (RETURN).

Om man har slagit in ett skrivbart tecken hamnar man direkt på 2060, där tecknet skrivs på skärmen och lagras upp i I$, varpå man börjar en ny runda på 2090. (Dessemellan har vi alltså en inlagd radkontroll.)

Så fortsätter man tecken efter tecken tills raden avslutas med RETURN. Därvid går programmet ut i rad 2050, gör radframmatning och (i det här fallet) skriver facitraden igen. I andra tillämpningar gör man något annat med I$ på 1015, t ex skriver på en fil.

Pilarna är litet svårare. Om man vill backa (och rätta) ett tecken med vänsterpilen, går programmet in i rad 2030. Då kontrolleras först att det finns något skrivet som man kan backa in i (längden av I$ får inte vara noll). Sedan skall markören backas, ;CHR$(8%);.

Vill man nu sudda på skärmen måste man skriva över det icke önskade tecknet med blanktecknet (ascii=32) och sedan backa på nytt.

Nu gäller det att ta bort sista tecknet i I$, vilket sker i slutet av 2030.

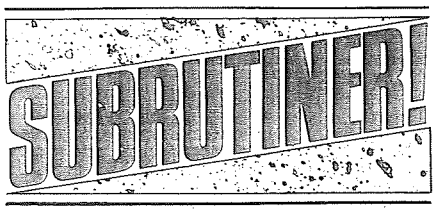
Om det finns något upplagrat som man vill stega fram med högerpilen händer det på 2040. Först kontrolleras att det finns något i J$ sedan hämtas första tecknet i J$, skrivs ut på skärmen och läggs till I$. Därefter tas det bort från J$.

Ganska bökigt, eller hur? Och mycket lätt att glömma något semikolon eller annan obetydlighet, vilket vanligen leder till något mystiskt fel i utskriften.

Någon skarpsynt iakttagare upptäcker säkert att I$=I$+W1$ också förekommer på rad 2060 fast i annan gestalt. En riktig smarting kan troligen komma på ett sätt att bara använda rutinen en gång. Jag har ett gammalt (numera inte riktigt klart) minne av att det kunde bli trassel i vissa situationer med osynliga tecken som adderades till I$. Därför höll jag fast vid villkoret W%>31% i rad 2060. Och därför måste jag tyvärr upprepa mig i 2040.

Sven Wickberg

```
100 REM INPUT
110 REM Subrutin för inmatning tecken f
    ör tecken
120 REM Bra när man vill räkna radlängd
    en
130 REM eller göra editeringar i redan
    befintliga rader
140 REM ------------------------------
    ----
1000 REM
1010 GOSUB 2000
1015 ; I$
1020 GOTO 1010
1030 REM ----------------------------
2000 REM
2005 J$=I$ : I$=''
2010 GET W$
2020 W%=ASC(W$)
2030 IF W%=8% IF LEN(I$) THEN ; CHR$(8%,
     32%,8%); : I$=LEFT$(I$,LEN(I$)-1%)
2040 IF W%=9% IF LEN(J$) THEN W1$=LEFT$(
     J$,1%) : ; W1$; : I$=I$+W1$ : J$=RI
     GHT$(J$,2%)
2050 IF W%=13% THEN ; : GOTO 2500
2060 IF W%>31% ; W$; : I$=I$+CHR$(W%)
2070 REM
2080 L%=5% : IF LEN(I$)>L% ; CHR$(7%); :
     REM pling om rad>L%
2090 GOTO 2010
2490 REM
2500 RETURN
```



SUBRUTIN MENY

Menytekniken är välkänd och populär i den meningen att den ofta används. Men det finns många (och krångliga) modeller, där var och en har sin egen fason.

Detta kapitel utreds mycket förtjänstfullt och noggrant i ABC om programmering och dokumentation, men jag visar här en enkel subrutin som är universellt användbar.

Jag har varit i den situationen att behöva många olika menyer i likartade sammanhang. I den här rutinen behöver man bara ändra DATA-satserna (och möjligen rubriken), så har man sin meny klar.

Grundprogrammet finns i slingan 100-150. Datasatserna läses i tur och ordning. Första trängen ger filnamnet, andra den förklaring som skall visas i menyn.

Denna meny har numerisk pekare, dvs man väljer ett tal för att få det program man vill ha. Rad 140 är ett vidlyftigt sätt att placera menyn en liten bit under rubriken MENY överst på sidan.

När datasatserna tar slut fungerar 110 och skickar programmet till rad 200 och på liknande sätt frågar efter menyval.

210-220 innehåller kontroll av att användaren väljer en siffra inom den rätta ramen.

290 är avslutningsraden om man väljer SLUT. Jag har valt denna utformning med tanke på att alla menyer innehåller rutinen SLUT, men att denna rutin kan komma på ganska olika ställen. Denna avslutningsrad är dock generell om motsvarande datasats är "SLUT".

Raden slutar med CHAIN"" vilket ju betyder "sudda minnet och hämta inget mer program". Man kunde lika gärna skriva END. Om END tilläggs på raden kommer minnet att suddas. Det visade sig vara bra att menyn fanns kvar, så det fick stå som det står med pip och allt. (STOP hade varit mera logiskt förstås).

Denna meny är bra när man skall välja program - särskilt om programnamnen är långa och/eller krångliga typ SYS:PROTFIL.-UTL där en ovan användare lätt skriver fel.

Somliga människor väljer hellre en bokstav än en siffra: L för lista, N för nästa och S för sluta. Dem kan man tillgodose genom att ta med valtecknet i datasatserna, exempelvis:

```
890 DATA B, LIB, 'Se BIBLIOTEKET'
900 DATA S, SKRIV, 'SKRIVA text'
....
```

Då måste man också ändra:

```
130 Read V$, A$(I%), A$
140 ; .. V$, A$(I%), A$
```

Då kan man inte heller ha den enkla anvisningen om val i rad 200 eller rimlighetskontrollen i 210-220. Men naturligtvis kan man fixa något i stil med

```
145 U$=U$+V$ : REM valmöjligheterna
...
210 W%=INSTR(1%,U$,W$)
220 IF W%=0% ;CHR$(7%) : GOTO 200
...
```

och menyn är fortfarande generell. Man kan få problem med att variera bokstäverna - hur skilja mellan Skriva och Sluta? Då får man ge fantasin spelrum. Kanske kan man följa högt föredöme och använda punkt (".") för Sluta. Allting ordnar sig.

Men hur gör man om man inte bara skall välja ett program utan en rad parametrar, som kanske också skall åtgärdas innan programmet löper vidare?

Därom mera i en annan artikel.

Sven Wickberg

```
 10 REM MENY            84 03 12 SW
100 ; CHR$(12%)'/// MENY ///'
110 ONERRORGOTO 200
120 FOR I%=1% TO 10%
130 READ A$(I%),A$
140 ; CUR(2%*I%+1%,10%)I%'  'A$
150 NEXT I%
190 REM ------------------------------
200 ; CUR(2%*I%+2%,0%)'Vad vill du göra
    ( 1 -'I%-1%'):'; : INPUT W$
210 ONERRORGOTO 200 : W%=VAL(W$)
220 IF W%<1% OR W%>I%-1% THEN ; CHR$(7%
    ) : GOTO 200
290 IF A$(W%)='SLUT' THEN ; CHR$(12%)'S
    LUT FÖR DENNA GÅNG' : CHAIN ''
300 ; : ; 'Vi hämtar 'A$(W%) : CHAIN A$
    (W%)
890 DATA LIB,'Se BIBLIOTEKET'
900 DATA SKRIV,'SKRIVA text'
910 DATA LÄS,'LÄSA textfiler'
920 DATA RÄKNA,'Öva huvudräkning'
940 DATA SLUT,SLUTA
```

BREV:

Hej alla glada ABC-tomtar!

En av de saker man ibland saknar på ABC80 är COMMON-variabler, dvs variabler som inte går förlorade vid CHAIN. Det går ju förstås att skriva med POKE i POKE-arean i adress 65408 till 65535, men det blir gärna lite väl arbetsamt. Det sätt som jag har upptäckt ger en sträng innehållande 128 tecken som gemensam variabel. Skriv i början av de program som du gör CHAIN emellan:

```
DIM A$=128% : POKE PEEK(65065)+
SWAP%(PEEK(65066))+6%,128%,255%,128%,
0%
```

A$ är hör COMMON-variabel. Det som händer är att pekaren till variabelvärdet ändras till 65408. Dessutom ändras strängen längd till 128 tecken,detta för att man inte kan veta hur lång den var innan man gjorde CHAIN. A$ kan ändras till vilken annan strängvariabel som helst utan att något mer behöver göras. Programraden verkar på första variabeln i variabellistan, inte på namn.

Så har jag ett förslag, eller rättare sagt en fundering grundad på det som stod på omslaget till ABC-kassett nummer 11 om belöning och val av bästa program för året. Varför inte kombinera detta med inbetalning av medlemsavgiften? På inbetalningskortet somm alla får, anges vilka kasetter som ingår i årets omröstning,och där skriver man (frivilligt) vilket programman röstar på, samtidigt som man betalar in sin medlemsavgift. Man noterar rösterna, och får sedan med datorns hjälp lätt ut årets populäraste program. Lämpligen röstas i flera klasser, ex.vis spel,utilityprogram, tekniskmatematiska program och administrativa program. Författarna belönas, och får i gengäld författa en artikel i tidningen där han/hon dissekerar och kommenterar sitt program och visar på nyttiga finesser mm.

Denna "tävling" kommer ochså ofelbart att peka ut vad folk anser vara bra med ett program. Är det nyttighet, dokumentation eller idiotsäkerhet som är viktigast? När man väl vet svaret på detta, kan man skaka fram artiklar i bladet som lär oss medlemmar hur man förbättrar sina program på de viktiga punkterna. Och alla blir glada och nöjda

hoppas

Isak Engquist <375>
Riksdalergatan 40
582 66 LINDKÖPING

Kommentar till insändare från Isak Enquist:

En del av ABC-Klubbens administration sköts numera på kommersiell basis av Gustavii Bokföringsbyrå (Berit Gustavii). Berit ser bland annat till så att uppgifter om inbetalda medlemsavgifter noteras i medlemsregistert.

Detta gör att jag, tyvärr, anser att vi inte kan låta Berit ta sin värdefulla arbetstid i anspråk med att registera tävlingsvar. Dessutom är inbetalningskorten bokföringsverifikat.

I och för sig så skulle man kunna använda den kupong, som följer med inbetalningskorten och på den markera sina svar. Finns det bara någon som kan ta hand om jobbet så tycker jag mycket väl att man skulle kunna ha en sådan omröstning. Jag tycker kanske att man inte skall blanda in avien för medlemsavgift alls, eftersom det på kortet som man skickar tillbaka kan finnas andra uppgifter.

Jag menar att om vi får in ett par hundra lappar och en del av dessa innehåller andra uppgifter, t ex om att någon inte vill ha direktreklam, så måste någon stå och kopiera dessa lappar så att informationen når rätt personer. Därför tycker jag det är enklare med lappar/kort för bara själva omröstningen.

Bo Kullmar

Bevingade ord av kända och okända programmerare

1 Konstigt.
2 Det känner jag inte till.
3 Det har ju fungerat förr.
4 Det är bara några småsaker som skall fixas.
5 Hur kunde det bli så?
6 Det måste vara något hårdvarufel.
7 Dom har kanske bytt release.
8 Ni måste har gjort något fel.
9 Ni måste har något fel i materialet.
10 Men jag har inte ändrat något i den modulen.
11 Jodå, det blir klart innan dess.
12 Ni måste ha fått tag i någon gammal version.
13 Det är bara ett skönhetsfel.
14 Jag är nästan klar.
15 Jodå, bara jag får införa dom sista ändringarna.
16 Det tar ingen tid.
17 Det beror antagligen bara på en olycklig slump.
18 Man kan inte testa allting.
19 Det här kan inte påverka det där.
20 Men jag trodde att jag hade fixat det.
21 Det är med, det är bara inte testat.
22 Egentligen fungerar det bra, fast det inte verkar så.
23 Frånsett att det inte fungerar, hur tycker Ni att det verkar.

Av okända författare
insänt av Bo Kullmar <1789>

Produkter som moderniserar din ABC-dator:

OBS! Inga smartaid:ar inkräktar på datorns arbetsminne.

smartaid III — Programmeringshjälp för ABC80.
Sveriges mest imiterade smartaid?

super smartaid — Programmeringshjälp för ABC80.
Har definierbara tangenter, textfils-editering, JOB-strömmar, batteriuppbackat CMOS-minne, printer-rutin m.m.

smartaid 800 — Äntligen ett kraftfullt programmerings-hjälpmedel för ABC800 seriens datorer.
Skärmeditor, KEY, CMOS-minne m.m.

startaid — Autostart för ABC80. Startar program från kassett, floppy el. PROM.

timeaid 80
timeaid 800 — Realtidsklocka med batteri drift. Alarm-funktion, tider från 0.0001 s till 128 år, I/O-portar, inbyggd timer/counter m.m.

De flesta produkterna kan anslutas tillsammans med CAT-net.

Klipp här ✂ Bättre brödlös än ramlös, använd smartaid.

Härmed beställes följande:

smartaid III	st.	á 1049.00 kr	endast datablad ☐
super smartaid	st.	á 1985.00 kr	endast datablad ☐
smartaid 800	st.	á 2850.00 kr	endast datablad ☐
startaid	st	á 1049.00 kr	endast datablad ☐
timeaid 80	st.	á 2850 00 kr	endast datablad ☐
timeaid 800	st.	á 2850.00 kr	endast datablad ☐

Extra manual till super smartaid 50 kr/styck. [st.]

Varorna sändes mot postförskott. Pris inkluderar moms men ej postförskottsavgift

Namn: ..... Tel .....
Adress : .....
Postnr: ..... Ort: .....

ABC-klubbens

MONITOR.

Telefonnummer

Televerket har meddelat att vi inte kan få ett gruppnummer förrän tidigast i maj. När vi får ett gruppnummer så kommer vi att behålla 08-80 15 23 och där ha en telefonsvarare med information om monitorn, bl a driftströnignar. Då har vi även möjlighet att tala om vilket gruppnummer vi har fått. I samband med gruppnummret införs så upphör 80 15 26 och 26 91 86.

När detta skrivs gäller förljande telefonnummer:

08-80 15 23 300/300 bps V21
08-80 15 26 300/300 bps V21
08-80 17 25 300/300 bps V21
08-26 91 86 1200/75 bps V23
08-80 11 55 1200/1200 bps V22

1200/1200 linjen är dock f n ej tillgänglig när systemarbete sker i klubblokalen. Det sker dock inte så ofta.

På 1200/75 linjen kommer Lars-Göran Göransson att prova fram en bra lösning för att köra såväl 1200/75 som 300/300. Sen skall man kunna köra både 300/300 och 1200/75 på alla linjer utom 1200/1200 linjen. När detta kan vara klart, vet jag inte.

Göteborgsmonitorn.

Göteborgsmonitorn är nu igång på 031-13 98 35, 300 bps halv duplex. Den är öppen för alla medlemmar. Ange medlemsnummer och lösenord för att logga in. Det skall vara det lösenord som stod på 1984 års medlemskort och inte det som du kan ha ändrat på nya monitorn i Stockholm.

Systemet består av den gamla monitorn från Stockholm. Kommandon är ändrade och anpassade till den nya monitorn.

Turbo CAT-NET

Vi har köpt ett 128 Kbytes extra minneskort till CAT-NET:et. Efter att vi hade installerat detta så gick systemet **mycket snabbare.** Kortet arbetar som buffert mot disken och det innebär att systemet läser ett helt spår från disken upp i RAM-bufferten. På kortet sitter det en lysdiod som blinkar när kortet accessas. Då ser man verkligen att winchestern får jobba mindre.

CAT-NET - LUX-NET

Luxor och DIAB ser gärna att vi går över från CAT-NET till LUX-NET. För att göra detta möjligt har man skänkt oss ett LUX-NET och tre LUX-NET noder samt en ABC802. Luxor lovar också att vi kan få fler noder och eventuellt lite annan utrustning. LUX-NET kan vara bättre än CAT-NET för denna typ av system. ABC-Klubben har ännu inte beslutat att gå över till LUX-NET, men vi tänker konvertra programvaran.

Observera att jag inte påstår att LUX-NET generellt är bättre än CAT-NET, jag hänvisar till en separat artikel om nät.

För att gå över till LUX-NET så behöver vi en ny winchester. Den Mica-winchester som vi har nu kan bara användas till CAT-NET. En övergång till LUX-NET förutsätter sålunda att vi kan sälja nuvarande CAT-NET utrustning och köpa en ny winchester med rabatt.

En effekt av detta kan tyvärr bli att inkoppling av ytterligare noder kan försenas.

Man kan konstatera att Luxor/DIAB är mycket mera angelägen om att vi skall använda LUX-NET än vad CAT är för att vi skall använda CAT-NET. Från början fick vi CAT-NET på långtidslån, men blev sedan tvingade att köpa det. Av Luxor/Diab fick vi däremot direkt ett LUX-NET som gåva.

Nya kommandon i Monitorn

WRITE har införts så att man kan skriva in en textfil i inlådan. Om man bara skriver WRITE <filnamn> så får man max 157 tecken per rad. Man kan minska detta ner till 40 tecken per rad genom att skriva WRITE <filnman>,<tkn per rad>. Då får kan man skiva in fler rader. Detta beror på att texten mellanlagras i MEM:.

Detta gör också att det har visat sig gå bra att dumpa in filer dirket i systemet med detta kommando. Enligt Robert Svedjehammar så fungerar det till och med i 1200/1200 bps.

Man kan editera texten med samma kommanodon som i MSG.

En märkning av filer håller på att införas, vilket innebär att man kommer att kunna ta bort främst de filer som man själv har skickat in med KILL <filnamn>.

Programbanken

Programbanken har nu tillförts en hel del filer. I tidningen publiserar vi en lista över vilka filer som finns. Denna lista är samma som filen ALFALIB.TXT som finns på masternivån, dvs där du hamnar när du har loggat in.

Programbanken är organiserad i fler bibliotek. Man kan se biblioteken som ett träd, med en rot, grenar och löv. För att hämta program måste du normalt ta dig ut till ett löv. Detta sker med DIR-kommandot. Du kan också ange sk switchar till DIR-kommandot.

Vill du gå till biblioteket för ABC80/spel och samtidigt se vilket bibliotek som du har hamnat i så skriver du DIR,SL ABC80/SPEL. Då får du reda på vilket bibliotek du befinner dig i och ev underbibliotek. Vill du ta dig till biblioteket för utility program för ABC80, så skriver du DIR,FS UTILITY. Genom F(ader) tar du dej upp en nivå och sedan ner till U(tility). S är i båda fallen default.

Vill du tillbaka till översta nivån så använder du switchen M(aster), genom att skriva DIR,M.

När du sedan skall skiva LIB så kan du göra det med switchar också. Vill du se filerna sorterade på datum skriver du LIB, ORD. O står för ordning, R för reverse (baklänges) och D för datum. Är du bara intresserad av vissa filer så kan du skriva t ex LIB,ORD ABC* eller LIB,ORD *.BAS.

Du kan numera få ut storlekar också genom switchen S. Använd dock inte detta för ofta, för det tar mycket resurser från systemet. Gör bara LIB med storlek på en fil som du vill hämta. Då skriver du LIB,S FIL.BAS.

I programbanken finns det en del BAC-filer. Det är främst ACB80 BAC-filer som inte kan läggas upp i BAS-format. Vi hoppas så småningom få igång en KERMIT så att även icke BAC-filer kan hämtas. Observera att för att hämta filer med GET eller skicka in filer med SEND måste man använda ett speciellt BASIC-program typ FILTRANS. Kom ihåg att filer som skall skickas in med SEND då måste läggas upp i BAS-format först.

En ny medlem har gjort några ABC800-program för att omvandla ABS-filer till textfilen i HEX. Med utgångspunkt från dessa program så skall jag lägga upp ABS-filerna konverterade till textfiler (i HEX). Någon får också göra ett program som fungerar i ABC80 för att konvertera tillbaka filerna till ABS-format.

MSG-systemet

Flera tycks har provat på systemet nu. I flera möten finns det en hel del inlägg. Du som inte har kört systemet tidigare kan rimligtvis inte hinna läsa alla dessa. Därför bör du använda "endast"-kommandot. Genom att skriva "endast 10" så får du läsa de 10 sista inläggen i ett möte.

"Hoppa"-kommandot använder du när du vill hoppa över nästa inlägg som systemet vill att du skall läsa. Eftersom man inte kan skriva endast 0, så kan du skriva "endast 1" och sedan hoppa. Vilket får effekten att systemet anser att du har läst alla inlägg.

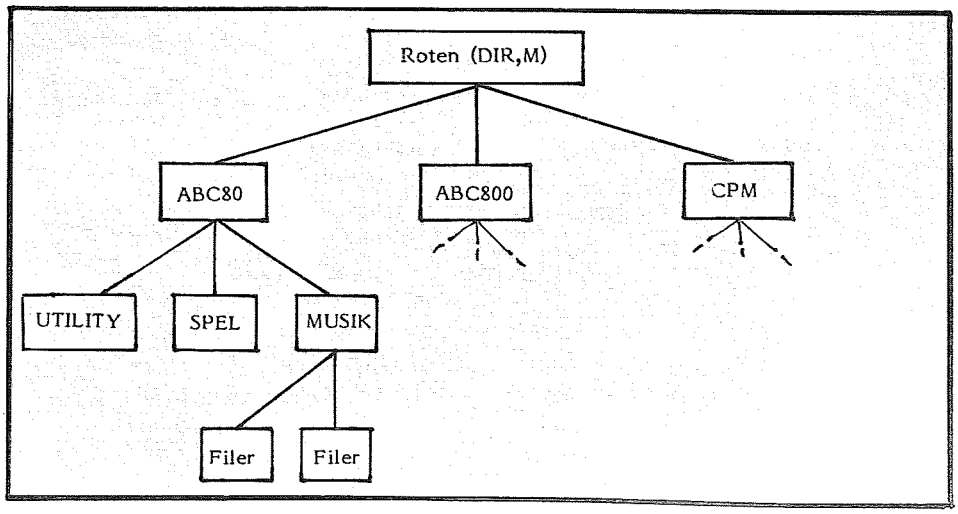
Använd också "skumma"-kommandot. Skriver du "skumma 5" så får du läsa de 5 första raderna. Sedan kan du skriva "resten" om du vill läsa resten eller "hela" om du vill läsa hela. Detta går mycket snabbt eftersom texten mellanlagras i RAM-minne. Du kan på samma sätt skriva "igen" när du har läst ett inlägg och får då se det igen.

"Prioritera" använder du för att ordna mötena i den ordnig som du själv väljer. Det möte som du prioriterar hamnar först. Skall du ändra ordnignen på alla möten så får du alltså börja med det som skall komma sist.

Glöm inte "utträda"-kommandot. Utträd ur de möten som du inte vill var med i. Du kan gå med i ett möte, vara med ett tag och sedan utträda om du vill. Du kan naturligtvis också utträda ur medforum, om du vill.

När du sparar brev så kan alla läsa sparade brev. Vi kommer senare att ändra detta och i samband med-detta görs så att du kan ta bort dina brev och filer som du skickar in till inlådan med KILL.

Bo Kullmar <1789>



NÄTVERK.

ABC, CAT och LUX-NET

Jag skall försöka att beskriva dessa nät lite. Jag har dock bara praktisk erfarenhet av CAT-NET så det jag skriver om de andra näten får du ta med en nypa salt.

ABC-NET

Detta nätverk är mycket dåligt, det erkänner till och med Luxor. Överföringshastigheten är otroligt låg, 19200 bps tror jag. Man kan dock fördubbla hastigheten, kontakta Benny Löfgren, DIAB, för information om detta.

Vad gäller hastighet på ett nät så brukar det inte vara kritiskt, eftersom det vid stor belastning brukar vara winchesterdiskens förmåga att betjäna flera användare samtidigt som begränsar systemet. I ABC-NET är det dock den mycket långsamma överföringshastigheten som begränsar effekiviteten. Som jämförelse kan jag nämna att Överföringshastigheten för en direktansluten floppy ligger mellan 250 000 och 500 000 bps.

Biblioteken hanteras på samma sätt som när man kör vanligt UFD-DOS, dvs med UFD. För att byta UFD måste man köra programmet UFD. Man kan inte kopiera mellan två underbibliotek, enbart mellan ett UFD och ett MFD, masterbibliotek. Inga speciella säkerhetsrutiner finns i ABC-NET.

De som har ABC-NET bör om möjligt byta det mot CAT eller LUX-NET.

CAT-NET

CAT-NET är troligen det nätverk till ABC maskiner som har flest installationer, över 600. CAT-NET är väl beprövat och har funnits på marknaden i cirka fem år. Överföringshastigheten är 153 600 bps, alltså betydligt snabbare än ABC-NET. CAT använde poll select, dvs centralen pollar varje nod och frågar om den har något att sända till centralen. Det innebär att all kommunikation måste gå via centralen.

CAT-NET ansluts till en ABC dator via nodkort, en typ finns för ABC80 och en för ABC800. CAT-NET har en helt egen filstruktur, så om man kör en central floppy i CAT-NET så kan man inte läsa den i en vanlig ABC-maskin. Däremot kan man köra lokal floppy för kopiering. Det är lite krångligt, men går till så att man har det vanliga ABC-DOS:et kvar och sedan laddar men CAT-NET DOS:et i RAM. Därefter kan man köra ett speciellt kopieringsprogram för att kopiera till och från CAT-NET.

I CAT-NET:centralen kan extraminne installeras som fungerar som buffert för disken. Eftersom ABC-Klubben har köpt ett sådant 128 Kbytes kort så vet jag att ett extra minneskort ger en slags Turbo-effekt. Svarstiderna för discaccess minskar en hel del. Detta är en mycket stor fördel, att det finns ett sådant kort. Kort finns för 128 Kbyte och 512 Kbyte. Drivrutin för central RAM-floppy håller på att tas fram.

På minneskortet finns det en lysdiod som talar om när kortet accessas. Det finns 9 bitar per byte, så man har även en paritetsbit. Sker paritetsfel i minnet så sker automatisk omstart av nätet. Man måste köra med version 1.5 av nätet för att använda kortet och då känner systemt automatsikt av hur mycket minne det finns i centralen.

För att förflytta sig mellan olika bibliotek i CAT-NET så skriver man sin biblioteksspecifikation i variablen Q9$. Q8$ kan också användas, men gäller då bara en discaccess. Man kan kopiera mellan alla bibliotek.

Det finns flera olika typer av filskydd i CAT-NET. Vill man göra allt så måste man vara SYSMAN. Först blir man falsk SYSMAN och för att bli äkta måste man köra programmet REAL.

Winchesterdiskar från Mica, Luxor, Tranfor och Ardo kan anslutas. En stor fördel med CAT-NET är ett program, REMOTE, till ABC80/ABC800 som gör att man kan köra en ABC-maskin i nätet från en terminal. Då kan man köra de flesta program, dock inte program som skriver direkt i bildminnet. PermoBAS körs fn. med en sådan REMOTE rutin.

Eftersom klubben har fått ett LUX-NET så kan det hända att vi går över till det. Anledningen till det är dels att klubben inte har något emot att få utrustning gratis. Från CAT har vi "bara" fått rabatt. Trots att vi blev lovade att få nätet på långtidslån så blev vi tvungna att köpa det! Klubbens monitorsystem ställer speciella krav på nätet, som normala användare inte har. Detta gör att LUX-NET kan vara bättre.

CAT-NET har en ganska ineffektiv hantering av filnamnen. Har man ett bibliotek med 1000 filer så går det trögt att få tag på en fil i ett sådant bibliotek. Klubben har haft många filer, men vi har måst dela upp det i flera bibliotek. Å andra sidan så kan man bara ha 256 filer i ett LUX-NET bibliotek (underbibliotek).

CAT-NET har ingen processor på nodkortet. Det innebär att datorn sköter jobbet när centralen pollar noden, vilket gör att om man slår hårt på tangentbordet när man laddar in en fil så dyker överföringen från nätet till datorn. Detta är inget man gör i ett vanligt system, men i klubbens monitorsystem så kan till och med störningar på telefonlinjen orsaka fel som gör att monitorsystemet dyker. I och för sig så kan man med 800-serien hantera dessa fel med ON ERROR GOTO, men det fungerar naturligtvis inte vid CHAIN. LUX-NET har en processor på nodkortet och därför uppstår inte detta problem.

Vill man köra ABC80 och eventuellt andra maskiner som IBM PC så bör man välja CAT-NET. ABC80 kan nämligen inte köras på LUX-NET, eftersom Luxor inte vill betala utvecklingen av ett DOS för ABC80 och LUX-NET. En del program måste anpassas till nätet för att kunna köras. Sådana program finns anpassade till CAT-NET. Kolla dock först att just ditt program finns till CAT-NET. CPM kan också köras på ABC-maskiner, men hur detta fungerar känner jag inte till.

Central ISAM finns, den är dock inte helt kompatibel med Luxors. CAT-NET är numera lagervara hos Luxor. Dessutom ingår CAT-NET i Luxors nya skolpaket tillsammans med ABC806.

Fördelen med CAT-NET är att man kan köra andra än ABC-maskiner mot nätet. Systemet är väl beprövat och buggfritt. Extra minneskort till centralen är en mycket stor fördel. Nätet är hyfsat snabbt och man kan även köra ABC80 mot nätet. REMOTE-program finns. Systemprogrammen levereras osquezade och därför kan man ändra dem om man vill.

Nackdelen är att kommandon för att byta bibliotek är lite konstiga och att processor saknas på nodkoret. Man får också säga att det är en nackdel att det inte är tillverkaren som står bakom systemet, detta på grund av att supporten från en tillverkare ofta kan bli bättre än från en "pirat".

LUX-NET

LUX-NET är mycket nytt. Det började att levereras till kund i december 1984. LUX bygger synkron HDLC överföring. På nätet sitter en linjeanpassningsenhet (modem) som ger RS422 till nodkoret i ABC:en. Överföringshastigheten är 500 000 bps. En sändning på nätet initieras av en nod och därför måste systemet hantera kollisioner på nätet. Detta är alltså en annan princip än den som CAT-NET använder.

Man kan i LUX-NET använda en kassettbandspelare samtidigt med att filaccess sker mot nätet. Detta går inte i ABC och CAT-NET.

LUX är kompatibelt med det vanliga skivformatet i ABC-maskiner. Man har ett nätdos som ligger mappat ovanför det vanliga doset. Det innebär att man utan problem kan köra en lokal floppydisk samtidigt som man kör mot LUX-NET. Man kan dock inte samtidigt köra central och lokal ISAM.

En rutin för utökad buffert mot disk kommer till LUX-NET. Därvid skall Luxors standard 128 Kbytes kort användas. Om det kommer att bli möjligt att köra en central RAM-floppy i LUX-NET, vet jag inte.

Kommandon för att förflytta sig mellan bibliotek här heter CD och är inlagt i nätdos:et. Man skriver CD bib1/bib2 om man vill till bib2 som då är ett underbibliotek till bib1. FILESTAT kommandot finns också för att förändra status på filer. Man kan ha max 256 filer i ett bibliotek. CAT-NET klarar fler, men det blir mycket trögt då.

Clusterstorleken för winchestrar som körs under ett UFD-DOS är normalt 32. Detta kan minskas genom att LUX-NET länkar bitkartorna. En praktisk gräns är dock 4. Som jämförelse kan jag nämna att begreppet clusterstorlek inte finns i CAT-NET. Där tar en fil alltid upp det utrymme som filen upptar fysiskt.

Länkar till filer kan läggas upp, så att en fil bara utgör en pekare till en annan fil. Det innebär att storleken på filen inte kan läggas upp i biblioteket.

En svaghet med LUX-NET är att det inte finns något REMOTE program likande det som finns till CAT-NET.

ABC80 kan inte anslutas, eftersom Luxor inte vill satsa pengar på att ta fram ett DOS som kan användas till ABC80. Eventuellt så kan DIAB dock ta fram någon lösning som kan göra det möjligt att använda ABC80 på LUX-NET. Annars går det ju att köra ABC80 på CAT-NET!

Vad är då fördelen med LUX-NET. Hårdvaran är nog bättre än CAT, men också dyrare. LUX är kompatibelt med det vanliga skivformatet, man kan enkelt läsa en ABC-floppy i en central LUX-floppy enhet. Kommandon mot nätet verkar vara enklare än CAT:s Q$ variabler. En fördel är också att Luxor/DIAB står bakom LUX-NET, vilket gör att supporten kan bli stark.

Nackdelen är priset, att REMOTE-rutin saknas och att det kan bli svårt att köra ABC80 på LUX-NET. En nackdel är också att det är nytt, all programvara är nog inte färdig och buggar finns säkert.

Bo Kullmar

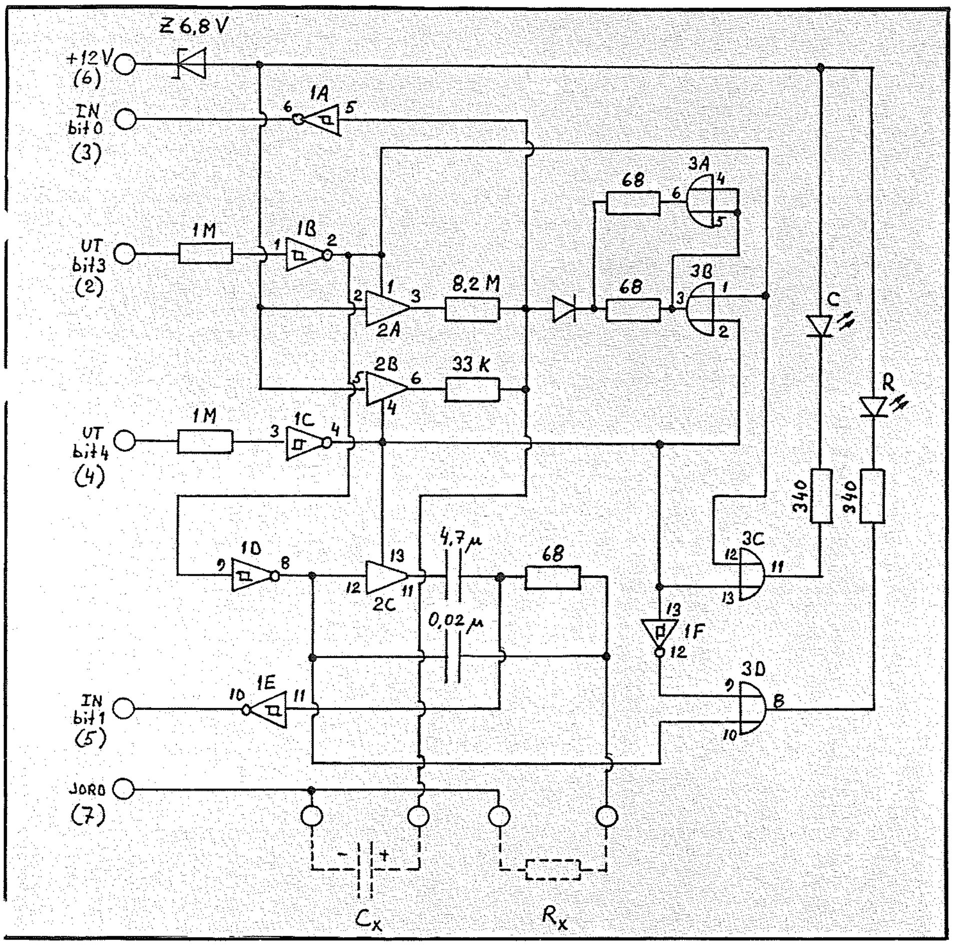
DATORN SOM MÄTINSTRUMENT.

ABC80

Alla som är intresserade av att bygga egna tillsatser till datorn eller av elektronikbyggen i största allmänhet har väl en mer eller mindre väl sorterad samling komponenter hemma. Halvledarna brukar vara hyggligt märkta med beteckningar i klartext och motstånden har ju sin färgkodning. Men kondensatorerna.... Det brukar bara leda till frågetecken. En kapacitansmeter är naturligtvis lösningen men den ryms inte alls i amatörens budget.

Hav emellertid tröst, man kan ju använda sin ABC80 till det med hjälp av några extra komponenter för en billig penning.

Detta behövs :
40106 Hex Schmitt trigger/inverter
74LS126 Quad 3-state buffer
7432 Quad 2 input OR



Dessutom några motstånd, två lysdioder, två kondensatorer,en zenerdiod och en vanlig kiseldiod. Värdena på motstånden och kondingarna är inte kritiska, man tager vad man haver, trimmningen sker med mjukvaran. Allt detta löds ihop enligt schemat på en experimentplatta eller som skatbo beroende på vilken metod man brukar använda.

Alla anslutningar till datorn sker via V24-kontakten så en sladd med kontakt måste man också ha. Dessutom något sätt att ansluta den kondensator eller det motstånd man vill mäta upp. Ja, en resistansmeter får vi så att säga på köpet med de kretsar vi har.

Så till hur det hela fungerar. Grundprincipen är den att en okänd kondensator laddas upp genom ett känt motstånd. Spänningen över kondensatorn stiger enligt en exponentialfunktion. En Schmitt-trigger känner av spänningen och slår om då den nått en viss nivå. Datorn mäter tiden fram till omslaget och ur denna kan kapacitansen beräknas. Det hela sköts av en maskinspråksrutinen här bredvid.

Programmet försöker först ladda genom 8,2 Mohm-motståndet och skulle det ta för lång tid, dvs kapacitansen är stor, görs laddning genom 33 kohm. Två uppladdningar med mellanliggande urladdning görs för att öka noggrannheten i mätningen. Urladdningen sker genom de två 68 ohms motstånden och dioden. Två lysdioder indikerar att mätning pågår och vilken typ av mätning. Zenerdioden på 6,8 volt (400 mW) är till för att ta ned 12-voltspänningen i V24 utgången till 5 volt TTL-nivå men det går ju att använda bara CMOS så att 12 volt kan användas.

I basicprogrammet sker omräkning till lämpliga sorter (pF,nF,uF) samt "skalning" i olika steg på rad 630-700. Storleken på dessa konstanter kan behöva ändras beroende på vilka komponenter man har använt i sin uppkoppling. Detta görs lämpligen genom att köra på några kondensatorer med kända värden och små (10%) toleranser och sedan justera konstanterna i programmet tills det visar rätt.

Ett annat sätt vore att i programmet med hjälp av e-funktionen för en kondensators uppladdning beräkna värdet.

Mätning av resistans sker efter samma principer fast "tvärtom", om Ni förstår vad jag menar.

Ulf Johnsen <674>

```
0000                        ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0000                        ;
0000                        ;                 KAPMÄT
0000                        ;
0000                        ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0000                        ;
0000                        ;Program som mäter upp
0000                        ;en okänd kondensator
0000                        ;medelst 2 st upp- och
0000                        ;urladdningar och
0000                        ;presenterar 2:a värdet
0000                        ;Två mätområden används
0000                        ;och rätt område
0000                        ;väljs automatiskt.
0000                        ;
0000                        ;I adress 65500 ligger
0000                        ;valt område och
0000                        ;eventuell felkod
0000                        ;Mätresultatet ligger
0000                        ;i HL-registret och
0000                        ;tas med till BASIC
0000                        ;
EA60   START  ORG  60000    ;Här lägger vi rutinen
EA60   IOPORT EQU  58       ;I/O-portens nummer (V24)
EA60   FLAGGA EQU  65500    ;Flagga för använt område
EA60                        ;och fel
EA60                        ;
EA60                        ;<< HUVUDPROGRAM >>
EA60                        ;
```

```
100 REM ****************************************
110 REM *  RCMÄT.BAC                            *
120 REM *  Version 1.2 / 81-10-24 / 84-10-10    *
130 REM *                                       *
140 REM *  Gjort och insänt av                  *
150 REM *  Ulf Johnsen  < 674 >                 *
160 REM ****************************************
170 REM Programmet mäter upp en kondensator eller
180 REM ett motstånd med hjälp av litet extra
190 REM hårdvara och presenterar kapacitans resp.
200 REM resistans på skärmen. Kapaciteten måste
210 REM vara mellan 15pF och 50 uF. Resistensen
220 REM måste vara mellan 50 ohm och 50 Mohm.
230 REM ========================================
240 REM Här läggs maskinspråksrutinerna in i
250 REM adresserna 60000-60209
260 REM ========================================
270 POKE -5536%,217%,30%,2%,217%,205%,176%,234%,
        62%,8%,211%,58%,219%,58%,203%,71%
280 POKE -5521%,32%,7%,3%,120%,177%,40%,15%,24%,
        243%,217%,29%,217%,32%,231%,197%
290 POKE -5506%,225%,62%,1%,50%,220%,255%,201%,
        217%,30%,2%,217%,205%,176%,234%,62%
300 POKE -5491%,16%,211%,58%,219%,58%,203%,71%,
        32%,7%,3%,120%,177%,40%,15%,24%
310 POKE -5476%,243%,217%,29%,217%,32%,231%,197%,
        225%,62%,2%,50%,220%,255%,201%,62%
320 POKE -5461%,9%,50%,220%,255%,201%,175%,211%,
        58%,17%,0%,0%,27%,122%,179%,32%
```

```
EA60 D9              EXX
EA61 1E02            LD   E,2         ;Antal mätningar i E'
EA63 D9              EXX
EA64 CDB0EA    START1 CALL NOLL
EA67 3E08            LD   A,00001000B;Bit 3 hög, område 1
EA69 D33A            OUT  (IOPORT),A
EA6B DB3A      FULL1? IN   A,(IOPORT) ;Uppladdad ?
EA6D CB47            BIT  0,A
EA6F 2007            JR   NZ,KLAR1   ;Hoppa då ur
EA71 03              INC  BC         ;annars Öka räknare
EA72 78              LD   A,B
EA73 B1              OR   C          ;Räknaren full ? (=0)
EA74 280F            JR   Z,OMRÅD2   ;Välj då område 2
EA76 18F3            JR   FULL1?     ;annars kolla igen
EA78 D9        KLAR1 EXX
EA79 1D              DEC  E          ;Skippa första försöket
EA7A D9              EXX
EA7B 20E7            JR   NZ,START1
EA7D C5              PUSH BC         ;Lägg resultatet i HL för BASIC
EA7E E1              POP  HL
EA7F 3E01            LD   A,1
EA81 32DCFF          LD   (FLAGGA),A ;Markera område
EA84 C9              RET             ;Till BASIC
EA85                                 ;
EA85 D9        OMRÅD2 EXX
EA86 1E02            LD   E,2        ;Antal försök i E'
EA88 D9              EXX
EA89 CDB0EA    START2 CALL NOLL
EA8C 3E10            LD   A,00010000B;Bit 4 hög, område 2
EA8E D33A            OUT  (IOPORT),A
EA90 DB3A      FULL2? IN   A,(IOPORT) ;Uppladdad ?
EA92 CB47            BIT  0,A
EA94 2007            JR   NZ,KLAR2   ;Hoppa då ur
EA96 03              INC  BC         ;annars Öka räknare
EA97 78              LD   A,B        ;Räknaren full ? (=0)
EA98 B1              OR   C
EA99 280F            JR   Z,OWFL
EA9B 18F3            JR   FULL2?     ;Kolla igen
EA9D D9        KLAR2 EXX
EA9E 1D              DEC  E          ;Skippa första försöket
EA9F D9              EXX
EAA0 20E7            JR   NZ,START2
EAA2 C5              PUSH BC         ;Lägg resultatet i HL för BASIC
EAA3 E1              POP  HL
EAA4 3E02            LD   A,2
EAA6 32DCFF          LD   (FLAGGA),A ;Markera område
EAA9 C9              RET             ;Till BASIC
EAAA 3E09      OWFL  LD   A,9
EAAC 32DCFF          LD   (FLAGGA),A ;Markera overflow
EAAF C9              RET             ;Till BASIC
EAB0                                 ;
EAB0                                 ;>>>>>> NOLLNINGSRUTIN <<<<<<
EAB0                                 ;
EAB0 AF        NOLL  XOR  A          ;Rensa A
EAB1 D33A            OUT  (IOPORT),A ;Ger urladdning
EAB3 110000          LD   DE,0
EAB6 1B        LOOP1 DEC  DE         ;2 st loopar ger
EAB7 7A              LD   A,D        ;tillsammans 1,13 s
EAB8 B3              OR   E          ;fördröjning
EAB9 20FB            JR   NZ,LOOP1
EABB 1B        LOOP2 DEC  DE
EABC 7A              LD   A,D
EABD B3              OR   E
EABE 20FB            JR   NZ,LOOP2
EAC0 010000          LD   BC,0       ;Nolla räknaren
EAC3 C9              RET             ;Till huvudprogram
EAC4                                 ;
EAC4           LÄNGD EQU  $-START    ;Beräkna programmets längd
EAC4                 FIL  KAPMÄT     ;Spara objektkoden på fil
EAC4           END   END


Pass 2 ready.
Errors ->> 0



Labels :

START = EA60 IOPORT= 003A FLAGGA= FFDC START1= EA64
FULL1?= EA6B KLAR1 = EA78 OMRÅD2= EA85 START2= EA89
FULL2?= EA90 KLAR2 = EA9D OWFL  = EAAA NOLL   = EAB0
LOOP1 = EAB6 LOOP2 = EABB LÄNGD = 0064 END    = EAC4
```

```
330 POKE -5446%,251%,27%,122%,179%,32%,251%,1%,0%,
    0%,201%
340 POKE -5436%,217%,30%,2%,217%,205%,28%,235%,
    62%,16%,211%,58%,253%,43%,253%,35%
350 POKE -5421%,219%,58%,203%,79%,40%,7%,3%,120%,
    177%,40%,15%,24%,243%,217%,29%
360 POKE -5406%,217%,32%,227%,197%,225%,62%,3%,50%,
    220%,255%,201%,217%,30%,2%,217%
370 POKE -5391%,205%,28%,235%,62%,0%,211%,58%,253%,
    43%,253%,35%,219%,58%,203%,79%
380 POKE -5376%,40%,7%,3%,120%,177%,40%,15%,24%,
    243%,217%,29%,217%,32%,227%,197%
390 POKE -5361%,225%,62%,4%,50%,220%,255%,201%,62%,
    8%,50%,220%,255%,201%,62%,24%
400 POKE -5346%,211%,58%,17%,0%,0%,27%,122%,179%,
    32%,251%,27%,122%,179%,32%,251%
410 POKE -5331%,1%,0%,0%,201%
420 REM ========================================
430 K1$=' Kapacitansen är' : R1$=' Resistansen är'
440 ; INP(3)CHR$(12) : F$=CHR$(127,127,127,127)
450 ; CHR$(12); : ; '<<<< R-C-MÄTNING >>>>' : ; : ;
460 OUT 58,16 : ; : ; : ; : ; 'Mätning av
    resistans (R)'
470 ; '           kapacitans (C)'
480 ; : ; 'Avsluta ( RET )                ';
    GET R$ : ; CHR$(12)
490 IF R$='R' OR R$='r' THEN 770 ELSE IF R$='C' OR
    R$='c' THEN 520
500 IF R$=CHR$(13) THEN ; CHR$(12) : END ELSE
    GOTO 460
510 REM ========================================
520 ; '<<<<< Uppmätning av kondensator >>>>>'
530 ; : ; 'Anslut kondensatorn och tryck'
540 ; 'sedan på en tangent !' : GET R$
550 ; : C=CALL(60000) : IF C<0 THEN C=65536+C
560 IF C<12 THEN 600
570 IF PEEK(65500)=1 THEN 630
580 IF PEEK(65500)=2 THEN 670
590 IF PEEK(65500)=9 THEN 730
600 ; : ; F$;K1$;' mindre än 15 pF'
610 ; 'För liten för att mätas här' : ; : ; : ;
620 GOTO 460
630 IF C<25 THEN C=1.3*C : GOTO 650
640 IF C<150 THEN C=2*C : GOTO 650 ELSE C=2.6*C
650 IF C>10000 THEN C=C/1000 : ; : ; F$;K1$;C' nF'
    ELSE ; : ; F$;K1$;C' pF'
660 GOTO 460
670 IF C<400 THEN C=875*C : GOTO 710
680 IF C<3000 THEN C=825*C : GOTO 710
690 IF C<15000 THEN C=770*C : GOTO 710
700 IF C<25000 THEN C=725*C : GOTO 710 ELSE
    C=625*C
710 C=C/1E+6 : ; : ; F$;K1$;C' uF'
720 GOTO 460
730 ; : ; F$;K1$;' större än 50 uF'
740 ; 'För stor för att mätas här' : ; : ;
750 GOTO 460
760 REM ========================================
770 ; : ; '<<<<< Uppmätning av resistans >>>>>'
780 ; : ; 'Anslut motståndet och tryck'
790 ; 'sedan på en tangent ! ' : GET R$
800 ; : C=CALL(60100) : OUT 58,16 : IF C<0 THEN
    C=65536+C
810 IF C=0 THEN 850
820 IF PEEK(65500)=3 THEN 880
830 IF PEEK(65500)=4 THEN 940
840 IF PEEK(65500)=8 THEN 1000
850 ; : ; F$;R1$;' mindre än 50 ohm'
860 ; 'För liten för att mätas här' : ; : ;
870 GOTO 460
880 IF C<22 THEN C=7.5*C : GOTO 920
890 IF C<1200 THEN C=6.5*C : GOTO 920
900 IF C<4000 THEN C=6*C : GOTO 920
910 IF C<30000 THEN C=5.35*C : GOTO 920 ELSE C=5.7*C
920 IF C>999 THEN C=C/1000 : ; : ; F$;R1$;C' Kohm'
    ELSE ; : ; F$;R1$;C' ohm'
930 GOTO 460
940 IF C>27000 THEN 1000
950 IF C<9000 THEN C=1100*C : GOTO 980
960 IF C<12000 THEN C=1260*C : GOTO 980
970 IF C<16000 THEN C=1480*C : GOTO 980 ELSE
    C=1740*C
980 IF C>1E+6 THEN C=C/1E+6 : ; : ; F$;R1$;C' Mohm'
    ELSE C=C/1000 : ; : ; F$;R1$;C' Kohm'
990 GOTO 460
1000 ; : ; F$;R1$;' större än 50 Mohm'
1010 ; 'För stor för att mätas här'
1020 GOTO 460
```

Vem behöver egentligen IBMs standard?

ABC 1600 tillhör den nya generationens smådatorer. En supermikro på en sådan teknisk nivå att den redan idag väl motsvarar nästa generations krav.

Den är för det första utrustad med ABCenix, ett DIAB-utvecklat operativsystem. En industristandard helt kompatibel med UNIX* System III och med System V på systemanropsnivå. ABCenix uppfyller alla krav på UNIX-standard, satta av den internationella användargruppen för UNIX (UNIX user/group).

ABCenix-maskin och samtidigt skicka över jobb via LUX-NET till en annan ledig ABCenix-maskin för bearbetning och därefter hämta tillbaka resultatet. Du kan även emulera ett stort antal olika terminaltyper, tex Tektronix 4014 och VT 100.

För det sjätte har vi utvecklat Basic II till Basic III. En ännu kraftfullare Basic med direktanrop till databashanterarna ISAM och MIMER. Dessutom finns högnivåspråken C, Assembler, Fortran 77, Pascal, Cobol och APL.

För det sjunde kan du ansluta Data-Board-seriens kort för olika styr-, mät-,

Alla känner vi till begreppet IBM-standard. En rörelse som samlar många och hängivna bekännare. Idag formligen översvämmas smådatormarknaden av IBM-kopior. Alla med i stort sett samma egenskaper, alla intill förvillelse lika.
Det finns också en annan rörelse på marknaden. En standard som formas av den nya tidens krav på smådatorer. Vi kan kalla det en "världsstandard" som dikteras av självklara användarkrav. Företag som AT&T, SPERRY, NCR, Burroughs och Motorola har redan anslutit sig.
Denna nya standard baserar sig på en unik öppenhet och samverkan mellan bl a operativsystem, processorteknologi, datahantering och grafisk presentation. En inre datorergonomi, så genomtänkt att den ger förutsättningar för ett helt nytt smådatortänkande. Vi skall förklara vad vi menar. Låt oss presentera ABC 1600. Vår senaste smådator, utvecklad efter den nya "världsstandarden."

För det andra har ABC 1600 GKS, en internationell standard för grafisk presentation.

För det tredje är ABC 1600 först i världen med relationsdatabashanterare (MIMER) som standard på smådatornivå.

För det fjärde är ABC 1600 utrustad med en kraftfull fönsterhanterare som saknar motstycke på marknaden. Den arbetar med upp till 16 fönster och kan köras mot vilken System III- eller V-kompatibel standardprogramvara som helst. Inga program behöver vara skrivna för fönsterhantering. Dom behöver inte ens veta att dom körs i fönster.

För det femte är ABC 1600 utrustad med ett av marknadens starkaste kommunikationskoncept. Du kan kommunicera med de flesta stordatorer tex IBM, DEC och UNIVAC. Med LUX-NET kan du koppla ihop upp till 32 arbetsplatser och tex arbeta i en regler- och kommunikationslösningar.

Flertalet av ovanstående punkter ligger till grund för den "världsstandard", som håller på att utvecklas. De är direkt avgörande för en smådators möjligheter i framtiden. Därför kan vi, till skillnad från IBM och deras kopior, erbjuda självklarheter som tex kraftfull kommunikation, avancerad grafik, portabilitet och stor frihet i val av programvara.

Det är bara att konstatera att IBM-standarden inte räcker till när det är marknadens krav och inte en enskild tillverkares idéer som styr utvecklingen. Ingen insatt användare kommer att nöja sig med mindre än den standard som inbjuder till hög teknik, fri konkurrens och öppenhet.

Precis det som ABC 1600 erbjuder redan idag.

*UNIX is a trademark of AT&T Bell Labs. XENIX is a trademark of Microsoft.

TEKNISKA DATA.

Dator ABC 1600.
Motorola 68008
0,5–1 Mbyte arbetsminne.
128–512 Kbyte grafikminne.
Memory Access Control (MAC).
Kommunikationsenhet:
- V 24 (RS 232C).
- V 11 (RS 422) (tillbehör).
- asynkron/synkron (NRZI).

13 Mbyte Winchester (formaterad), utbyggbar i grundenheter >50 Mbyte i separat enhet och med ytterligare expansionsmöjligheter.
640 Kbyte flexskiveenhet, även för 8" i separat enhet.
Expansionsmöjligheter med Data-Board-serien, för mät-, styr- och regler-funktioner.

Bildskärm ABC 1615.
Vridbar till stående (porträtt) eller liggande (landskap) format.
Högupplösning 1024x768 pixels.

Tangentbord ABC 99.
Markörplacerare (8 olika riktningar).
15 programmerbara funktionstangenter.
Uttag för mus.

Operativsystem.
ABCenix.
Fönsterhanterare.

Databashanterare.
MIMER DB (relationsdatabas).
MIMER QL (frågespråk).
MIMER IR (informationssökning).
MIMER PG (programgenerator).
MIMER SH (skärmhanterare).
ISAM.

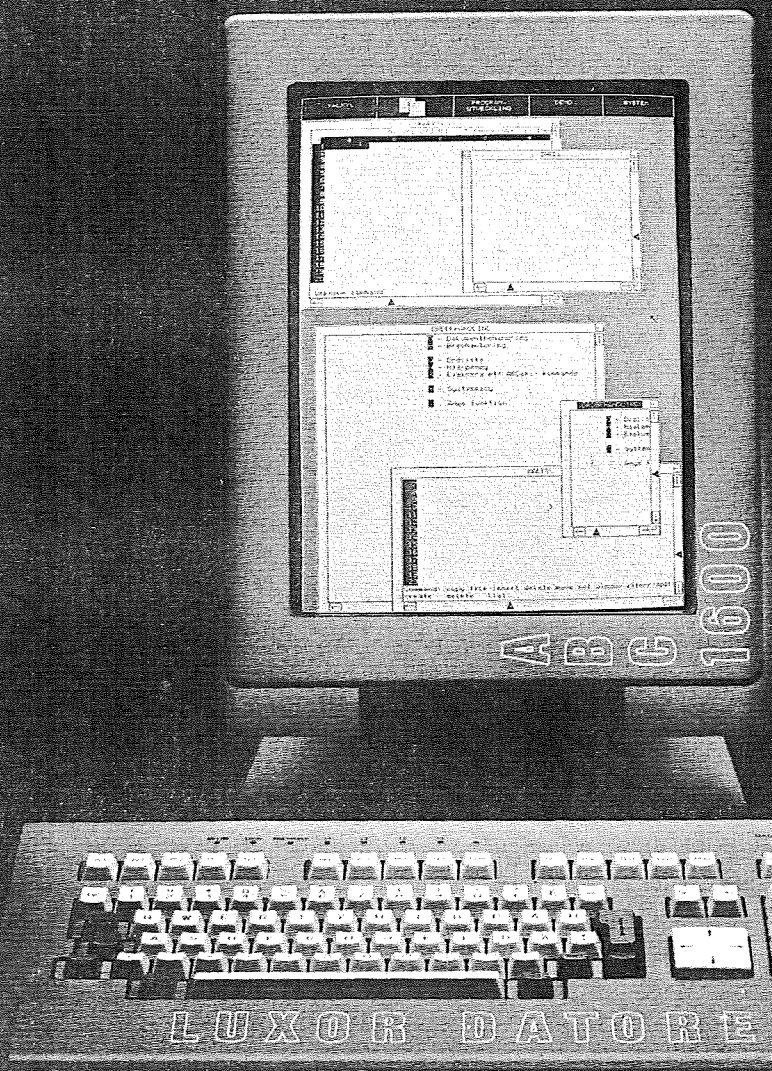
Utvecklingsverktyg.
XENIX* utvecklingspaket med C, Assembler, Länkare och ett 80-tal utilities.
Basic III med direktanrop till ISAM och MIMER.
ABC-filhanterare, för enkel överföring av datafiler skapade under ABC-DOS.
SIV, EMACS (fullskärmseditorer).
LEX-68 ordbehandlingsprogram.
NECTAR (4:e generationens språk och applikationsverktyg).
GKS (Graphical Kernel Systems).
Fortran 77 Pascal, Cobol, APL.

Kommunikation.
Nätverk: LUX-NET.
Synkron: IBM 2780/3780, IBM 3770 SNA/RJE, IBM 3270 BSC, IBM 3270 SNA/SDLC, UTS 400.
Asynkron: ingår som standard i ABCenix.
Terminalemuleringar: VT100, Tektronix 4014 m fl.

LUXOR DATOREX

Fönsterhanteraren i ABC 1600 arbetar med upp till 16 fönster och kan köras mot vilken System III- eller V-kompatibel standardprogramvara som helst. Inga program behöver vara skrivna för fönsterhantering. Dom behöver inte ens veta att dom körs i fönster.

LUXOR DATORER



A B C 1 6 0 0

KORT-BESKRIVNING AV

ABC80

VDO-editorn.

ver 2.3 / 1984-01-17

Editorn initieras med RUN VDO.
Tryck ned en tangent för att starta editorn.

Kommandona som beskrivs nedan används för att redigera texten.

CTRL-KOMMANDON

CTRL-U Markören flyttas rakt upp till föregående rad
CTRL-N Markören flyttas rakt ned till nästa rad
CTRL-H eller vänsterpil:
Flytta markören åt vänster.
CTRL-J, CTRL-I eller högerpil:
Flytta markören åt höger.
CTRL-Y Flytta markören en sida bakåt.
CTRL-Å Flytta markören en sida framåt.
CTRL-O Flytta markören till början av texten.
CTRL-P Flytta markören till slutet av texten.
CTRL-A Radera tecknet under markören.
CTRL-U Radera rad, när cursor står först på rad.
Radera resten av rad, när cursorn ej står först på rad.
CTRL-Ö Flyttar markören till slutet av raden.
Upprepad tryckning flyttar markören till slutet av nästa rad.
CTRL-T Flyttar markören framåt till nästa tabulatorstopp.
RETURN eller **CTRL-M**
Används för radslut och för att dela en rad i två delar genom att placera markören där raden skall delas och trycka RETURN.

ESC-Kommandon
Dessa kommandon aktiveras med en tryckning av CTRL-Ä (ESC) och följs sedan av ytterligare ett tangentnedslag. När CTRL-Ä trycks visas följande meny:

S	Spara	L	Läs
D	Spara del	M	Markera
F	Sök/byt ut	!	Radera allt
*	Radera del	.	Textstatus
A	Avsluta		

ESC S Används för att spara texten på diskett eller kassettband.
ESC L Läser en text från en fil.
ESC D Spar den del av texten som finns mellan markeringen som görs med ESC M kommandot och markören.
ESC M Markering. Kommer ihåg markörens position i texten. Används tillsammans med kommandona ESC D (spara del) och ESC * (radera del).
ESC ! Raderar all text.
ESC * Raderar texten mellan markeringen skapad av ESC M och markörens nuvarande position.
ESC . Skriver ut tre tal på raden för felmeddelande.
1) Antal bytes i textbufferten till vänster om markören.
2) Anger totala antal bytes av text i textbufferten.
3) Anger antal bytes kvar i textbufferten.

ESC A Avsluta (återvänd till BASIC). Texten raderas ej och man kan återstarta med RUN VDO:.

SÖKKOMMANDON

ESC F inleder en sökning med utbyte. Editorn frågar här efter "Sökord?" och "Byt till?".
CTRL-K fortsätter en sökning inledd med ESC F och hittar nästa förekomst av texten.
CTRL-E byter ut den hittade texten mot innehållet i insättningsbufferten och fortsätter sökningen på samma sätt som CTRL-K.

EGNA PARAMETRAR I BASIC-PROGRAMMET VDO
Läs filen VDO.REM

PRINTERKODER
DATA PR,80,3,"Printerkoder"

Se annan plats för mer förklaring

Björn Gustavsson <485>

DATA PR,80,3,"Printerkoder"

VR: används hellre i programmet för att skriva ut text på seriell skrivare. Man kan då använda parallell skrivare utan att låsas av PR: -koden. Eller om man har seriell skrivarutgång, definierar själv PR:VSA..... och så vidare. 80 är radbredd. 3 är 1200 baud och 4 är 2400 baud. Rutinen stjäl 175 byte.

VDO2.32K-PROGRAMMET

Förklaring till printerrutinen för olika BAUD-tal

```
610 READ L%-----Läser 80 på rad 1070
                vilket är radlängd
620 IF L%<40% OR L%>255% L%=80%
630 READ B% : IF B%<1% OR B%>4% B%=3%

                Läser 3 på rad 1070
                vilket förklaras med
                nedan REM
                En 4 ger 2400 BAUD

640 IF B%=1% B1%=249% : REM 300 BAUD
650 IF B%=2% B1%=124% : REM 600 B
660 IF B%=3% B1%=62% : REM 1200 B
670 IF B%=4% B1%=28% : REM 2400 B
```

Rader 680-700 rest från test för högre hastighet 4800-19200 BAUD vilket inte går bra, -INTERRUPTET ger skräp ut, raderna kan strykas.

```
680 REM IF B%=5% B1%=12%
690 REM IF B%=6% B1%=5%
700 REM IF B%=7% B1%=2%
```

Skriv VR i stället för PR så kan Ni använda både PR: i CENTRONICS och VR: I V24-RUTINER

```
1070 DATA VR,80,4
```

Ni som har serieinterfejs redan öppnar förstäss med PR:VSA..... som vanligt, det går ju förbi bara PR:. I varje fall på min DATADISK 83 - där det var 1200 BAUD förinställt jag inte ville ha.

LIB-funktion
Ett program från en tidig kassett har Claes S 1182 lagt in i startprogrammet: 'CKOLON'. Detta program är ett LIB-program och finns lagrat i DOS-buffert 5 som ej används och det stjäl inget minne från huvudprogrammet. Det anropas med 'RUN C:' när det står ABC80 på skärmen och från programmet med CTRL-Ä och L, vid Infil: anges C: och return.
I den här versionen av VDO är det automatstart vilket gör att när C: är utförd 'felkoden 8 finns ej i detta system' ges samt gör återhopp in i texten igen. Detta fann jag olägligt och kan kringgås med att först ge ett return på frågan Infil:. När detta är gjort en gång kan man ge Infil: C:. LIB-listan finns kvar att studeras. Programmet återstartas med RUN VDO: och texten finns kvar.

I programmet CKOLON.BAS på kassett ett finns följande rad:

```
42 ONERRORGOTO 43 : OPEN 'c:' ASFILE 1%
   : CLOSE 1% : CHAIN '' : REM Här kan
   CHAIN göras till egna rutiner.
```

Den har jag delat på två rader enligt nedan.

```
41 ONERRORGOTO 43 : OPEN 'c:' ASFILE 1%
   : CLOSE 1%
42 POKE 65053%,210%:CHAIN "dr1:vdo2.32k"
```

Då får jag in LIB-funktionen C: vilken jag berättar om ovan.

<485> Björn Gustavsson
<1182> Claes Schibler

Instruktioner till

LÖKAREN.

ver 4.3

Bildskärmseditorn (som nu ska beskrivas) är ständigt inkopplad.

Ctrl-E "Fryser" programexekevering tills någon annan tangent trycks, om CTRL-P trycks under det att "HOLD" blinkar så fås avskrift av skärmen till skrivaren (PR:).

Ctrl-A Flyttar markören ett tecken åt vänster.
Ctrl-B Läser in allt mellan markören och på skärmen först förekommande mellanslag.
Ctrl-D Drar tillbaka texten på skärmen ett steg.
<— Vänsterpil. Tar bort ett tecken ur inmatningsbufferten och backa en position, utan att radera något på skärmen.

—> Högerpil. Kopiera in "överkört" tecken till inmatningsbufferten.
Ctrl-Q Flytta markören till övre vänstra hörnet.
Raderar ingenting på skärmen.
Ctrl-L Tömmer skärmen utan att förstöra inmatningsbufferten.

Ctrl-O Stänger av TRACE.
Ctrl-P Skjuter fram resten av texten på skärmen ett steg.
Ctrl-S Flyttar markören ett steg åt höger.
Ctrl-T Sätter på TRACE.
Ctrl-U Skriver ut det som finns i inmatningsbufferten på det ställe på skärmen där <RETURN> sist trycktes.
Ctrl-W Flyttar markören en rad uppåt.
Ctrl-V Byter skärmbredd från 40->80, och vice versa. OBS: Endast för MyAB & GeJo.
Ctrl-X Tömmer inmatningsbufferten, och flyttar markören till början av raden.
Ctrl-Z Flyttar markören en rad neråt.
Ctrl-Ö Tömmer resten av skärmen.
Ctrl-U Sätter på eller stänger av INSERT-mode.
Om INSERT är påsatt så kommer texten framför markören att skjutas framåt för varje tecken du matar in.
Ctrl-Shift-O (Bokstaven O)
Bryter allting. Om datorn arbetade på disken kan det ta ca: 15 sekunder innan den börjar läsa eller skriva igen vid nytt anrop. Detta avhjälpes med att skriva ; INP(7).

Följande extra-kommandon finns:

DEL n1-n2 Tar bort alla rader from. n1 tom. n2.
DEL - Tar bort hela programmet. Samma funktion som NEW.

DEL n1- Tar bort alla rader from. n1.
DEL -n1 Tar bort alla rader tom. n1.
Bindestrecket kan bytas ut mot ett komma.
VAR Listar alla variabler i programmet.
OBS: Denna funktion fungerar inte om du har ett syntaxfel i ditt program. T.ex: FOR utan NEXT. Hopp till en rad som inte finns etc.
VAR v Skriver ut radnumren som variabeln v finns på.
Vektorer skrivs som v(), och matriser som v(,).
RUN nn Startar exekeveringen på rad nn.
Nollställer alla variabler, och stänger alla filer.
CON nn Fortsätter exekevering på rad nn efter Ctrl-C, ett fel eller STOP.
Nollställer inga variabler, och stänger inga filer.
OBS: Det går ej att fortsätta i subrutiner eller loopar.

FIND <text> Letar upp <text> i programmet och listar alla rader där <text> finns.
SYS Skriver ut BOFA, EOFA, HEAP och STACK-pekarna samt hur många bytes programmet tar upp (FILE), och antalet lediga bytes (FREE). Skriver dessutom ut vilka enheter som finns i systemet.

SYS B=nn Skriver ut samma saker som SYS, men ställer om värdet på BOFA-pekaren till nn.
SYS E=nn Samma som ovan, men för EOFA-pekaren.
SYS S=nn Samma som ovan, men för STACK-pekaren.
HELP Visar tillgängliga kommandon.
EXIT Avslutar LÖKAREN. Ställer om BOFA och I-registret till vad dom var innan LÖKAREN laddades in.
LIB Visar diskettinnehållet på båda drivarna (0/1).
LIB n Visar diskettinnehållet på drive n. N får vara minst 0 och max 7.
DISP <fil> Visar filen <fil> på skärmen, rad för rad.
Utan att förstöra något i minnet. Klarar både text- och binärfiler.
OBS: Om du enbart har kassett, så tas LIB och DISP bort ur kommandolistan, och du spar därmed en massa minne.
OLD Tar fram ett program efter RESET, NEW, SCR, DEL-.

OLD nn Som ovan, men ställer om värdet på BOFA-pekaren till nn.
AUTO Ger automatisk radnumrering från rad 10, med intervallen 10.
Om du redan har ett program i minnet kommer kommandot att börja från sista radnummer + 10.
AUTO n1 Som ovan men med intervall n1.
AUTO n1,n2 Ger automatisk radnumrering från rad n1, med intervall n2.
Om du börjar en rad i AUTO med ett radnummer så skrivs raden i på det nummret, och AUTOs radnummer kommer upp igen.
LIST Fungerar som vanliga LIST, men listar endast de 15 första skärmraderna. Kan för övrigt förkortas till U (tyskt-Y).
CHANGE v-v1
Byter namnet på variabeln v till v1 i hela programmet. Samma konventioner som vid VAR v.
RENUM n1-n2,n3
Delrenumrering av ett program. Från rad n1, till rad n2, med steget n3.
STEP Stegvis exekevering av ett program.
STEP kör programmet rad för rad och väntar på en tangenttryckning före varje rad. Om du då skriver ett L, så listas hela raden ett T, så skrivs radnumret ut eller ett E, så körs programmet som vanligt.
STEP <fil> Som ovan med laddar först in filen <fil>.
STEP nn Som STEP, med börjar på rad nn *UTAN* att nollställa några variabler eller stänga några filer.
RAM Visar interkoderna från och med första raden.
Skriver ut radnr. och startadress för raden och sedan koderna. RAM väntar sedan på en tangenttryckning, avbryt med <RETURN>.
RAM nn Som ovan, med börjar from. rad nn.
PEEK nn Listar minnesinnehållet enlingt följande:
Dec.adr. Dec.inh. Hex.adr. Hex.inh. ASCII word.
Word fungerar enligt följande exempel. Säg att PEEK befinner sig på adress 10 word skulle då bli det samma som ; PEEK(10)+256*PEEK(11).
Man kan scrolla åt båda hållen i PEEK, genom att använda piltangenterna.

Hoppas allt fungerar som det ska säger vi som varit med om att göra LÖKAREN.
Vi är:

Kalle Lindström -TCP- <837>
Allan Varcoe <3435>
Gunnar Forsell <1631>
Niclas Wiberg <1382>

JÄMFÖRELSE.

mellan Superbasic, Lökaren, SmartAid 3 och SupersmartAid

namn	Superbasic	Lökaren	Smartaid 3	Supersmartaid
typ	ROM	RAM	ROM	ROM (+CMOS RAM)
adressomr.	16384-22527 28672-30719	BOFA+4270	16384-22527	-6143 30720-31743
tillverkare	Nyfors Data	"ABC-klubben"	Owoco	Owoco
ca. pris	1000:-	140:-	1000:-	2000:-

Bildskärmseditorn.

Gemensamt för alla är markörstyrningen med ctrl -A -W -S -Z samt anpassningen till både 40 och 80 teckens skärm.

--> läser in tecken från bildskärmen till radbufferten.
<-- tar bort överkört tecken ur radbufferten.

Markören till första positionen på skärmen görs med ctrl-Q på alla utom Superbasic som gör det med ctrl-K.

Ctrl-L	tömmer skärmen (ctrl-L + ctrl-L i båda Smartaids).
Ctrl-P	skjuter text framåt.
Ctrl-D	"suger upp" text (inte i Superbasic).
Ctrl-X	som standard men suddar inte texten.
Ctrl-B	skriver ut radbufferten (ctrl-U i Lökaren).

I övrigt finns många Ctrl-funktioner som dock inte stämmer mellan de olika hjälparna.

Superbasic

Ctrl-Ö	raderar resten av raden.
Ctrl-Shift-O	(bokstaven O) fryser körning och därefter: Ctrl-B ger avbrott Ctrl-P ger skärmdump (ej grafik) Ctrl-Shift-O fortsätter körningen.

Lökaren

Ctrl-E	fryser körningen, alla tangenter fortsätter körningen igen
Ctrl-T	TRACE
Ctrl-O	(bokstaven O) NO TRACE
Ctrl-V	växlar 40/80 tecken
Ctrl-Ö	tömmer skärmen efter markören.
rl-U	slår på eller av "insert mode" dvs tecken som skrivs i en textmassa skjuts in mellan de övriga
Ctrl-Shift-O	(bokstaven O) ger avbrott.

Smartaid 3

Ctrl-Ä	tömmer radbufferten, markören står kvar
Ctrl-C	fryser körningen och därefter: Ctrl-S ger stegvis körning Ctrl-T ger stegvis med TRACE RETURN eller Ctrl-C igen bryter Ctrl-Shift-O ger skärmdump övriga tangenter fortsätter körningen.

Supersmartaid

Ctrl-U	variant av Ctrl-D (finns också)
Ctrl-V	läser in allt fram till nästa dubbla mellanslag
Ctrl-Ä	tömmer radbufferten, markören står kvar
Ctrl-Y	stoppar jobström
Ctrl-F	gör inmatning av ASCII 0-127 möjlig
Ctrl-G	pip

Kommandon.

AUTO	automatisk radnumrering
BIB	(Superbasic) visa biblioteket på 5" skiva
BOFA	(Superbasic) visar och ställer BOFA-pekaren
BYT	(Superbasic) byter variabelnamn och visar sedan aktuella rader
CHANGE	byter variabelnamn
CLEAR	ändring av HEAP-pekaren (Supersmartaid)
CMD	Startar upp Superbasic, kan användas i program och ger då möjlighet att använda alla kommandon i programmet.
CON	(Lökaren) återstartar avbruten programkörning
CONT	(Supersmartaid) som CON
COPY	(Superbasic) kopierar en fil mellan två skivenheter
DATE	(Superbasic) visar systemets datum
DEC	(Superbasic) omvandlar decimaltal till hexadecimaltal och binära tal
DEL	tar bort basicrader i ett program
DIR	(båda Smartaids) visar biblioteket på 8" skiva
DISP	(Supersmartaid, Lökaren) visar en fil från skiva på skärmen
ED	(Supersmartaid) ändra flera programrader
EDIT	Superbasic) ett nästan komplett ordbehandlingsprogram med över 20 interna kommandon!!
EOFA	(Superbasic) visar och ställer EOFA-pekaren
EXEC	jobströmshantering
EX eller EXIT	kopplar ur hjälparen
FIND	söker i programmet
FIX	(Superbasic) gör så att GOTO och GOSUB hoppar till befintliga adresser
FOR...NEXT	(båda Smartaids) kan användas som kommando
HELP	visar kommandona (MENY i Superbasic)
HEX	(Superbasic) omvandlar hexadecimaltal till decimaltal och binära tal
JOB	jobströmshantering
KEY	(Supersmartaid) definierar egna tangenter
LIB	visa biblioteket på 5" skiva (Superbasic även UFD--dos och ABC-NET dos)
LIST eller U	programlistning utan inledande scrollning
MEDIT	minneseditor, i ASCII eller hex, där man kan läsa eller skriva direkt i minnet
NEW	(Supersmartaid) ställer om BOFA-pekaren
NOTIME	(Supersmartaid) stänger av visningen av klockan
OLD	(Supersmartaid) återställer programpekare
PAC	(Superbasic) komprimerar ett basicprogram i minnet
PEEK	visar minnet både i ROM och RAM (SCAN i Superbasic)
PEW	(Supersmartaid) visar ett 16-bitars tal i minnet
POW	(Supersmartaid) pokar in ett 16-bitars tal
RAM	(Superbasic, Lökaren) visar kompilerad basickod
REN	(båda Smartaids) utökad renumrering
RENUM	(Lökaren) utökad renumrering
RESUME	återstartar programkörning
RUN	(Lökaren) om radnummer anges startas körningen på denna rad
SCAN	(Superbasic) nästan samma som PEEK i övriga
SPOOL	(Supersmartaid) direkt utskrift av fil från skiva
STACK	(Supersmartaid) ställer om stackpekaren
START	om radnummer anges startas körningen på denna rad (RUN i Lökaren). I Superbasic kan man dessutom återuppta avbruten körning
STEP	(Superbasic, Lökaren) stegvis körning, raden som ska exekveras visas först. I Superbasic kan man dessutom återuppta avbruten körning
SYS	visa några programpekare. I Lökaren kan man dessutom ställa programpekarna
TSAVE	(Supersmartaid) lagrar skapad textfil på skiva
TLOAD	(Supersmartaid) hämtar textfil från skiva
TMERGE	(Supersmartaid) lägger ihop två textfiler, en från skiva den andra i minnet
TAKET	(Superbasic) visar och ställer stackpekaren
TAB	(Supersmartaid) ställer tabvärdet för högerpil
TIME	(Supersmartaid) ställer och visar internklockan
TKN	(Superbasic) växlar 40/80 tecken
UP	(Superbasic) krypterar och krypterar upp variabler
VAR	listar programmets variabler
VISA	(Superbasic) samma som DISP

forts >