ABC80
ABC800

STYR OCH MÄT UNDER AVBROTT.

Eller hur jag får min ABCmaskin att göra flera saker samtidigt.

Givetvis kan en ABCmaskin inte göra flera saker samtidigt så undertiteln ovan är en överdrift. Men överdriften är inte så stor om man betänker att en nyss avfyrad gevärskula inte hinner mycket mer än en millimeter på den tid maskinen utför en instruktion i maskinkod. Om maskinen således gör en sak först och en annan sedan spelar därför ofta inte så stor roll. Låt mig illustrera med ett exempel. För att kontrollera en del av programmet som denna artiklen handlar om gjorde jag en uppkoppling och ett program för ABC80 och ABC800. Programmet var ett kort maskinprogram som lagts till huvudprogrammet. Detta korta program läser av klockans minst signifikanta byte och matar ut värdet till en digital-analogomvandlare på ett kort. D-A-omvandlaren kopplades till en förstärkare som driver en liten motor. D-A-omvandlaren och förstärkaren är gjorda så att om D-A-omvandlaren får byten 128 så står motorn still. Då byten är 255 går motorn med maximalhastighet framåt och om byten är 0 så går den med maximalhastighet bakåt. Klockans minst signifikanta byte räknas ner 50 gånger i s. Programmet genomlöpes några gånger i s och har till följd att motorn först roterar maximalt framåt. Därefter saktar motorn in för att efter ca 2.5 s stanna och börja gå åt andra hållet. Då ännu 2.5 s gått har motorn uppnått full fart bakåt. Därvid slår den plötsligt om till full fart framåt och förloppet upprepas om och om igen. Det ser ut som om motorn står där och andas. Stannar man sitt basicprogram med CTRL-C fortsätter motorn att andas. Därefter börjar man editera basicprogrammet och sedan lägga ut den nya versionen på skivan. Hela tiden andas motorn, till och med under själva arbetet mot disketten. I den situationen upplever åtminstone jag på ett tydligt sätt att maskinen samtidigt sköter både motorn och diskdriven. Det är inte förrän man trycker reset eller man kommenderar systemet att stanna som motorn påverkas i sin andning.

BASIC är ett språk som inte lämpar sig väl för styrning och mätning. BASIC beskriver saker som skall utföras. Det finns i BASIC goda möjligheter att styra i vilken ordning dessa saker skall ske. Då man styr och mäter är man oftast inte bara beroende av säkernas ordning utan också mycket bunden till när sakerna skall utföras. Eftersom BASIC inte har kommandon för tid så blir styrning och mätning i BASIC inte särskilt elegant. Låt oss titta på ett enkelt exempel. Antag att man skall mäta ett värde och lägga mätvärdet på en diskfil. Hur ofta kan man utföra mätningen? Oftast läggs mävärdet i någon dosbuffert och det går ganska snabbt. Då dosbufferten är full skall den läggas på skivan och det kan ta upp till sju sekunder. För att få en viss säkerhet säger vi var 10de sekund. Till slut har man ändå den oroskänslan att man kanske inte hittat det fall som tog längst tid. Ett värde var 10de sekund är inte speciellt imponerande. Ska man utföra lite besvärligare mätningar blir dessutom programmet mycket komplicerat och helt utan struktur.

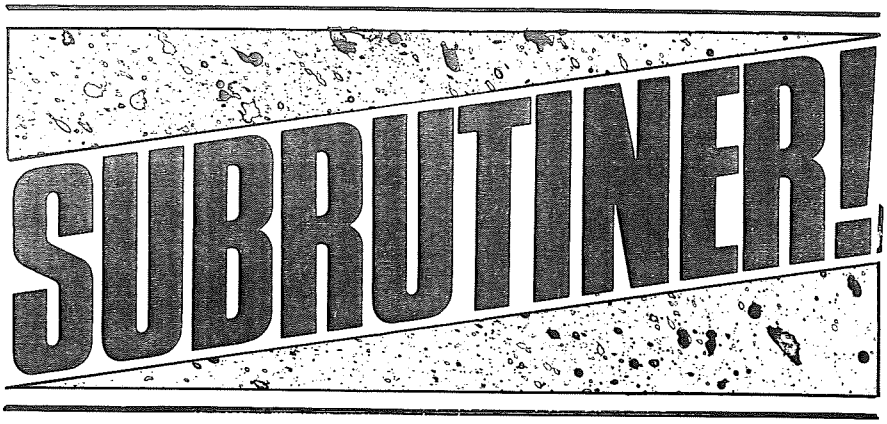
Mitt maskinspråksprogram har jag kallat ISM som står för interruptstyrd styrning och mätning. I ABC800 har jag avlett klockavbrotten så att de går via ISM. I ABC80 kan man såvitt jag vet inte enkelt avleda NMIavbrotten till något eget program. I ABC80 startar jag därför avbrotten från en egen låda som i stort sett bara innehåller en CMOSkrets och en fyrpolig DIP-omkopplare. Lådan anslutes till V24-kontakten från vilken den också får sin strömförsörjning. Lådan avger pulser till kontakten som är ansluten till en port. Porten är programmerad att starta ett avbrott som gör att ISM genomlöpes. Med DIPomkopplaren kan man ställa avbrottsfrekvensen mellan ett var åttonde sekund och 8192 avbrott per sekund. Maskinspråksprogrammet ISM ligger på address 32768 och uppåt. Programmet börjar med en tabell genom vilken all kommunikation med programmet går. Antag att man har tre rutiner som skall utföras enligt följande. Den första rutinen skall utföras varje avbrott, den andra rutinen vart annat avbrott och den tredje vart tionde avbrott. Varje sådan rutin kallas en servicerutin och är skriven i maskinspråk. Servicerutinerna måste således användaren ordna själv. Förutom den inledande tabellen innehåller ISM en tabell som beskriver servicerutinerna. För varje servicerutin står i tabellen hur många avbrott det skall vara mellan varje utförande och hur många avbrott som återstår till nästa gång servicerutinen utföres samt var i minnet servicerutinen finns. En stor del av ISM upptas för att sköta servicerutinerna och deras tabell. En annan stor del av ISM upptas av en bufferthanterare. Denna bufferthanterare användes för att ta hand om data som skall överföras från servicerutinerna till basic. Hanteraren håller reda på om bufferten innehåller data, om den är tom eller om den är överfull. Vidare finns i ISM rutiner för att starta och stoppa avbrotten. Såvitt jag känner till kan man inte avgöra vilket kort som valts sist om man inkluderar floppykortet. ISM addreserar därför alltid floppykortet som sista åtgärd. För att kunna använda andra andra kort från basic finns i ISM rutiner som sköter INP och OUT. Till slut finns i ISM en rutin som bryter all om man trycker Ctrl-Shift-O liksom i programmet 'hjälparen' (lökaren). Programmet är mindre än 3/4 kbytes långt och innehåller då en buffert på 256 bytes.

Ett program sådant som detta har givetvis både fördelar och nackdelar. Eftersom det är lite enklare att se nackdelarna så börjar jag med dem. På ABC80 används V24kontaktens IB0 och dess avbrottshantering. Detta får som konsekvens att kasetten inte går att använda för läsning. Jag har inte kontollerat om man kan använda kasetten för skrivning. Tidsstyrningen av pulsarna vid skrivning på kassetten sker med programloopar. Dessa avbryts av NMIklockan och om de tål ytterligare avbrott vet jag inte. Terminal och printerrutiner som använder V24kontakten får också problem. Ingångsbiten för terminalanvändning används i ISM för att generera avbrottet och utmatningsrutiner som använder V24 kan därför bara sända. Liksom i kassettrutinerna har printerrutinerna sin tidsstyrning bestämd av programloopar och således blir sändningen också ett problem. En test gav som resultat att man kan använda 1200 baud om servicerutinerna inte är särskilt långa. Ovanstående problem gäller inte ABC800. Det andra stora problemet har antytts ovan och har att göra med floppykortets kortval. ISM gör alltid kortval av floppykortet som avslutning. Har då ISM avbrutit ett annat kortval blir det fel då ISM är färdigt. Varje kortval måste därför ske med avbrottshanteringen avstängd. Sådana rutiner finns i ISM men gammal programvara kan vara svår att anpassa. System med flera kontrollerkort för drivar fungerar inte men det kan kanske ordnas på något sätt. För övrigt känner jag inte andra problem än sådana som är rent uppenbara. Programmet måste laddas in, det tar plats i minnet, det kan kollidera med annan programvara det skall skrivas i maskinspråk o. s. v.

Fördelarna med systemet är många. Det finns två fördelar som jag tycker är klart viktigare än de andra. Den första är möjlighet till styrning och mätning under arbete på flexskivorna. Den andra är den programmässiga uppdelningen i tidsberoende och icke tidsberoende delar. De tidsberoende delarna placeras i ISM och de icke tidsberoende i BASIC. Andra fördelar med systemet har mycket att göra med hur det är skrivet. Det finns t.ex möjligheter att mycket snabbt skifta från en uppsättning servicerutiner till en annan. Vidare finns det möjligheter att då och då utföra rutiner som tar lång tid att utföra. Ett exempel på en sådan rutin kan vara en rutin som lägger data på flexskiva.

Till programmen har jag gjort två användarmanualer på tre sidor, en för ABC80 och en för ABC800. Till avbrottsgeneratorn för ABC80 finns ett enkelt schema och jag har även några testprogram som är sämre dokumenterade ochmest framtagna för kontroll av ISM. Är Du intresserad så hör av dig med ett kort eller brev.

Lars Stage <2289>



ABC80
ABC806

SUBRUTIN "ENTER".

Programmet insänt av Sten Öhman, klin em lab, Regionsjukhuset, 591 85 Linköping.

Programmet finns i en version för ABC80 'ENTER80' och en för ABC806 'ENTER806'. PF-tangenterna i ABC806 används ej i subrutinen, men deras ASCII-värden kan användas i huvudprogrammet.

Subrutinen möjliggör att man kan mata in alternativt a) sträng, b) siffervärde, eller c) flagga för avlänkning i huvudprogrammet. Man kan även sätta rimlighetsgränser i huvudprogrammet och bryta gränserna genom att markera med ! i subrutinen. Flaggan F% avgör vad subrutinen skall returnera. Sätt F% till ett udda tal, om huvudrutinen testar om flagga sätts i subrutinen. Om F% är ett jämnt tal returneras alltid F%=0. Om huvudprogrammet sätter F%=0 eller 1 returneras flyttal i variabeln X, och strängen I$ är tom. Om F% sätts till 2 eller 3 returneras strängen I$, och variabeln X=0. Om F% sätts till 4 eller 5 returneras flyttalat X,och subrutinen testar, att värdet inte är mindre än X vid ingången i subrutinen. Om F% sätts till 6 eller 7 testas på samma sätt på maximum. Om F% sätts till 8 eller 9 testas både på min-värde och max-värde. Variabeln för min-värde är X0 och max-värde X9. Om F% > 9 returneras F% endast om tangenten med ASCII-värdet = F% trycks ned.

Kontrolltangenter används för redigering av inmatade värden enligt följande kod:

CTRL-E:
Raderar det tecken där cursorn står. Om cursorn står efter sista tecknet raderas sista tecknet.

CTRL-A:
Backar cursorn till närmaste blanktecken.

CTRL-B:
Backar cursorn ett steg.

CTRL-D:
Skjuter in blanktecken. (Mellanslag ger samma effekt).

CTRL-E:
Flyttar cursorn till radens slut.

CTRL-F:
Flyttar fram cursorn ett steg.

CTRL-G:
Flyttar fram cursorn till närmaste blanktecken.

CTRL-H: Reserverat för yttre styrning. Samma som 'pil-back' tangenten! Returnerar F%=8.

CTRL-I:
Reserverat för yttre styrning. Samma som 'pil-fram'-tangenten! Returnerar F% = 9.

CTRL-J: Repeterar senast inslagna värde. Lagras i strängen J$.

CTRL-M:
Samma som <RETURN>-tangenten!

<RETURN>: Returnerar inslaget värde till huvudrutinen. Cursorn behöver inte stå i slutet av strängen för returen. Om inslaget värde är felaktigt kommer felsignal och inget värde returneras.

Övriga kontrolltangenter: Returnerar sitt ASCII-värde i F%, om F% var udda vid ingången i subrutinen.

Sten öhman
klin kem lab, RiL

P.S.
Ytterligare en liten detalj i anvisningarna till programmet. Som framgår av funktionen FNZ9% returneras inte ASCII-värdet för flaggorna nr 14-32 utan 13 minus ASCII-värdet. Detta för att "sila bort" de interna kontrollkoderna inom subrutinen, och möjliggöra en ON-sats direkt vid returen utan "tomma" rader för de lägsta koderna (1-13).

```
10 REM PROGRAM 'TEST'
12 ; CHR$(12)CUR(5,0)
14 ; "FAS 1. MATA IN 5 TAL"
16 FOR I%=1 TO 5 : F%=0 : GOSUB 250 : ; X : NEXT I%
18 ; CHR$(12)CUR(5,0)
20 ; "FAS 2 MATA IN TAL. CTRL-TKN= SLUT"
22 FOR I%=1 TO 10 : F%=1 : GOSUB 250
24 IF F% THEN 28
26 ; X : NEXT I%
28 ; CHR$(12)CUR(5,0)
30 ; "FAS 3. MATA IN 5 STRÄNGAR"
32 FOR I%=1 TO 5 : F%=2 : GOSUB 250
34 ; I$
36 NEXT I%
38 ; CHR$(12)CUR(5,0)
40 ; "FAS 4. MATA IN STRÄNGAR. CTRL-TKN=SLUT"
42 FOR I%=1 TO 10 : F%=3 : GOSUB 250
44 IF F% 48
46 ; I$ : NEXT I%
48 ; CHR$(12)CUR(5,0)
50 ; "FAS 5. SKRIV MIN-GRÄNS" : F%=0 : GOSUB 250 : Y=X
52 ; "SKRIV EN SERIE TAL"
54 FOR I%=1 TO 10 : F%=5 : X=Y : GOSUB 250
56 IF F% THEN 60
58 NEXT I%
60 ; CHR$(12)CUR(5,0)
62 ; "FAS 5. SKRIV MAX-GRÄNSEN" : F%=0 : GOSUB 250 : Y1=X
64 ; "SKRIV EN SERIE TAL"
66 FOR I%=1 TO 10 : F%=7 : X=Y1 : GOSUB 250
68 IF F% THEN 72
70 NEXT I%
72 ; CHR$(12)CUR(5,0)
74 ; "FAS 7. MIN OCH MAXGRÄNS SOM OVAN.          SKRIV EN
   SERIE TAL"
76 FOR I%=1 TO 10 : F%=9 : X0=Y : X9=Y1 : GOSUB 250
78 IF F% THEN 82
80 NEXT I%
82 ; CHR$(12)CUR(5,0)"SLUT PA TEST"
84 END
200 REM
206 REM ******************************
208 REM
210 REM PROGRAM 'ENTER'
212 REM
214 REM INMATNINGSRUTIN, STEN ÖHMAN
216 REM
218 REM EDITION ABC-80, 1984-10-11
220 REM
222 REM ******************************
224 REM Flagga = F%. Sätt F% enligt :
226 REM F%=Jämn :  Ej flagga F% i retur
228 REM F%=Udda :  Flagga F% i retur
230 REM F%=0 el 1 : Input flyttal
232 REM F%=2 el 3 : Input sträng
234 REM F%=4 el 5 : Input flyttal med min-kontroll = X
236 REM F%=6 el 7 : Input flyttal med max-kontroll = X
238 REM F%=8 el 9 : Input flyttal med min-kontroll = X0 och
    max-kontroll = X9
240 REM
242 REM
244 REM
246 REM F%>9 : Flagga F% sätts om ASCII = F% matas in
248 REM ******************************
250 R9%=PEEK(65011%) : K9%=PEEK(65012%)
```

```
252 ONERRORGOTO 352
254 I$='' : L9%=0% : K8%=1%
256 ; CUR(R9%,K9%)I$CUR(R9%,K8%+K9%-1%); : GET Z$ : IF F9%
    GOSUB 354
258 Z9%=Z% : Z%=ASC(Z$)
260 IF Z%=13% THEN 374
262 IF F%>9% AND Z%=F% THEN 372
264 IF Z%<32% THEN 284
266 IF K8%<L9%+1% THEN 336
268 I$=I$+Z$ : L9%=LEN(I$) : K8%=K8%+1%
270 ; CUR(R9%,K9%)STRING$(40%-K9%,32%);
272 GOTO 256
274 REM ********************************
278 REM KONTROLLKODER
280 REM
282 REM ********************************
284 IF Z%>7% AND Z%<>10% THEN 340
286 ON Z%+1% GOTO 288,300,310,414,314,318,320,324,414,
    414,334
288 IF K8%=1% AND L9%>0% I$=RIGHT$(I$,2%) : GOTO 298
290 IF K8%=1% THEN 270
292 IF K8%>L9% I$=LEFT$(I$,K8%-2%) : GOTO 296
294 I$=LEFT$(I$,K8%-1%)+RIGHT$(I$,K8%+1%)
296 IF K8%>1% K8%=K8%-1%
298 L9%=LEN(I$) : GOTO 270
300 IF K8%<2% THEN 270
302 FOR P9%=K8%-1% TO 2% STEP -1%
304 IF ASC(MID$(I$,P9%,1%))=32% THEN 308
306 NEXT P9%
308 K8%=P9% : GOTO 270
310 IF K8%>1% K8%=K8%-1%
312 GOTO 270
314 I$=LEFT$(I$,K8%-1%)+CHR$(32%)+RIGHT$(I$,K8%)
316 L9%=LEN(I$) : GOTO 270
318 K8%=L9%+1% : GOTO 270
320 IF K8%<L9%+1% K8%=K8%+1%
322 GOTO 270
324 IF K8%>L9%-1% THEN 270
326 FOR P9%=K8%+1% TO L9%
328 IF ASC(MID$(I$,P9%,1%))=32% THEN 332
330 NEXT P9%
332 K8%=P9% : GOTO 270
334 I$=J$ : L9%=LEN(I$) : K8%=L9%+1% : GOTO 270
336 I$=LEFT$(I$,K8%-1%)+Z$+RIGHT$(I$,K8%)
338 K8%=K8%+1% : L9%=LEN(I$) : GOTO 270
340 X=0% : I$='' : Z9%=Z% : IF F% AND 1% THEN 376 ELSE 352
342 REM ********************************
346 REM FELHANTERING
348 REM
350 REM ********************************
352 ; CUR(R9%+1%,0%)CHR$(7%,127%,127%,32%)"FELAKTIG
    INMATNING! GÖR OM!"; : F9%=-1% : GOTO 254
354 ; CUR(R9%+1%,0%)STRING$(40%,32%); : F9%=0% : RETURN
356 ; CUR(R9%+1%,0%)CHR$(7%,127%,127%,32%)"UTANFÖR GRÄNSVÄ
    RDET! Tryck ! om OK"; : GET Z$
358 ; CUR(R9%+1%,0%)STRING$(40%,32%); : IF ASC(Z$)=33% THEN
    382
360 ; CUR(R9%,0%)STRING$(40%,32%); : GOTO 254
362 REM ********************************
364 REM
366 REM RETURN
368 REM
370 REM ********************************
372 X=0% : I$='' : GOTO 378
374 ON F%/2%+1% GOTO 380,384,386,392,398
376 IF F% AND 1% F%=FNZ9%(Z9%) ELSE F%=0%
378 ; : J$=I$ : ONERRORGOTO 408 : RETURN
380 IF FNI9% THEN 352
382 X=VAL(I$) : GOTO 376
384 X=0% : GOTO 376
386 IF FNI9% THEN 352
388 F9%=INSTR(1%,I$,"!") : IF F9% I$=LEFT$(I$,F9%-1%)
390 IF VAL(I$)<X AND F9%=0% THEN 356 ELSE 382
392 IF FNI9% THEN 352
394 F9%=INSTR(1%,I$,"!") : IF F9% I$=LEFT$(I$,F9%-1%)
396 IF VAL(I$)>X AND F9%=0% THEN 356 ELSE 382
398 IF FNI9% THEN 352
400 F9%=INSTR(1%,I$,"!") : IF F9% I$=LEFT$(I$,F9%-1%)
402 IF (VAL(I$)<X0 OR VAL(I$)>X9) AND F9%=0% THEN 356 ELSE
    382
404 DEFFNZ9%(Z0%)=-Z9%*(Z9%<14%)-(Z9%-13%)*(Z9%>13 AND Z9%<
    32)
406 DEFFNI9%=ASC(I$)>57% OR ASC(I$)<45% OR ASC(I$)=47%
408 REM *** EXIT ***
410 ; "FEL HAR UPPSTÄTT. FELKOD = "ERRCODE
412 END
414 ; "EJ PROGRAMMERAT"
416 END
```

```
10 ! Program TEST
12 ; CHR$(12) CUR(5,0)
14 ; "Fas 1. Mata in en serie på 5 tal"
16 FOR I=1 TO 5 : F=0 : GOSUB 250 : ; X. : NEXT I
18 ; CHR$(12) CUR(5,0)
20 ; "Fas 2. Mata in en serie tal. Kontrolltecken = slut"
22 FOR I=1 TO 10 : F=1 : GOSUB 250
24   IF F THEN 28
26 ; X. : NEXT I
28 ; CHR$(12) CUR(5,0)
30 ; "Fas 3. Mata in 5 strängar"
32 FOR I=1 TO 5 : F=2 : GOSUB 250
34   ; I$
36 NEXT I
38 ; CHR$(12) CUR(5,0)
40 ; "Fas 4. Mata in strängar. Kontrolltecken = slut"
42 FOR I=1 TO 10 : F=3 : GOSUB 250
44   IF F THEN 48
46 ; I$ : NEXT I
48 ; CHR$(12) CUR(5,0)
50 ; "Fas 5. Skriv min-gränsen" : F=0 : GOSUB 250 : Y.=X.
52 ; "Skriv en serie tal"
54 FOR I=1 TO 10 : F=5 : X.=Y. : GOSUB 250
56   IF F THEN 60
58 NEXT I
60 ; CHR$(12) CUR(5,0)
62 ; "Fas 6. Skriv max-gränsen" : F=0 : GOSUB 250 : Y1.=X.
64 ; "Skriv en serie tal"
66 FOR I=1 TO 10 : F=7 : X.=Y1. : GOSUB 250
68   IF F THEN 72
70 NEXT I
72 ; CHR$(12) CUR(5,0)
74 ; "Fas 7. Min- och max-gräns som ovan. Skriv en
   serie tal."
76 FOR I=1 TO 10 : F=9 : X0.=Y. : X9.=Y1. : GOSUB 250
78   IF F THEN 82
80 NEXT I
82 ; CHR$(12) CUR(5,0) "SLUT PÅ TEST"
84 END
200 !
206 STOP
208 ! ****************************************
210 !
212 ! PROGRAM 'ENTER'
214 !
216 ! INMATNINGSRUTIN, STEN ÖHMAN
218 !
220 ! EDITION ABC-806 1984-10-11
222 !
224 ! ****************************************
226 ! Flagga = F%. Sätt F% enligt önskemål:
228 ! F%=Jämn: Ej flagga F% i retur; F%=Udda: Flagga F% i
   retur
230 ! F% = 0, 1:  Input flyttal
232 ! F% = 2, 3:  Input av sträng med ev. blankar
234 ! F% = 4, 5:  Input av flyttal med min-kontroll = X
236 ! F% = 6, 7:  Input av flyttal med max-kontroll = X
238 ! F% = 8, 9:  Input av flyttal med min-kontroll = X0,
   maxkontroll = X9
240 !
242 !
244 !
246 ! F% > 9:  Flagga F% sätts om ASCII = F% matas in
248 ! ****************************************
250 R9=PEEK(65363) : K9=PEEK(65362)
252 ON ERROR GOTO 352
254 I$='' : L9=0 : K8=1
256 ; CUR(R9,K9) I$ CUR(R9,K8+K9-1); : GET Z$ : IF F9
   GOSUB 354
258 Z9=Z : Z=ASCII(Z$)
260 IF Z=13 THEN 374
262 IF F>9 AND Z=F THEN 372
264 IF Z<32 THEN 284
266 IF K8<L9+1 THEN 336
268 I$=I$+Z$ : L9=LEN(I$) : K8=K8+1
270 ; CUR(R9,K9) STRING$(80-K9,32);
272 GOTO 256
```

```
274 ! ************************************
276 !
278 ! KONTROLLKODER
280 !
282 ! ************************************
284 IF Z>7 AND Z<>10 THEN 340
286 ON Z+1 GOTO 288,300,310,430,314,318,320,324,430,430,334
288 IF K8=1 AND L9>1 I$=RIGHT$(I$,2) : GOTO 298
290 IF K8=1 THEN 270
292 IF K8>L9 I$=LEFT$(I$,K8-2) : GOTO 296
294 I$=LEFT$(I$,K8-1)+RIGHT$(I$,K8+1)
296 IF K8>1 K8=K8-1
298 L9=LEN(I$) : GOTO 270
300 IF K8<2 THEN 270
302 FOR P9=K8-1 TO 2 STEP -1
304   IF ASCII(MID$(I$,P9,1))=32 THEN 308
306 NEXT P9
308 K8=P9 : GOTO 270
310 IF K8>1 K8=K8-1
312 GOTO 270
314 I$=LEFT$(I$,K8-1)+CHR$(32)+RIGHT$(I$,K8)
316 L9=LEN(I$) : GOTO 270
318 K8=L9+1 : GOTO 270
320 IF K8<L9+1 K8=K8+1
322 GOTO 270
324 IF K8>L9-1 THEN 270
326 FOR P9=K8+1 TO L9
328   IF ASCII(MID$(I$,P9,1))=32 THEN 332
330 NEXT P9
332 K8=P9 : GOTO 270
334 I$=J$ : L9=LEN(I$) : K8=L9+1 : GOTO 270
336 I$=LEFT$(I$,K8-1)+Z$+RIGHT$(I$,K8)
338 K8=K8+1 : L9=LEN(I$) : GOTO 270
340 X.=0 : I$='' : Z9=Z : IF F AND 1 THEN 376 ELSE 352
342 ! ****************************************
344 !
346 ! FELHANTERING
348 !
350 ! ****************************************
352 ; CUR(R9+1,0) CHR$(7,127,127,32) "FELAKTIG INMATNING!
    GÖR OM!" : F9=-1 : GOTO 254
354 ; CUR(R9+1,0) STRING$(80,32); : F9=0 : RETURN
356 ; CUR(R9+1,0) CHR$(7,127,127,32) "UTANFÖR GRÄNSVÄRDET!
    Tryck på ! om OK"; : GET Z$
358 ; CUR(R9+1,0) STRING$(80,32); : IF ASCII(Z$)=33 THEN
    382
360 ; CUR(R9,0) STRING$(40,32); : GOTO 254
362 ! ****************************************
364 !
366 ! RETURN
368 !
370 ! ****************************************
372 X.=0 : I$='' : GOTO 378
374 ON F/2+1 GOTO 380,384,386,392,398
376 IF F AND 1 F=FNZ9(Z9) ELSE F=0
378 ; : J$=I$ : ON ERROR GOTO 408 : RETURN
380 IF FNI9 THEN 352
382 X.=VAL(I$) : GOTO 376
384 X.=0 : GOTO 376
386 IF FNI9 THEN 352
388 F9=INSTR(1,I$,"!") : IF F9 I$=LEFT$(I$,F9-1)
390 IF VAL(I$)<X. AND F9=0 THEN 356 ELSE 382
392 IF FNI9 THEN 352
394 F9=INSTR(1,I$,"!") : IF F9 I$=LEFT$(I$,F9-1)
396 IF VAL(I$)>X. AND F9=0 THEN 356 ELSE 382
398 IF FNI9 THEN 352
400 F9=INSTR(1,I$,"!") : IF F9 I$=LEFT$(I$,F9-1)
402 IF (VAL(I$)<X0. OR VAL(I$)>X9.) AND F9=0 THEN 356
    ELSE 382
404 DEF FNZ9(Z0)=-Z9*(Z9<14)-(Z9-13)*(Z9>13 AND Z9<32)
406 DEF FNI9=ASCII(I$)>57 OR ASCII(I$)<45 OR ASCII(I$)=47
408 REM *** EXIT ***
410 ; CHR$(12,10,10,10) "FEL MED FELKOD " ERRCODE "HAR
    UPPSTÄTT"
412 ; "PROGRAMMET STOPPAT"
414 ; : ; "TRYCK PÅ <RETURN> OM DU VILL ÅTERGÅ TILL MENYN"
416 ; "ELLER ANNAN TANGENT FÖR PROGRAMSLUT"
418 GET Z$ : IF ASCII(Z$)=13 THEN 206
420 CLOSE
422 ; CHR$(12) CUR(10,10) "*** HEJ DÅ! ***" CUR(22,0)
424 END
426 ; CHR$(12,10,10) "        *** PROGRAMMET KLART? ***"
    CHR$(10,10)
428 GOTO 414
430 ; "TYVÄRR!!!" CHR$(10,10) "PROGRAMMET ÄR INTE KLART
    ÄN!"
432 GOTO 414
```

ABC80

IN.BAS.

Denna subrutin klarar av att göra något under tiden som man matar in. Nu visas tiden som exempel, därför kan subrutinen vara litet långsam.

De invariabler subrutinen behöver är Y och X som står för den rad respektive kolumn som inmatningen börjar på och L som står för maximala längden av indatat. Indata returneras i I$.

Funktionen FNA(Y,X) som definieras på rad 290, används till att bestämma adressen till positionen Y,X. På red 300 får Y1 och X1 samma värde som Y respektive X. Dessa två variabler används för att komma ihåg ursprungspositionen. De behövs när man trycker på Ctrl-X.

Rad 320-330 nollställer flaggan som talar om att en tangent har tryckts ner och väntar på att man skall hinna släppa en eventuellt nedtryckt tangent. En blinkande markör sätts ut på position Y,X på rad 340.

T1 blir lika med T på rad 360. På senare plats i programmet används T1 för att kolla om man har hunnit släppa tangenten.

Rad 370 känner av tangentbordet och lägger resultatet i T.

Om man har överskridit den maximala längden på indatat, så plingar det till och det sista tecknet tas bort. Det händer på rad 380.

Rad 390 känner av att man har hunnit släppa tangenten. Är så inte fallet, så går programmet till rad 340.

Om man inte har tryckt ner ett Ctrl-tecken, så skrivs tecknet ut på rad 400 och läggs även i I$.

Rad 410 sköter om radering vid nedtryckning av bakåtpil.

Har man tryckt på RETURN så hoppar datorn till rad 470. Detta känns av på rad 420.

Är Ctrl-X nedtryckt, så sköter rad 430 om att I$ töms och att markören hoppar tillbaka till utgångskolumn och rad.

Är markören i 39:nde kolumnen så fixar rad 440 att den hoppar ner till nästa rad. Om man har tryckt på bakåtpil först på en rad, så sköter rad 450 om att det fungerar ändå.

Rad 470 med sitt RETURN innebär att subrutinen är slut.

På rad 500 - 989 lägger du in den rutin som datorn skall utföra under tiden. Här ligger klockrutinen nu, med den tar man enkelt bort.

<4323>
Patrik Eveborn

```
 10 REM ---------------------
 20 REM Testprogram
 30 Y=10 : X=0 : L=40
 40 GOSUB 200
 50 END
 60 REM Slut på testprogram
200 REM ----------------------
210 REM IN.BAS
220 REM Inputrutin som klarar av
230 REM att göra något annat
240 REM samtidigt för ABC80
250 REM 1985-02-22
260 REM Patrik Eveborn <4323>
270 REM ------------------------
280 REM
290 DEFFNA(Y,X)=PEEK(884+Y*2)+256*PEEK(8
    85+Y*2)+X
300 Y1=Y : X1=X
310 DIM I$=80
320 POKE 65013,0
330 FOR J=1 TO 500 : NEXT J
340 POKE FNA(Y,X),160
350 GOSUB 500
360 T1=T
370 T=INP(56)-128
380 IF I>L THEN ; CHR$(7) : I=L : POKE F
    NA(Y,X),32 : X=X-1 : I$=LEFT$(I$,LEN
    (I$)-1) : GOTO 450
390 IF T=T1 THEN 340
400 IF T>31 THEN ; CUR(Y,X)CHR$(T) : I=I
    +1 : X=X+1 : I$=I$+CHR$(T)
410 IF T=8 AND I<>0 THEN I=I-1 : POKE FN
    A(Y,X),32 : X=X-1 : I$=LEFT$(I$,LEN(
    I$)-1)
420 IF T=13 THEN POKE FNA(Y,X),32 : GOTO
    470
430 IF T=24 THEN ; CUR(Y1,X1)SPACE$(L) :
    I=0 : X=X1 : I$='' : Y=Y1
440 IF X=40 THEN Y=Y+1 : X=0
450 IF X=-1 THEN X=39 : Y=Y-1
460 GOTO 340
470 RETURN
500 R1%=255% XOR PEEK(65008%) : IF R1%>2
    50% THEN 500
510 R2%=255% XOR PEEK(65009%)
520 R3%=255% XOR PEEK(65010%)
530 Z=(SWAP%(R3%)+R2%)*5.12+R1%/50 : R1%
    =-1*INT(Z/30)
540 H=INT(Z/3600) : Z=Z-H*3600 : M=INT(Z
    /60) : Z=Z-M*60
545 Z=INT(Z)
550 H$=RIGHT$(NUM$(H),2) : IF H<10 THEN
    H$='0'+H$
560 M$=RIGHT$(NUM$(M),2) : IF M<10 THEN
    M$='0'+H$
570 Z$=RIGHT$(NUM$(INT(Z)),2) : IF Z<10
    THEN Z$='0'+Z$
580 ; CUR(0,30)H$':'M$':'Z$
990 RETURN
```

MSG

1985-01-26 20.55.45
Ola Arwidsson <5275>
Inlägg i mötet: ABC80
ÄRENDE: INTERUPT

Som någon annan påpekat så kommer man ibland själv på lösningen till ett problem som man frågat om. Någon kanske påminner sig min fråga om hur en interuptrutin skall utformas. Nåväl, här kommer en. Den använder sig av PIO:s 128 microsekundersinterupt. Den innehåller en rutin som får tangentbordet att fungera normalt nästan). Även en rutin som visar att det hela fungerar som det ska.

```
ORG 32767 (t.ex) START JR STRT INTV
DEFW INTRUT STRT DI
   LD A,79
   OUT (57),A
   LD A,135
   OUT (57),A
   LD HL,INTV
   LD A,L
   OUT (57),A
   LD A,H
   LD I,A
   DEFB 251;EI;ASSEMBLERN TAR EJ EI
   RET
INTRUT PUSH AF
   LD A,(65535)
   DEC A
   LD (65535),A
   JR NZ,SLUT
   LD A,255 ;FÖRDRÖJNINGSRÄKNARE
   LD (65535),A
   IN A,(56)
   BIT 7,A
   JR Z,I1
   CALL TKN
   JR I2
I1 XOR A
   LD (65533),A I2 CALL RUTIN
SLUT POP AF

   DEFB 251;EI
   RETI
TKN IN A,(56)
   CP 131 ;CTRL-C
   JR NZ,T1
   LD (65031),A
   LD A,128
   OUT 6,A
T1 LD A,(65533)
   CP 70
   RET Z
   LD A,128
   LD (65013),A
   LD A,70
   LD (65015),A
   LD (65533),A
   RET
RUTIN PUSH HL
   PUSH DE
   PUSH BC
   LD A,(65534)
   LD C,A
   LD HL,31744
   LD DE,TEXT
   LD B,0
   ADD HL,BC
   EX DE,HL
   LD BC,15 ;LÄNGDEN AV TEXT
   LDIR
   LD C,A
   CP 25 ;40-LÄNGDEN AV TEXT
   JR NZ,R1
   LD A,I
   LD (65532),A
R1 CP 0
   JR NZ,R4
   XOR A
   LD (65532),A
   R4 LD A,(65532)
   CP 0
   JR Z,R2
   DEC C
   JR R3 R2 INC C
R3 LD A,C

   LD (65534),A
   POP BC
   POP DE
   POP HL
   RET TEXT DEFM " <<BY OLA A.>> "
```

Gör CALL på START och det ska fungera. Ni får ursäka om programmet inte är så 'smart' skrivet.

Ola Arwidsson <5275>

1984-10-28 14.47.32
David Asztely <2920>
Inlägg i mötet: ABC80

Ärende: SETDOT
Är det inte typiskt???
När man väl frågat någon så kommer man på lösningen själv!!
(Precis vad som hände mig idag)
Här kommer min lösning:

```
ORG 65408
LD H,0  ;HL ska innehålla
LD L,E  ;raden
LD B,0  ;BC ska innehålla
LD C,D  ;kolumm
CALL 8940  ;checksumma 9913(?!?)
           ; PEEK(8940)=124
           ; vid retur innehåller
           ; HL address i bild-
           ; minnet
           ; och A-registret
           ; innehåller en
           ; "ASCII" kod.
OR (HL)     ; innehållet i bildm.
.          ;"adderas" t. A-reg.
OR 32       ; sen "adderas" 32 t. A
LD (HL),A   ; Nu skrivs punkten ut!
RET         ; åter till basic

indata:
D - kolum
E - rad
```

Kanske är detta värt att använda ifrån assembler om man inte skriver en egen rutin.

M.v.h.
David A <2920>

1984-11-02 20.12.57
David Asztely <2920>
Inlägg i mötet: ABC80

Ärende: CLRDOT

Här kommer nu på mångas (???) begäran en CLRDOT-rutin i assembler.

```
ORG 65408
LD H,0
LD L,E
LD B,0
LD C,D
CALL 8940
CPL   ; komplementera A-reg
AND (HL)
OR 32
LD (HL),A
RET
```

Se även mitt första inlägg angående SETDOT

M.v.h
David A

1984-11-15 08.38.17
David Asztely <2920>
Inlägg i mötet: ABC80

Ärende: DOT

Jag tänkte nu avsluta min "serie" om hur man kan anropa ABC80's grafikrutiner i från assembler.

Denna gång: DOT

```
ORG 65408
LD H,0
LD L,E
LD B,0
LD C,D
CALL 8940
AND (HL)
LD L,A
LD H,0
RET

Vid retur:
L = 0      punkten släckt
L <> 0     punkten tänd
```

För vidare information se även min andra inlägg!

Mvh
David
<2920>

Kassettbandspelare till ABC806.

Här följer en beskrivning på hur man kan ansluta en kassettbandspelare till ABC806. Först måste Du kontrollera att din 806:a har kvar sitt kassettinterface. Luxor har nämligen slutat med att sätta i de komponenter som enbart används till kassettinterfacet. Det enklaste är att kontrollera om reläet till motorstyrningen finns med. Om så är fallet finns säkerligen resten av komponenterna också med. Reläet sitter precis framför tangentbordskontakten.

Framför reläet sitter ett bygelfält, som ser ut som på bilden nedan. Byglarna är i standardutförannde placerade så att mittraden är ansluten till den främre raden. Om Du flyttar byglarna så att mittraden ansluts till den bakre raden, så har Du förvandlat Ditt tangentbordsuttag till bandspelaruttag, men det är litet svårt att få igång bandspelaren eftersom Du nu saknar ett fungeradne tangentbord.

Dessutom saknar Du en drivrutin för bandspelare, eftersom denna på grund av platsbrist har tagits bort ur prommarna i 806:an. Det sistnämnda problemet kan vi lösa på två sätt. Antingen skriver vi en egen drivrutin som vi laddar från skiva, eller så byter vi ut vårt optionsprom mot det som sitter i 802:an, ty detta innehåller en kassettdrivrutin.

Det troliga är dock att vi även vill ha tillgång till 806:ans optionsprom på ett enkelt sätt ,eftersom det innehåller rutinerna för HR-grafiken.

Den nedan beskrivna modifieringen visar, dels hur Du ansluter kontakten till din bandspelare och dels hur Du monterar in två optionsprom som Du kan växla mellan med en omkopplare.

För att göra modifieringen behövs följde komponenter.

1 st 5-polig DIN-kontakt, hona
en liten bit kabel, 5-ledare

Om Du dessutom vill har 802:ans optionsprom i samtidigt med 806:ans så behöver Du

2 st motstånd, 1 kOhm
1 st en-polig omkopplare
några decimeter tunn enkel-ledare
optionsprom från en 802:a

Den 5-poliga DIN-kontakten ansluter Du på byglingsfältet enligt skissen nedan. Pinne 2 i DIN-kontakten skall kopplas till signaljord, t ex vänster sida på motstånd R37 (sett från datorns sida).

Monteringen av de två optionsprommen.

Börja med att montera ur det befintliga optionsprommet i position J2. Böj nu försiktigt ut pinne 18 så att den står rakt ut och gör sedan likadant på prommet från 802:an.

Därefter monterar Du prommen ovanpå varandra och löder försiktigt ihop dem (utom pinne 18). Om Du har valt omkopplare av lämplig storlek kan Du montera den i ett av de två hål som finns ovanför V24-anslutningarna på 806:ans baksida.

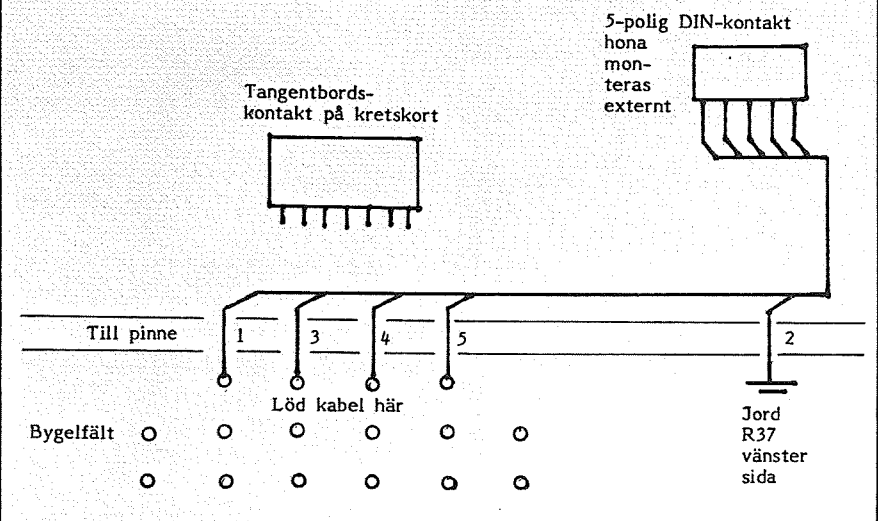
Nu är det dags att montera in prommarna och koppla ihop det hela enligt skissen nedan.

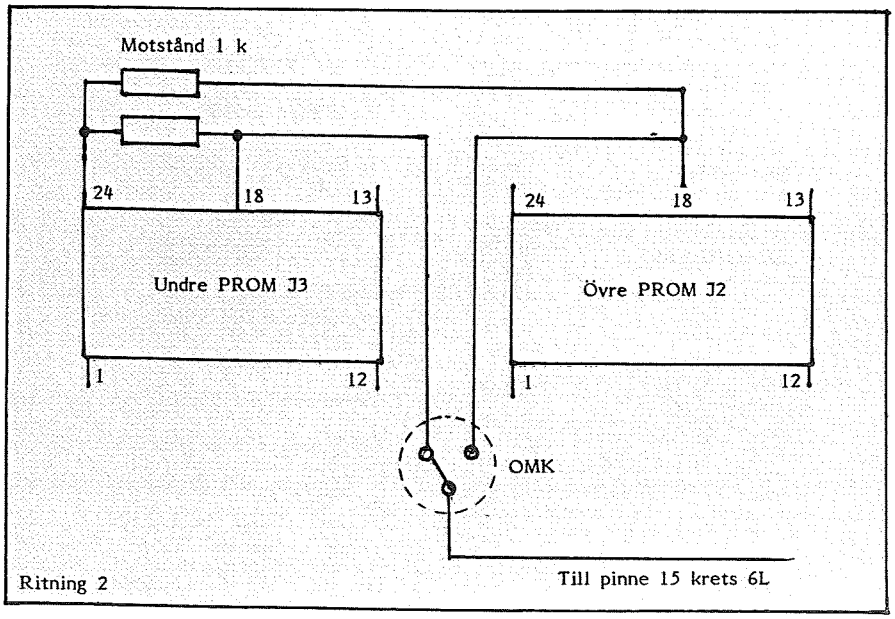
Var noga när Du löder direkt på kretsarna så att Du inte får kortslutningar eller värmer för mycket.

Nu kan Du själv välja vilket optionsprom Du vill använda med hjälp av omkopplaren. Du måste dock trycka RESET varje gång Du slår om den, eftersom optionsprommet måste initieras på nytt när Du har gjort en omkoppling.

<746>
Mats Knuts

Ritning 1





Ritning 2

ABC80

Så här bär jag mig inte åt.

Ett hemma-hos-mig-själv-reportage av Bernt Figaro (412) med bigga glimten i ögat och fjällfiske norr polcirkeln i blick !

Beträffande vem som gjort den knepiga rad 10 i TVMAIN får jag säga som datorn i spelprogrammet BUTTON-BUTTON: Who ıe? I don't have it ! Däremot har jag äst i pressen att STRÄNG fått en hög orden. Varför får inte mina strängar en hög ABC-orden?

A) Kan man flytta programdelar i BASIC till en plats utanför programmet och vice versa?

B) Kan man byta ut en symboltabell i rogrammet mot en egen?

C) Kan man skapa ett nytt program utanför det ordinarie programmets slut (efter HEAP)?

Under vissa förutsättningar svarar jag tveklöst JA på samtliga frågor ! En förutsättning är att man inte försöker flytta SUPERSMARTAID till ledigt utrymme. Det går inte på detta sätt, för såna program används LAZYMAN.SPC. Jag skall nu presentera varför jag svarar JA på frågan A. De två andra svaren kommer någon gång när jag får tid. För att inte förlora för mycket text får du hänga med från början. Jag tänker helt öppet demonstrera vad många av de stora pojkarna kan eller vet om variabelmanipulering. För detta ändamål presenterar jag därför ett demonstrationsprogram om strängens överlägsenhet när det gäller ABC80-programmering. Håll till godo och så även med mina efterföljande kommentarer.

ABC80 har en del viktiga adresser som varje användare bör känna till:

1) BOFA 65052,pekare programstart
2) EOFA 65054,pekare programslut
3) HEAP 65056,pekare variabellistans slut
4) VARIABELROT 65065, pekare till symboltabellen.(variabler m m)

Vi kan med POKE ändra innehållet i essa adresser. Observera att den efterföljande adressen,t ex 65053 tillhör den föregående 65052, eftersom varje adress upptar två byte. I mitt program MANIPUL skall vi skoja friskt med variabelrotens innehåll.

I rad 60 dimensioneras strängen A$ till längden 128 bytes. På rad 70 hämtas adressen till var strängen börjar, således 6 bytes efter innehållet på adresserna 65065-65066. Var denna adress ligger är ointressant, men vill vi veta det, kan vi lägga in en rad så att vi kan printa A%.

I adressen A%+2% läggs strängens längd vid behov in, låg byte och hög byte. Låg <256,hög >255. I rad 80 pokar vi i denna adress, adressen till skärmens första punkt, dvs CUR(0%,0%). Adresslistan till skärmen börjar på adress 884-85. I rad 90 får strängen ett innehåll och eftersom vi nu talat om för ABC80 att lagringsplatsen för A$ är skärmens första punkt, hamnar följaktligen A$ på skärmen. I rad 100 så ändras strängens innehåll. Men strängen inleds med en GET-sats,och det händer inget förran en tangent trycks ner. Vad som händer med denna tangentnedtryckning samlas upp i I%. Efter tangentnedtryckningen blir A$=CHR$(2%)+SPACE$(LEN(A$)).

Vad händer? Jo,texten suddas ut på skärmen. Om svarsalternativet är J/N inträffar det att A$ endera blir innehållet i rad 110 eller i rad 120. Om inget av dessa alternativ inträffar så skrivs inget mera ut förrän längre fram. I rad 130 får strängen B$ ett innehåll. Rad 140 räknar ut adressen till rad 130. Rad 150 räknar ut LEN(B$). Rad 160 räknar ut adressen till var strängen börjar. I rad 170 flyttar vi adressen för A$ att bli lika med B$-strängens början. Samtidigt matar vi in len(B$) och tillfogar dessutom 0%. (Höga byten eller 256%*0%). På rad 180 skrivs A$ ut och se:A$=B$. På rad 190 ligger en definition till skärmadresserna. Raderna 200-240 innehåller data.

Rad 250 ger en uppmaning. Den inleds med en text och sedan förväntas en tangentnedtryckning och de två efterföljande CHR$(8%,32%) tar bort eventuellt tecken till följd av CALL-anropet. Rad 260 släcker skärmen och sedan är det dags med en ny manipulering.

Rad 270-300 utför följande: I loopen pokas adressen till skärmens inledande 5 rader. 5 datavärden inläses till A$ och därefter får A$ ett grafiktecken som inleder stränginnehållet. Allt detta hamnar på skärmen och sedan upprepas uppmaningen att trycka på en tangent. Rad 10-50 innehåller REM-rader med lika många kryss. Räkna rätt !

I rad 320-400 beräknar vi adressen till dessa remrader. Vi flyttar innehållet på skärmen till remraderna. Är innehållet i REM-raderna för korta hamnar bara en del av bilden i remraderna. När man flyttar en sträng från en adress till annan måste man ha en slasksträng och här får B$ gripa in. När vi pokat in den nya adressen så återför vi B$ till A$. Rad 410 suddar ut innehållet på CUR(6%,0%) rad 430-450 skriver ut strängen A$ följt av en uppmaning. När en tangent tryckts ner ändras A% attpeka på adress 65408. Till strängen A$ lagras en liten assemblersnutt från ABC-bladet, en text samt en terminator, CHR$(13%)= textslutsmarkering.

Avslutande rad 480 anropar sedan adress 65408 som i själva verket är ett kommando till ABC80:LIST 10-50 och det är vad som sker som avslutning.

OBS ! Slaskströngen bör dimensioneras = den ordinarie, om längden överstiger 80 tecken för den ordinarie. Och då bör det mesta var förklarat.

De lyckliga innehavarna av en SUPER-SMARTAID (inte helt perfekt den heller) kan göra det enkelt för sig genom att definiera en KEY-funktion:

```
KEY_J="POKE PEEK(884)+256*peek(885)+
7,23_M"
```

Därefter rensas skärmen med CTRL-L.

Man trycker på CTRL-J, går längst upp på skärmen och skriver: 10 REM samt för högerpilen till läget efter det svarta hålet /CHR$(23)/ och så lägger man in sin grafik. Efter RETURN går man upp, ändrar 10 till 20 och upprepar tills man är klar.

Vill man flytta CHR$(23) till annan plats använder man CTRL-D och CTRL-P.

Vill man lägga in CHR$(1) för att använda små bokstäver eller rätta ASCII-tecken gör man en ny definition med beräkning var tecknet skall ligga.

Programmet ovan är avlusat, men texten står och faller med läsarens kunskapsbehov.

```
 10 REM XXXXXXXXXXXXXXXXXXXXXXXXX
 20 REM XXXXXXXXXXXXXXXXXXXXXXXXX
 30 REM XXXXXXXXXXXXXXXXXXXXXXXXX
 40 REM XXXXXXXXXXXXXXXXXXXXXXXXX
 50 REM XXXXXXXXXXXXXXXXXXXXXXXXX
 60 DIM A$=128%,B$=128% : ; CHR$(12%)
 70 A%=PEEK(65065%)+PEEK(65066%)*256%+6%
 80 POKE A%,PEEK(884),PEEK(885%)
 90 A$="Jag är väl duktig, eller hur ?"
100 A$=CHR$(CALL(2%))+SPACE$(LEN(A$)) :
    I%=INP(56%) AND 95%
110 IF I%=74% A$="JA ! Tack skall Du ha
    för det !"
120 IF I%=78% A$="NEJ! Typiskt för dig,
    det där !"
130 B$="Uppvisningen fortsätter !"
140 Z%=CALL(3897%,130%) : REM Adress til
    l rad 130
150 X%=PEEK(Z%)-14% : REM LEN (B$),dispo
    nibel längd
160 Z%=Z%+12% : REM Adress till strängen
    s början
170 POKE A%,Z%,SWAP%(Z%),X%,0% : REM X%
    <255%, höga byten = 0%
180 ; A$ : REM Nedan en definition för b
    ildskärmadresser
190 DEFFNS%(I%,J%)=PEEK(884%+2%*I%)+256%
    *PEEK(885%+2*I%)+J%
200 DATA "7$$$$$$$$$$$$$$$$$$$$$$$k"
210 DATA "5                        j"
220 DATA "5 DETTA ÄR FANTASTISKT j"
230 DATA "5                        j"
240 DATA "uppppppppppppppppppppppz"
250 ; CUR(6%,0%)' Tryck på en tangent 'C
    ALL(2%)CHR$(8%,32%)
260 ; CHR$(12%)
270 FOR I%=0% TO 4%
280 POKE A%,FNS%(I%,0%),SWAP%(FNS%(I%,0%
    )) : REM Skärmrad 0-4
290 READ A$ : A$=CHR$(23%)+A$
300 NEXT I%
310 ; CUR(6%,0%)' Tryck på en tangent 'C
    ALL(2%)CHR$(8%,32%)
320 FOR I%=1% TO 5%
330 Z%=CALL(3897,I%*10%) : REM Adresser
    till rad 10-50
340 X%=PEEK(Z%)-6% : REM Utrymme att man
    ipulera
350 Z%=Z%+5% : REM Adress till start för
    manipulering
360 POKE A%,FNS%(I%-1%,0%),SWAP%(FNS%(I%
    -1%,0%)),X%
370 B$=A$ : REM LAGRA A$ i B$
380 POKE A%,Z%,SWAP%(Z%),X%,0% : REM Fly
    tta A$
390 A$=B$ : REM A$ återfår sitt rätta in
    nehåll
400 NEXT I%
410 POKE A%,FNS%(6%,0%),SWAP%(FNS%(6%,0%
    )) : A$=SPACE$(30%)
420 ; CUR(16%,0%); : REM Ej 'ABC80' i bi
    lden.
430 POKE A%,FNS%(8%,0%),SWAP%(FNS%(8%,0%
    ))
440 A$=' Klappat och klart !'
450 ; CUR(10%,0%)' Tryck på en tangent '
    CALL(2%)CHR$(12%)
460 POKE A%,128%,255%
470 A$=CHR$(213%)+CHR$(33%,133%,255%,201
    %)+'10,50'+CHR$(13%)
480 Z%=CALL(65408%,2799%)
```

SNABBREFERAT.

från årsmötet 1985-02-16.

Här tar vi bara med de viktigare besluten och hur diskussionen gick kommer i en utförligare rapport i nästa nummer av ABC-bladet.

Årsmötet hölls sedvanligt i Brommasalen, Gustavlundsvägen 168 i Bromma. Det är i samma hus som klubblokalen ligger, där den nya monitorn visades upp. Som vanligt hade leverantörer med anknytning till ABC-datorerna bjudits in att visa sina produkter.

Årsmötesförhandlingarna började kl 13.00 med att ordförande Gunnar Tidner hälsade alla välkomna och läste upp en hälsning från hedersmedlemmarna. Torbjörn Alm valdes till mötesordförande och Ulf Sjöstrand mötessekreterare. Till justeringsmän och rösträknare valdes Kalle Lindström och Tom Sjöberg. I och med att årsmötet hade utlyst i ABC-bladet nr 4, 1984 ansågs att mötet var behörigt utlyst. Mötet beslöt att fastställa styrelsens förslag till dagordning för årsmötet. Man beslöt också att godkänna verksamhetsberättelsen för 1984. Även vinst- och förlusträkningen för 1984 projektredovisningen för 1984 godkändes. Revisorerna föredrog revsionsberättelsen och tillstyrkte att styrelsen beviljas ansvarsfrihet för år 1984 vilket också mötet beviljade.

Mötet beslöt att fastställa balansräkningen och att årets underskott på 39:- balanseras i ny räkning för verksamhetsåret 1985. Årsmötet beslöt att fastställa styrelsens beslut om medelmsavgift för 1985 till 140 Skr för seniorer och 80 Skr för juniorer.

Årsmötet beslöt att ge styrelsens mandat att med ett tak av 160 kronor för senior och 100 kronor för juniorer besluta om en preliminär medlemsavgift för 1986.

Årsmötet beslöt att fastställa budgeten för 1985 enligt styrelsens förslag.

Mötet beslöt att till ordförande för ABC-klubben under 1985 välja Stig Löfgren, och att till vice ordförande välja Bo Kullmar.

Vidare beslöts att till ledamöter i ABC-klubbens styrelse för 1985 välja följande:

Magnus Hedner
Jan Holmberg
Jan Liebe-Harcot
Sigvard Nilsson
Tom Sjöberg
Björn Sjöborg

och som suppleanter välja:
Kjell Brealt
Terry Engström
Arne Hartelius

Till revisorer för ABC-klubben för 1985 välja Kjell Järbin och Kalle Lindström. Beslöts även att revisorerna har mandat att inkalla godkänd revisor. Vidare beslöts att till revisorssuppleant för ABC-klubben för 1985 välja Erik Nilsson. Beslöts att välja Göran Sundqvist, Gunnar Tidner och Joe Johnson till valberedning för ABC-klubben för 1985.

Beträffande motion av Lasse Frej angående medlemsmatrikel beslöts att med styrelsens yttrande lägga motionen till handlingarna. Därifrangår att styrelsen har tagit fram en medlemsmatrikel som skall komma under våren.

Beträffande motion från ABC-Väst och ABC-Öst angående röstning med fullmakt beslöts efter diskussion att antaga styrelsen förslag.

Där föreslås att ABC-klubbens stadga § 4 ändras till följande lydelse:

---

§ 4. RÖSTRÄTT Såväl aktiv medlem som hedersmedlem har rösträtt med en röst vid årsmöte och extra föreningsmöte. Röstning med fullmakt är tillåten endast i nedan angivna fall.

Röstning med fullmakt är tillåten för ombud för lokalavdelningar i region som ej omfattar den ort där mötet hålles.

Varje ombud för en lokalavdelning får representera högst 10 röster avgivna med fullmakt från medlemmar i lokalavdelningen.

Tillsammans får ombuden från lokalavdelningen inte utöva rösträtt för mer än 15 % av totala antalet röster på mötet.

samt att i § 8. ÅRSMÖTE görs ett tillägg av ny punkt

5. Justering av röstlängd

och numreringen av tidigare punkter 5 till 17 ökas med 1.

Beträffande motion från ABC-Öst angående familjemedlemskap beslöts efter diskussion att avslå motionen.

Som slutvinjett avtackades Gunnar Tidner, som avgående ordförande av Bo Kullmar, som också meddelade att ABC-styrelsen hade utsett Gunnar Tidner till hedersmedlem i ABC-klubben.

DIV:

Kansliet

Berit Gustavii, som sköter vårt kansli får ta emot rätt många samtal. I och för sig är det bra att vi har någon som kan svara i telefon, men jag är rädd för att hon kan få sitta i telefon hela dagarna. Det blir dyrt för oss och i sista hand för dej som medlem!

Har du några frågor av teknisk karaktär så ring i första hand någon återförsäljare, Luxor eller DIAB. Gäller det klubbfrågor så kan du ringa direkt till någon funktionär, för telefonnummer se medlemsmatrikeln eller telefonkatalogen. Rör din fråga medlemsregistret eller beställningar så skall du dock ringa Berit direkt.

Berit har nu börjat att studera vid sidan om sitt arbete. Det innebär att hon inte kan svara i telefon så ofta på dagtid. Du kan dock lämna ett meddelande till telefonsvaren med namn och telefonnumer så ringer Berit upp senare.

Ring alltså inte i onödan till Berit Gustavii! Måste du ringa så får du vara medveten om att hon inte har tid att tala i telefon hur länge som helst!

Bo Kullmar

MEM: på ABC802

Luxor skickar numera med BASIC-manualen till ABC802 ett förtydligande om MEM:.

Jag citerar:

"Enheten MEM: använder DOS-buffert 7 för överföring av data. Detta medför att då enheten MEM: används, kan 6 filer samtidigt vara öppna mot flexskivan. För enheten MEM: gäller dessutom att endast en fil kan vara öppen."

Så långt Luxor. Tilläggas kan här att MEM: är en form av RAM-floppy utan bibliotek. Man får själv hålla reda på i vilka av de 128 sektorerna som man sparar information. Vill man spara en programfil i sektor 10 och framåt så skriver man SAVE MEM:10. Om man bara skriver SAVE MEM: så underförstås MEM:0. Drivrutinen till MEM: ligger i prom och tar därför inte något minne i anspråk. Drivrutinen utnyttjar inte DOS:et, diskoperativsystemet.

Man kan även utnyttja extraminnet för att köra all systemprogramvara från RAM-minne, se Benny Löfgrens artikel i ABC-Bladet nr 4 1984. Detta minne kan också användas om man vill köra CPM på 802:an.

Vi använder MEM: för temporära filer i klubbens monitorsystem.

Bo Kullmar

Registerhantering med strukturerad programmering

Fortsätt med struktur är en direkt fortsättning på den mycket populära boken Börja med struktur, den första elementära läroboken om hur man ritar data- och programstruktur enligt JSP-tekniken.

Fortsätt med struktur behandlar filhanteringsproblem och användning av JSP-tekniken.

I boken tas upp filhanteringsbegreppen, uppdatering av sekvensiella och direkta filer, olika typer av listor samt formulärteknik. Dessutom förklaras enkelt och åskådligt hur felhanteringsproblem hanteras i s k antagsselektioner eller "BackTracing".

Kapitlen om sortering visar speciellt påtagligt JSP-teknikens logiska renhet i såväl strukturer som kod. JSP-metodiken och registerhanteringen utgör en god grund för att fortsätta med studier i systemering.

Efter genomgång av Fortsätt med struktur utgör boken ett uppslagsverk över hur de vanligaste registerproblemen struktureras. Läsaren kan senare lätt övergå till egna problem eller andra programmeringsspråk.

Till boken tillhandahåller Liber flexskiva, som innehåller alla de program och register som boken behandlar.

Ca-priset anges inkl moms till 146:- resp 1 659:- om flexskiva är med. För ytterligare information kontakta:
LIBER/Franz Smidek
08-739 93 92.

BASIC II boken för ABC806

Liber har gett ut en version av BASIC II boken för ABC806. Författare är Jan Lundgren och Sören Tornell. Sedan tidigare finns boken också för ABC802. De olika versionerna av BASIC II boken tar hänsyn till de små skillnader som finns mellan ABC800, ABC802 och ABC806.

För ABC806 boken gäller det främst att enheten RAM: nämns och att grafikavsnittet är anpassat till ABC806.

BASIC II böckerna kan rekommenderas som nybörjarböcker. Man kan få programexemplen på skiva.

Bo Kullmar

SUBRUTINER!

MINIHARD

MINIHARD är en liten subrutin avsedd för ABC800/806 och dumpar hela eller delar skärmen på skrivare eller fil. Dumpningen sker radvis och före subrutinanropet anges första och sista rad som skall dumpas.

Så här fungerar den:

Åttio mellanslag läggs först in i variabeln P$. Därefter "kopieras" en ad från bildskärmen in i P$ genom MID$-instruktionen och läsning av teckenminnet. P$ skrivs ut, och nästa rad på skärmen "kopieras" och skrivs ut, osv.

Fördelen med detta är att vi kan bestämma vilka rader som skall dumpas från skärmen. Radvariablerna R1 och R2 anger första och sista rad som dumpningen skall omfatta. I testprogrammet dumpas hela skärmen (24 rader) till Epson-MX, men för att överföra skärmen till en sekventiell fil kan man byta rad 110 mot exempelvis PREPARE "FIL.HCY" AS FILE 8

Den som är intresserad kan själv försöka modifiera rutinen så att även kolumnerna begränsas.

<5316>
Hans Bergman
Linköping

ABC800

```
100 REM Testprogram
110 REM
120 OPEN "PR:VSA30C72" ASFILE 8 :
    REM Epson
130 R1=0 : R2=23 : GOSUB 350
140 END
    REM
160 REM Slut testprogram
170 REM ================================
200 REM MINIHARD.BAS
210 REM
220 REM MINIHARD läser bildminnet
230 REM i ABC800/806 och dumpar
240 REM hela eller delar av skärmen
250 REM på skrivare eller fil
260 REM Hans Bergman <5316>
270 REM ================================
280 REM R1 : första radnummer (0 - 23)
290 REM R2 : sista radnummer  (0 - 23)
300 REM A  : adressvariabel
310 REM I  : loopvariabel
320 REM P$ : radsträng
330 REM ================================
340 REM
350 A=30720+80*R1 : P$=SPACE$(80)
360 FOR I=1 TO 80
370 MID$(P$,I,1)=CHR$(PEEK(A)) : A=A+1
380 NEXT I : ; §8 P4
390 IF A<30720+80*R2 THEN 360
400 RETURN
410 REM
420 REM ================================
```

ABC800

CTRC-Kretsen.

Cursorn på ABC800

CRTC-kretsen 6845 är en programmerbar controller-krets som används för bland annat generering av horisontala och vertikala synkpulser till bildskärmen. För att denna skall fungera måste den programmeras. Den har 16 register som man måste lägga in värden i.

CTRCn är kopplad som en intern I/O enhet, de portadresser som används är 49, 56 och 57. För att ange vilket register man vill läsa/skriva i gör man "OUT 56,reg", sedan kan man skriva i reg med "OUT 57,data". "INP(49)" läser av värdet i adresserat register. OBS, man kan bara läsa av register R12-R17 och skriva i register R0-R15

Tabell 1

Det som jag kommer att beskriva här är hur cursorn programmeras för de olika moderna som man kan sätta cursorn. Vi tar och börjar med att se hur cursorns utseende och 'synlighet' programmeras.

Registren som innehåller information om cursorn är R10 (Cursor start register),R11 (Cursor end register). R10 kan vara från 0-9 och anger i vilken rad cursorn skall börja, rad i teckenmatrisen. R11 anger då i vilken rad den skall sluta. För att tex få gammal 'ABC80' cursor lägger man 0 i R10 och 9 i R11. I ABC806 läggs 8 i båda registren, på ABC802 lär det vara 9. Om man,för att ta ett ytterligare ex,vill indikera i någon egenhändigt gjord INPUT-rutin att man är i Insertmode, kan man lägga ut 8 i R10 och 1 i R11 vilket innebär att man får en cursor som består av 2 rader längst ner och 2 rader längst upp.

Nu litet om hur hastigheten på cursorn samt om den överhuvudtaget skall finnas på skärmen. Detta styrs med hjälp av bit 5,6 i R10 (samma register som används för att välja start rad för cursorn,se ovan). Om vi studerar tabellen nedan förstår vi hur det ligger till.

Tabell 2

Det som är initierat på ABC806 är att bit 6 är satt och cursorn startar/slutar på rad 8 (Underline).

För att sedan ange var någonstans på skärmen cursorn skall placeras används register R14 och R15. Där R15 låga adressbyten och R14 den höga adressbyten,R14 är för övrigt bara på 6 bitar. Adressen skall vara den normala adressen till bildminnet,alltså 30720-32640 (-32720), det senare om man har 25 rader.

Vad har man då för nytta av att kunna detta? Jo, vill man ha fram cursorn i tex ett terminalprogramm så vet man hur man gör. I ett terminal programm kan man ju inte använda GET (Q$),trycker man inte tangent 'fastnar' man ju i programmet och text från V24 går till spillo. Man måste även notera att om man själv har satt på cursorn så får man inte använda GET då denna rutin stänger av cursorn.

Jag hoppas att detta är någon till hjälp. Ni får gärna skicka ett brev på ABC-monitorn, PermoBas om det är något ni undrar. Vanliga brev går också bra. Jag skall försöka ge ett svar på det ni undrar.

<5455>
Patric Ljung
013-131482

Tabell 1

	Register	Register file	Program unit
	R0	Horizontal total	Char.
	R1	Horizontal displayed	Char.
	R2	H. sync position	Char.
	R3	Sync width	-
	R4	Vertical total	Char. row
	R5	V. total adjust	Scan line
	R6	Vertical displayed	Char. row
	R7	V. sync position	Char. row
	R8	Interlace mode & skew	-
	R9	Max scan line address	Scan line
->	R10	Cursor start	Scan line
->	R11	Cursor end	Scan line
	R12	Start address (H)	-
	R13	Start address (L)	-
->	R14	Cursor (H)	-
->	R15	Cursor (L)	-
	R16	Light pen (H)	-
	R17	Light pen (L)	-

Tabell 2

Bit 6	Bit 5	Cursor mod
0	0	Fast cursor
0	1	Ingen cursor på bildskärmen
1	0	Blinkande, 1/16 field rate
1	1	--"-- , 1/32 field rate

Om makrokommandon i

TV-editorn.

ABC-klubbens ordbehandlingsprogram

Texthanteringen med TV-editorn blir åtskilligt kraftfullare om man kan en del om sammansatta makrokommandon.

På ABC-kassett 3 fanns den utomordentliga textfilen TVEDIT.REM, med instruktioner och exempel på hur man använder TV-editorn. Där framgår även att man kan definiera en kommandosträng och sedan utföra den med ;M$$.

Ganska snart kom jag på att man kan sätta vänstermarginal genom att definiera *I $$ med så många mellanslag som önskas och sedan göra ;M på varje rad. Men hur gör man för att få kommandot att påverka samtliga rader i texten?

På en träff med ABC-Stockholm var det någon som talade om rekursiva makrokommandon. Jag begrep ingenting då men blev ganska nyfiken. Av en händelse hamnade jag i höstas hemma hos en medlem som gav mig några ledtrådar.

Editorns möjligheter skall naturligtvis inte jämföras med en basicrutin, men kan ändå fås att utföra en hel del nyttigheter.

De "funktioner" som kan användas är editorns egna:

nC för att flytta cursorns n steg åt höger.
nD ta bort n tecken från cursorn räknat.
S<text> sök text
R<gammal text>$<ny text>$$ byt text
I inmatning av text
nI sätt in tecken med asciivärdet n
etc. etc.

Det gäller bara att kombinera dem på ett vettigt sätt.

Hur ser då en textrad egentligen ut,och hur gör man ett kommando rekursivt (dvs så att det anropar sig självt) och därigenom påverkar hela texten i bufferten? I basic slutar en textrad med de 'osynliga' tecknen ASCII 13 - CTRL M (return) och ASCII 10 - CTRL J (linefeed).

I det följande visas några exempel med kommentarer som just bygger på detta.

I exemplen är:
* editorns promter
M inleder alla makrokommandon
I =insättning
$ står för högerpilen.

$$ " i exemplen innebär två tryckningar på högerpilen. Jag har konsekvent använt $ som avgränsare i kommandosträngarna (vilket inte alltid är nödvändigt).

Vid sökning med kommandot S<text>, söker editorn upp texten<text> och ställer cursorn efter texten. När editorn t.ex funnit tecknet för linefeed står cursorn därför i själva verket i början på nästa rad.

Rekursiviteten erhålls genom det ;M som avslutar kommandosträngen och som får kommandot att upprepa sig självt tills det inte finns någon text kvar i bufferten att bearbeta.

I TV-editorn kan de "osynliga" tecknen ASCII 0 till 32 matas in från tangentbordet på två olika sätt.

a) Som *Ictrl+en bokstav$$
b) Eller *Ascii-nrIascii-nrIascii-nrI$$

I båda fallen visas tecknen på skärmen som ^ för ctrl åtföljt av bokstaven alternativt den bokstav som svarar mot inmatat ascii-nr. Det här gör det möjligt att skriva in t.ex styrkoder till skrivaren i texten. Koderna visas på skärmen, men kommer självfallet inte med som synliga tecken på utskriften. En komplett beskrivning av tangentbordets koder finns i den gamla bruksanvisningen till ABC80. Jag har samlat de "osynliga" ASCII-koderna i en fil som sänts till ABC-monitorn i Stockholm under namn CTLKOD.TXT.

Här följer några exempel:

1) Lägg in 5 blanktecken i början på alla rader. *MI $SüJ$;M$$ (sätt in 5 blanka, sök linefeed, börja om)

2) Ta bort tecken som satts in med 1) *M5D$SüJ$;M$$ (ta bort 5 tecken, sök linefeed, börja om)

3) Sätt in blanktecken i slutet på en rad *MRüM$ üM$;M$$ (byt ut return mot fyra blanktecken och return, börja om)

4) Lägg in blankrad mellan varje rad *MSüJ$I3I10I$;M$$ (sök läget för linefeed, sätt in return och linefeed, börja om)

5) Sätt in fyra * på position 4 *M4C$I*** **$SüJ$;M$$ (gå in fyra tecken på raden, sätt in 4 *, sök linefeed, börja om)

6) Ta bort tecken som satts in med 5) *M4C$4D$SüJ$;M$$ (gå in fyra tecken på raden, ta bort fyra tecken, sök linefeed, börja om)

Kommandona utförs som vanligt med ;M$$.

Om man har några tusen tecken inne så kan det ta ett antal sekunder att få ett kommando utfört. Men det fungerar.

Är det någon som lyckats med att få ett kommando att arbeta på ett bestämt avsnitt av texten, från rad - rad? Hör av Er i så fall.

Sandgren <2776>

TV-editorn.

ABC-klubbens ordbehandlingsprogram

Ett av de mest intressanta programmen som vi medlemmar fått tillgång till genom ABC-klubben är TV-editorn. Heder åt alla som medverkat till den. Alla hittillsvarande revisioner till trots finns det fortfarande önskemål angående programmets egenskaper och funktion som väntar på att uppfyllas. Denna sak skall inte uppfattas som kritik mot de som arbetat med editorn. Deras insats uppskattas mycket av många däribland av mig själv.

I stället för att här presentera mina små försök till modifiering vill jag föreslå ett samarbete med alla som intresserar sig för programmet. Den som har gjort nya versioner, nya programmrader eller har förslag på ändringar är välkomna att höra av sig med brev, kasetter eller telefon.

Genom egen användning av programmet och kontakter med andra användare har jag samlat en del synpunkter som jag vill framföra.

1: Antalet tecken och markörens pos. kunde tillsammans med annan status information presenteras på rad 23. Detta kunde förnyas efter varje gång ett kommando utförts.
2: Om en fil saknar CR så försvinner slutet vid inläsning. Detta borde vara en sak för programmet att kolla och vid behov lägga till.
3: Samma gäller för långa rader. Om dessa blev kontrollerade i förväg undgår man risken att förlora det som finns efter den för långa raden.
4: Möjlighet att välja drive. Annars får man laborera med öppna och stängda luckor, eller trixa på annat sätt.
5: Engelska i all ära men finns det ingen som i likhet med mig tycker att dotorn kunde få tala vårt språk.
6: För att få plats med finesser och stora texter borde REM-satserna tas undan, eventuellt till ett eget instruktionsprogramm.
7: TV-editorn och klubbens medlemmar förtjänar omfattande instruktioner. Nu krävs det nästan att man lusläser alla ABC-blad.
8: Nu finns olika versioner för kasett och disk. Kanske kan de förenas, till till nytta t.ex. för den som vill ta ut programmet från skiva men lägga sin text på kasett.

Det finns säkert andra som har id'er om hur programmet skall kunna bli mer fulländat och användarvänligt. Alla är välkomna att höra av sig. Jag vill nämna att jag har en ABC-80 med 32 kB, checksumma 11273, FD2, och bandspelare. Ingen egen printer. Någon ägare av en dylik kanske är villig att göra en insats med utprovning av de funktionerna. Säkert fordras flera medverkan för att nå resultat. Det är min förhoppning att detta skall resultera i en el. flera nya versioner av TV-editorn och att det skall kunna bli en bra instruktion för den. Som kan komma medlemmarna till del inom en inte alltför avlägsen framtid.

Referenser: (ABC-bladet år:nr sida)
81:4-5 26 TV skärmeditor Gunnar Tidner
82:1 6-7 TV-editorn G. Tidner
82:2 6-7 Mera om TV-editorn G. Tidner
82:3 14 Innehållet i kasett 4
83:2 21 Radlängdskontroll G. Tidner
83:4 25 DIV (tom fil) Stig Berlin
83:4 25 TVVMAIN.800 Bo Kullmar
84:2 34 List-run-rad 10 Lars Frej
84:2 36-37 Rad 10 Sigvard Johansson
84:4 19 Radjustering L Lundström

Göran Engström <3457>
Box 200
685 00 Torsby
Tel 0560/10702

TERM100.DOC

TERM100.DOC

1984-10-15

David Andersson

Terapivägen 14 A
141 56 HUDDINGE
Telefon 08-711 74 34

Terminalprogram för ABC80

Beskrivning av TERM100 Version 1.4

Innehåller kapitel:

KARAKTÄRISTIK

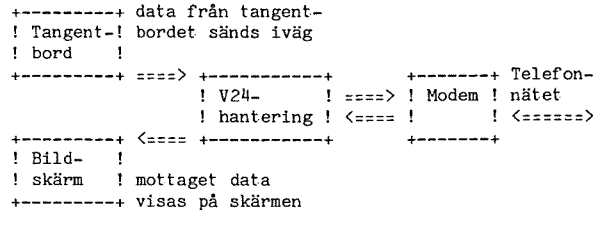
Seriellt gränssnitt. Använder V24-utgången på ABC80.

Emulerar VT100. Hanterar de vanligaste kommandon som DEC VT100-terminaler använder (ANSI kommandon). Terminalprogrammet har även en VT52-kompatibel mode.

Möjlighet att **dumpa bildskärmens** innehåll på skrivare. Möjlighet att spara mottagen text på fil samt sända fil.

Simulerar 80-teckens bildskärm om man har 40 tecken på ABC80.

Mycket schematisk bild av terminalprogrammet:



KOMMUNIKATIONSDATA

Seriellt gränssnitt V24 (RS232C).
Baudrate från 50 bit/s till 1200 bit/s.
Flödesstyrning med XON/XOFF protokoll.

Inställning av mottagningshastighet och sändningshastighet separat (split speed).

50 bit/s
75
110
134
150
200
300
600
1200

7 bit överföring utan paritetsbit
8 bit överföring där 8:de biten kan vara:
  Paritetsbit för udda paritet
  Paritetsbit för jämn paritet
  8 bits data där 8:de biten alltid är 0
  8 bits data där 8:de biten alltid är 1

Alltid en startbit och en stoppbit.
Ingen paritetskontroll på mottaget data.

BILDSKÄRM

En virtuell terminal-bildskärm på 24 rader om 80 tecken. Om ABC80 är utrustad med 80-teckens tillsats så visas hela rader. Om ABC80 har 40-teckens skärm så visas en del av den virtuella terminal-bildskärmen. Automatiskt visas den del av den virtuella terminal-bildskärmen som markören för tillfället befinner sig i. Man kan även styra själv vilken del av den virtuella terminal-bildskärmen som ska synas.

TANGENTBORDET

Nedtryckning av en tangent medför att dess ASCII-kod sänds ut på linjen (över V24 kontakten) under förutsättning att mottagaren (modemet) är redo att ta emot data samt, om XON/XOFF protokoll används, att XOFF inte är mottaget. Nedtryckning av den tangen som är reserverad för keypad-mode medför dock inte att motsvarande ASCII tecken sänds. Denna tangent kan vara CTRL-É (control e-accent), CTRL-Q eller CTRL-Ö.

Keypad mode

Med någon av tangenterna CTRL-É (control E-accent), CTRL-Q och CTRL-Ö sätter man terminalprogrammet i s.k. keypad-mode. Vilken av dessa tangenter som gäller är valbart (default är CTRL-É). I keypad-mode syns texten "KEYPAD" i övre högra hörnet av den reella bildskärmen.

I keypad-mode kan man inte sända de vanliga ASCII-koderna utan tangenterna på tangentbordet får nya funktioner. Bl a kommer en del tangenter att motsvara markörförflyttningstangenterna (piltangenterna) på DEC VT100 terminaler, och vid nedtryckning kommer de att sända en escape-sekvens.

I keypad-mode kan man även styra vilken del av den virtuella terminal-bildskärmen som ska synas, om man har 40-teckens ABC80. Framåtpil- och bakåtpil-tangenterna rullar bildskärmen åt valfritt håll och CTRL-"<" gör att markören alltid syns i den reella bildskärmen.

för ABC80

Med CTRL-É, CTRL-Q eller CTRL-Ö går man ur keypad-mode och alla tangenter får den funktion som är graverade på dem, d.v.s. normal alfanumerisk mode.

Den control-tangent man använder för att gå ur keypad-mode med måste man sedan använda för att gå in i keypad-mode nästa gång.

Tangenten för att gå in i keypad-mode fungerar även som "NO SCROLL" tangent om XON/XOFF flödessyrning används. När man går in i keypad-mode slutar TERM100 att visa mottaget data. Om mer är 63 tecken buffrats upp i TERM100 så sänds XOFF. När man går ur keypad-mode igen eller när man sänder någon escape-sekvens (t.ex. markörförflyttning i keypad-mode) så visas återigen mottagna tecken. Har terminalprogrammet sänt XOFF så sänds nu ett XON tecken för att starta dataflödet igen.

I keypad-mode gäller följande tangenter:

A	Vänsterpil på VT100
S	Högerpil på VT100
W	Uppåtpil på VT100
Z	Nedåtpil på VT100
!	PF1 på VT100
"	PF2 på VT100
§	PF3 på VT100
$	PF4 på VT100
0 .. 9	Motsv. tang. 0 till och med 9 på keypadden på VT100. (Se appendix)
- . ,	Motsv. tangenter på VT100's keypad. (Se appendix)
RETURN	Motsv. tangenten "ENTER" på VT100. (Se appendix)
CTRL-M	Samma som RETURN.
%	Sänder namngivarsträngen. Motsvarar CTRL-BREAK på VT100.
&	Sänder egendefinierad sträng. Definierad i setupmode U. Parameter U6.
/	Sänder egendefinierad sträng. Definierad i setupmode U. Parameter U7.
FRAMÅTPIL	Scrolla den virtuella bilden åt vänster/nedåt. samt stäng av automatik. (gäller ej om man har 80-teckens tillsats)
BAKÅTPIL	Scrolla den virtuella bilden åt höger/uppåt samt stäng av automatik. (gäller ej 80-teckens tillsats)
CTRL-<	Sätt på automatik så att den del av den virtuella bildskärmen som innehåller markören alltid visas. (gäller ej om man har 80-teckens tillsats)
CTRL-É	Gå ur keypad-mode. Man går in i keypad-mode igen med CTRI-É.
CTRL-Q	Gå ur keypad-mode. Man går in i keypad-mode igen med CTRI-Q.
CTRL-Ö	Gå ur keypad-mode. Man går in i keypad-mode igen med CTRI-Ö.
R	Gå in i inställningsmode R (setup R).
T	Gå in i inställningsmode T (setup T).
Y	Gå in i inställningsmode Y (setup Y).
U	Gå in i inställningsmode U (setup U).

Praktiskt taget alla keypad-funktioner kan användas med CTRL-tangenten nedtryckt om man så önskar. T.ex. CTRL-T för setup-mode T.

Inställningsmode (setup)

I keypad-mode betyder R, T, Y och U att man ställer terminal-programmet i inställningsmode (setup-mode). Med mellanslag, CTRL-É, CTRL-Q eller CTRL-Ö går man ur inställningsmode.

I setup R kan följande parametrar ställas in.

1. Ställ in sändningshastighet (baudrate 50..1200).
2. Ställ in mottagningshastighet (baudrate 50..1200).
3. Ställ in antal databitar och paritet.
4. Om XON/XOFF-protokoll (flödesstyrning) ska användas.
5. Om halv eller full duplex ska användas. Halv duplex betyder att terminalen ekar inmatade tecken.
6. Transparent mode. Om alla mottagna styrtecken (CTRL-tecken) ska synas på skärmen. Används t.ex. vid felsökning.

I setup T kan följande parametrar ställas in.

1. Om ANSI- (VT100) eller om VT52-standard ska gälla.
2. Om terminalen automatiskt ska börja skriva på ny rad när man skriver förbi högra kanten. (wraparound)
3. Om RETURN-tangenten ska sända CR+LF och om mottaget LF ska tolkas som CR+LF. (Newline)
4. Om terminalen ska gå i lokal mode eller ej. I lokal mode sänds inte data, inmatade tecken ekas direkt på bildskärmen.
5. Välj om 12 rader om 80 tecken eller 24 rader om 40 tecken ska visas. Om 80-teckens tillsats finns kan full skärm (24 rader om 80 tecken) väljas.
6. Betydelsen av CTRL-< samt bakåtpil och CTRL-H. Är CTRL-< inställd på BS så kommer bakåtpil att sända <DEL> (ASCII=127) och mottagna <DEL> tecken kommer att presenteras som fylld ruta ■.
7. Om tangenterna ska vara repeterande eller ej.

I setup Y kan följande parametrar ställas in.

1. Välj filnamn för att spara mottaget data. En markör visar att man ska skriva ett filnamn. Editera filnamnet med bakåtpil och CTRL-X. Mellanslag, CTRL-É, CTRL-Q och CTRL-Ö avbryter inmatning av filnamn. Med RETURN så skapas filen och skrivning på filen av mottaget data kan börja.
2. Välj fil för att sända data. Filnamn anges på samma sätt som ovan. Med RETURN så öppnas filen och data börjar sändas.
3. Dumpa den virituella bildskärmens innehåll till fil. Filnamn anges på samma sätt som ovan. RETURN skapar filen och startar dumpningen. När dumpning pågår kan inget data mottagas via V24. (Om dumpning sker till skrivare via V24, glöm inte att byta kontakten från modemet till skrivaren).
4. Avbryt sparande på fil eller sändning av fil. Filen stängs.
6. Välj fil-format. Text-format eller data-format. I data-format markeras CR och LF med _M respektive _L. I text-format markeras CR och CR+LF med normalt radslut.

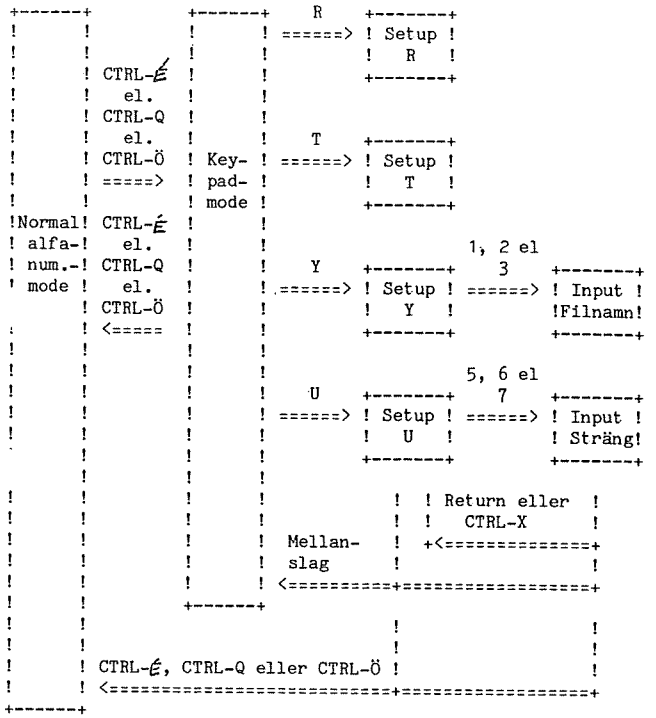
I setup U kan följande parametrar ställas in.

1. Lämna terminalprogrammet. Gå ur TERM100.
2. Återställ alla parametrar till ursprungsvärdet samt rensa terminalens bildminne.
5. Mata in och editera namngivarsträngen (answer back code). Styrtecken anges med understrykningstecken + en bokstav. Mellanslag anges med framåtpil eller CTRL-S. Ta bort tecken med bakåtpil. Avsluta inmatning med RETURN eller CTRL-X.
6. Definiera vad egendefinierad tangent keypad-& ska sända. Editeras på samma sätt som namngivarsträngen.
7. Definiera vad egendefinierad tangent keypad-/ ska sända. Editeras på samma sätt som namngivarsträngen.

Med mellanslag lämnar man inställningsmode (setup-mode) och kommer till vanlig keypad-mode.

Med CTRL-É, CTRL-Q eller CTRL-Ö lämnar man både inställnings mode samt keypad-mode och kommer till normal alfanumerisk mode.

Följande figur försöker beskriva relationerna mellan olika moder i terminalprogrammet. En ruta betyder en mode och pilarna mellan dessa anger hur man kan byta mode.



Filöverföring

I setup-mode Y kan man välja att sända och ta emot data till och från filer. OBS! Spara på en file (setup Y1) och sända en fil (setup Y2) fungerar inte med kassettminnet. En printer med seriellt gränssnitt kan normalt inte heller användas för dessa funktioner. Vid send (setup Y2) så kan filen vara printer om man har parallellgränssnitt, till denna. Vid bildskärmsdump (setup Y3) så kan filen gå till kassett eller printer, även om printern är ansluten via V24. Glöm bara inte att ansluta printern till V24 kontakten innan dumpningen startas.

Funktionen setup Y1, spara mottagen text, gör att det som kommer in från V24 kontakten lagras på en fil (samtidigt som det visas på skärmen som vanligt). Detta fortgår ända tills man avbryter med setup Y4 eller tills t.ex. flex-skivan blir full.

Funktionen setup Y2, sända en fil, gör att varje tecken i filen kommer att skickas, som om man hade skrivit tecken själv på tangentbordet.

Filtypen är enkla sekvensiella filer som t.ex kan hanteras med instruktionerna INPUTLINE § och PRINT § i ABC80-basic.

Om man har valt lagring av data i textformat (setup Y6 = TEXT) när man tar emot och lagrar data på en fil (setup Y1) så kommer alla synliga tecken (ASCII 32 t.o.m. ASC 127) att skrivas ut på filen. CTRL-tecken ignoreras. Carriage return (ASCII 13) ger ny rad i filen. Tabulatortecknet (ASCII 9) görs om till ett lämpligt antal mellanslag på filen. Inga rader tillåts bli längre än 120 tecken på filen. Det är lämpligt att använda textformatet när man t.ex. hämtar och sänder filer från databaser.

Om man har valt lagring av data i dataformat (setup Y6 = DATA) när man tar emot en fil så kommer alla tecken att lagras på filen. CTRL-tecken kommer då att lagras som två tecken, ett understrykningstecken ("_") och en bokstav. Ett mottaget understrykningstecken kommer att lagras som två understrykningstecken om man valt dataformatet. Carriage return lagras som "_M" och ger inte ny rad i filen. Alla rader kommer att bli knappt 120 tecken långa och ett radslut har ingen information om mottagna tecken. Dataformatet används om man vill ha exakt kontroll över vilka tecken man tar emot och sänder iväg från en fil.

Om värddatorn (den dator som man är uppkopplad mot) kan hantera flödesstyrning med XON/XOFF så bör man ställa in detta med setup R4 = YES, för att garantera att TERM100 respektive värddatorn hinner med att skriva data till sekundärminnet. Att ta emot filer i 300 bit/s brukar dock fungera bra utan flödesstyrning.

KOMMANDON:

Mottagna tecken och teckensekvenser som har en bestämd betydelse för terminalprogrammet.

Styrtecken (control character)

Enquire	ENQ	(5)	Begäran om namngivarsträng
Bell	BEL	(7)	Pling i högtalaren
Backspace	BS	(8)	Backa markören ett steg
Horisontal tab	HT	(9)	Flytta markören till nästa tabläge
Line feed	LF	(10)	Flytta markören neråt ett steg
Vertikal tab	VT	(11)	(ger line feed)
Form feed	FF	(12)	(ger line feed)
Carriage return	CR	(13)	Flytta markören till början på raden
Shift out	SO	(14)	Välj designator G1
Shift in	SI	(15)	Välj designator G0
X-ON	DC1	(17)	Starta transmission, flödesstyrning
X-OFF	DC3	(19)	Stoppa transmisson, flödesstyrning
Escape	ESC	(27)	Inleder en escape-sekvens

ANSI-kompatibel mode (VT100)

Parametrarna till följande kommandon är normalt decimala siffror. Escape-sekvenser får innehålla styrtecken (control characters) och mellanslag. Ett escape-tecken avbryter dock en escape-sekvens och börjar en ny.

Markörförflyttningskommandon

Upp	<ESC> Ä<Nr> A
Ner	<ESC> Ä<Nr> B
Höger	<ESC> Ä<Nr> C
Vänster	<ESC> Ä<Nr> D ;
Direkt positionering	<ESC> Ä<Line>; <Column> H
Direkt positionering	<ESC> Ä<Line>; <Column> f
Index (line feed)	<ESC> D
Reverserat index	<ESC> M
Spara markör och attribut	<ESC> 7
Återställ markör och attribut	<ESC> 8

<Nr> = multiplikator (default 1)
<Line> = radposition (default 1)
<Column> = kolumnposition (default 1)

Sätt teckenattribut

När ett attribut är inställt kommer text att skrivas ut med detta attribut tills alla attribut stängs av.

<ESC> Ä <Par> ; <Par> ; ... ; <Par> m

<Par> = 0 stäng av alla attribut (default parameter)
<Par> = 4 sätt på understrykning (*)
<Par> = 5 sätt på blink-attribut
<Par> = 7 sätt på inverterad video (*)

(*) Understrykning och inverterad video visas som blinkande text i ABC80.

Raderings kommandon

Dessa kommandon fyller valda delar av skärmen med tecknet mellanslag.

Från markör till slutet på raden	<ESC> Ä K
Från markör till slutet på raden	<ESC> Ä 0 K
Från början på raden till markör	<ESC> Ä 1 K
Hela raden som markören står på	<ESC> Ä 2 K
Från markören till slutet på skärmen	<ESC> Ä J
Från markören till slutet på skärmen	<ESC> Ä 0 J
Från början av skärmen till markören	<ESC> Ä 1 J
Rensa hela skärmen	<ESC> Ä 2 J

Rapportera markör position

Aktivera rapport med <ESC> Ä 6 n
Svar från terminalen <ESC> Ä <Line> ; <Column> R
<Line> = radposition
<Column> = kolumnposition

Rapportera terminalens tillstånd (device status report)

Aktivera rapport med <ESC> Ä 5 n
Svar från terminalen <ESC> Ä 0 n

Rapportera terminalattribut (device attribute)

Aktivera rapport med <ESC> Ä c
Aktivera rapport med <ESC> Ä 0 c
Svar från terminalen <ESC> Ä ? 1 ; 0 c

Scroll region kommandon

Sätt scroll region <ESC> Ä <Top> ; <Bottom> r
<Top> = övre scroll gräns (default 1)
<Bottom> = nedre scroll gräns (default 24)

Programerbara lysdioder
'Lysdioderna' syns på översta raden på skärmen i keypad-mode.

Styrsekvens <ESC> <Par> ; <Par> ; ... ; <Par> q

<Par> = 0 eller inget	Släck alla lysdioder
<Par> = 1	Tänd L1
<Par> = 2	Tänd L2
<Par> = 3	Tänd L3
<Par> = 4	Tänd L4

Välj tecken uppsättning

Teckenuppsättning	G0 designator	G1 designator
United Kingdom (*)	<ESC> (A	<ESC>) A
United States (*)	<ESC> (B	<ESC>) B
Grafisk teckenuppsättning	<ESC> (0	<ESC>) 0

G0 och G1 designator väljs med styrtecken Shift Out respektive Shift In. (*) United Kingdom = United States = Svensk ASCII i ABC80.

Ställ in tabulator stoppar

Sätt tab i aktuell kolumn	<ESC> H
Ta bort tab i aktuell kolumn	<ESC> Ä g
Ta bort tab i aktuell kolumn	<ESC> Ä 0 g
Ta bort alla tabulator stoppar	<ESC> Ä 3 g

Sätt och reset mode kommandon

	Set	Reset (*)
New line	<ESC> Ä 20 h	<ESC> Ä 20 l
Cursor key mode	<ESC> Ä ? 1 h	<ESC> Ä ? 1 l
VT52 mode	(**)	<ESC> Ä ? 2 l
Origin mode	<ESC> Ä ? 6 h	<ESC> Ä ? 6 l
Wraparound	<ESC> Ä ? 7 h	<ESC> Ä ? 7 l
Auto repeat	<ESC> Ä ? 8 h	<ESC> Ä ? 8 l
Application keypad	<ESC> =	<ESC> >

(*) Sista tecknet i sekvensen är litet L.
(**) Ej tillämpbar. Se ESC < i VT52 mode.

VT52 kompatibel mode

Markör upp	<ESC> A
Markör ner	<ESC> B
Markör höger	<ESC> C
Markör vänster	<ESC> D
Välj grafisk teckenuppsättning	<ESC> F
Välj ASCII	<ESC> G
Flytta kursorn 'home'	<ESC> H
Reverserad line feed	<ESC> I
Radera till slutet på skärmen	<ESC> J
Radera till slutet på raden	<ESC> K
Direkt markör positionering	<ESC> Y <Line> <Column>
Status rapport	<ESC> Z Svar: <ESC> / Z
Sätt applikation keypad	<ESC> =
Reset applikation keypad	<ESC> >
Sätt ANSI mode (VT100)	<ESC> <

<Line> = ett tecken med värdet för raden + 31
<Column> = ett tecken med värdet för kolumnen + 31

Appendix

Restriktioner

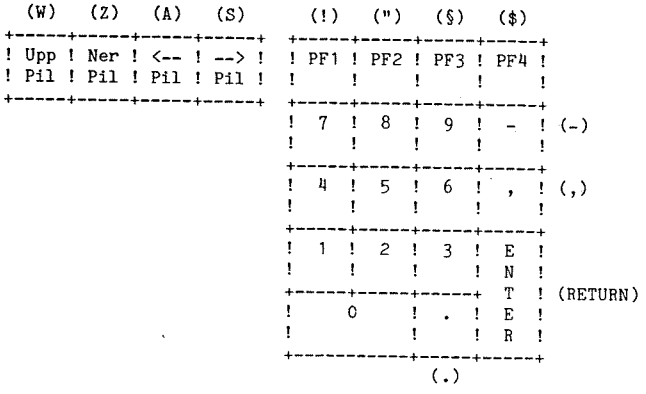
Funktioner som finns i DEC VT100-terminaler men ej i TERM100:
Dubbel teckenhöjd eller dubbel teckenbredd kan inte ställas in.
132 teckens radlängd kan ej ställas in.
Soft scroll kan inte ställas in.
Understrykning och invers video presenteras som blinkande text.
Semigrafisk teckenuppsättningen finns inte men simuleras med "-", "!" och "+" för linjer samt "_" (fylld ruta) för en del andra grafiska tecken.
Initiering av confidence test är inte implementerat.
Begränsad kommunikations hastighet.

Nationella ASCII-tecken

Svensk ASCII	United States	United Kingdom
§ (brädgård, hash)	§ (brädgård, hash)	(pund tecken)
$ (sol)	(dollar)	
E (versal e-accent)	(alfa slang, commersial at)	
Ä (versal ae)	(vänster hak parentes)	
Ö (versal oe)	(vertikalt streck)	
Å (versal aa)	(höger hak parentes)	
U (versal tyskt y)	(tak, uppåtpil, cirkumflex)	
e (gemen e-accent)	(grav accent)	
ä (gemen ae)	(vänster krull parentes)	
ö (gemen oe)	(bakåt slash)	
å (gemen aa)	(höger krull parentes)	
ü (gemen tyskt y)	(tilde, växelström)	
_ (fylld ruta)	<DEL> (ett ej synligt styrtecken)	

Keypadden på VT100

Så här ser keypadden ut på Digital VT100 terminaler. Inom parentes anges motsvarigheten i TERM100 i keypad-mode. Notera att !,";§ och $ är shift av 1,2,3 och 4.



TERM100.LAT

TERM100.LAT 1984-09-26 Lathund för TERM100 Version 1.4

SÄTTA IGÅNG

Startas med RUN TERM100 <RETURN>
CALL-adresser om man gått ur TERM100 temporärt. Kallstart: CALL(53000)
Varmstart: CALL(53003)

Gå ur TERM100 med Keypad-U-1 (CTRL-É U 1)

KEYPAD MODE

Gå in i keypadmode med aktuell keypadknapp (CTRL-É,CTRL-Q eller CTRL-Ö)

		(W) Upp Pil	(Z) Ner Pil	(A) <-- Pil	(S) --> Pil
Shift-5	Sänd namngivarsträng				
Shift-6	Sänd KEY&				
Shift-7	Sänd KEY/				

R	Setup R (Kommunikationsinst.)
T	Setup T (Terminalinst.)
Y	Setup Y (Filhantering)
U	Setup U (Definiernara strängar)
<--	Sidscroll (40 tkn)
-->	Sidscroll (40 tkn)
CTRL-<>	Auto sidscroll (40 tkn)
CTRL-É	Gå ur keypadmode
CTRL-Q	- " -
CTRL-Ö	- " -

(!)	(")	(§)	($)	
PF1	PF2	PF3	PF4	
7	8	9	-	(-)
4	5	6	,	(,)
1	2	3	ENT R	(RETURN)
0		.		(.)

SETUP MODE

R1	Sändhastighet (Transmit speed)
R2	Mottagningshastighet (Recieve speed)
R3	Antal bit/paritet
R4	Flödesstyrnings protokoll XON/XOFF
R5	Full/half duplex
R6	Diagnostik, visa alla tecken
R7	

T1	VT100/VT52
T2	Wraparound
T3	Newline
T4	Local mode
T5	Bildskärm typ
T6	CTRL-<> resp bakåtpil
T7	Repeterande tangenter

U1	Spela in en fil
U2	Sänd en fil
U3	Dumpa bildskärmen till fil
U4	Stäng fil
U5	
U6	Filformat
U7	

RETURN	= Starta filöverföring
CTRL-X,mellanslag	= Ångra
bakåtpil	= ta bort tecken

Y1	Exit, lämna TERM100
Y2	Reset
Y3	
Y4	
Y5	Definiera namngivare
Y6	Definiera KEY&
Y7	Definiera KEY/

RETURN,CTRL-X,mellanslag	= Spara
framåtpil	= Mellanslag
Shift-_	= CTRL-tecken

STYRTECKEN

Decimal	Ascii	Betyder
5	ENQ	skicka namngivare
7	BEL	pling
8	BS	backspace
9	TAB	tab
10	LF	line feed
11	VT	vertikal tab
12	FF	form feed
13	CR	carriage return
14	SO	shift out (sel. G1)
15	SI	shift in (sel. G0)
17	XON	starta flöde
19	XOFF	stoppa flöde
27	ESC	escape

för ABC80

ANSI (VT100)-KOMPATIBEL MODE

Upp	<ESC> Ä <Nr> A		
Ner	<ESC> Ä <Nr> B		
Höger	<ESC> Ä <Nr> C		
Vänster	<ESC> Ä <Nr> D ;		
Direkt positionering	<ESC> Ä <Line> ;	<Column> H	
Direkt positionering	<ESC> Ä <Line> ;	<Column> f	
Index (line feed)	<ESC> D		
Reverserat index	<ESC> M		
Spara markör och attribut	<ESC> 7		
Återställ markör och attribut	<ESC> 8		
Radera till slutet på raden	<ESC> Ä K	eller	<ESC> Ä 0 K
Radera till början på raden	<ESC> Ä 1 K		
Radera hela raden	<ESC> Ä 2 K		
Radera till slutet på skärmen	<ESC> Ä J	eller	<ESC> Ä 0 J
Radera till början på skärmen	<ESC> Ä 1 J		
Radera hela skärmen	<ESC> Ä 2 J		
Rapportera markörposition	<ESC> Ä 6 n	Svar:	<ESC> <Lin> ; <Col> R
Rapportera device status	<ESC> Ä 5 n	Svar:	<ESC> Ä 0 n
Rapportera device attribute	<ESC> Ä c	eller	<ESC> Ä 0 c
		Svar:	<ESC> Ä ? 1 ; 0 c
Sätt teckenattribut	<ESC> Ä <Par> ;		<Par> ; ... ; <Par> m
Sätt scroll region	<ESC> Ä <Top> ;		<Bottom> r
Programerbara lysdioder	<ESC> Ä <Par> ;		<Par> ; ... ; <Par> q
Reset, omstarta av terminalen	<ESC> c		
Välj teckenuppsättning	G0 designator		G1 designator
United Kingdom	<ESC> (A		<ESC>) A
United States	<ESC> (B		<ESC>) B
Grafisk teckenuppsättning	<ESC> (0		<ESC>) 0
Sätt tab.stopp i aktuell kolumn	<ESC> H		
Ta bort tab i aktuell kolumn	<ESC> Ä g	eller	<ESC> Ä 0 g
Ta bort alla tabulatorstoppar	<ESC> Ä 3 g		
Set och reset mode	Set		Reset (Sista tecken är litet L)
New line	<ESC> Ä 20 h		<ESC> Ä 20 l
Cursor key mode	<ESC> Ä ? 1 h		<ESC> Ä ? 1 l
VT52 mode			<ESC> Ä ? 2 l
Origin mode	<ESC> Ä ? 6 h		<ESC> Ä ? 6 l
Wraparound	<ESC> Ä ? 7 h		<ESC> Ä ? 7 l
Auto repeat	<ESC> Ä ? 8 h		<ESC> Ä ? 8 l
Application keypad	<ESC> =		<ESC> >

VT52 MODE

Markör upp	<ESC> A	Markör ner	<ESC> B
Markör höger	<ESC> C	Markör vänster	<ESC> D
Välj grafisk teckenuppsättning	<ESC> F	Välj ASCII	<ESC> G
Flytta kursorn 'home'	<ESC> H	Reverserad line feed	<ESC> I
Radera till slutet på skärmen	<ESC> J		
Radera till slutet på raden	<ESC> K		
Direkt markörpositionering	<ESC> Y	Line>(+31) <Column>(+31)	
Statusrapport	<ESC> Z	svar: <ESC> / Z	
Sätt applikation keypad	<ESC> =		
Reset applikation keypad	<ESC> >	Sätt ANSI mode (VT100)	<ESC> <

CALC.80K

CALC
KALKYL-program för ABC80 med 80-teckensskärm och utbyggt minne (32 Kbyte) av S-Å Abrahamsson. Denna hjälptext har skrivits av Bengt Sandgren.

BESKRIVNING:
Programmet består av följande filer:

1. CALC.80K
Startmodul med menyval av ny kalkyl/befintlig kalkyl/hjälptext/sluta.

2. CALC1.80K
Rutin som visar denna hjälptext via menyn och ger återhopp till CALC.80K.

3. CALC3.80K
Huvudmodul med printerrutin.

4. CALCOM.DAT
Kontrollfil som skapas och läses av proammet.

5. CALC.TXT
Denna textfil.

6. CALCEX.CA3
Ett litet kalkylexempel.

Filerna CALC.80K, CALC3.80K och ALCOM.DAT har hämtats från ABC-monitorn i Stockholm.

Efter bearbetning och komplettering med CALC1.80K, CALC.TXT och CALCEX.CA3 presenteras nu filerna som ett programpaket.

Programmet är av typ "spredsheet" dvs Du kan lagra text, värden och formler i ett rutnät.

All text och alla värden kan ändras varefter omkalkylering sker med de nya värdena. Formlerna kan ändras. Storleken på den valda matrisen kan dock ej ändras.

Rutnätet är numrerat på följande sätt:

rader:	AA, AB, AC,...AJ
.	BA, BB, BC,...BJ etc
kolumner:	00, 01, 02,....nn

Rutan längst upp till vänster är alltså AA00.

I varje ruta kan Du använda EN funktion av en typ. Villkorssatser kan ej användas i rutorna. I rutorna kan Du skriva text, siffror eller skiljetecken.

HANTERING:
Programmet startas med RUN CALC.80K. I huvudmenyn kan Du välja att skapa ny kalkylmodell, läsa in en befintlig kalkyl, läsa hjälptexten eller sluta.

1. Ny kalkylmodell.
Ange antal rader, kolumner och hur många tecken varje fält skall innehålla. Fälten behöver inte vara större än att text, värden och resultat ryms. Programmet kontrollerar att matrisen ryms och ger annars möjlighet att minska något av värdena. Om allt är OK skrivs kontrollfilen "CALCOM.DAT" ut på disketten varefter huvudprogrammet "chainas" in.

2. Befintlig kalkyl.
Ange kalkylnamn utan extension. Namnet kontrolleras. Om filen inte finns ange nytt namn. Return går tillbaks till menyn.

3. Läsa hjälptext.
Du kan läsa den här hjälptexten via 3. i menyn som gör "CHAIN" på CALC1.80K.

När texten kommer upp på skärmen mata raderna med mellanslag. Med S eller s stoppas läsningen och du hamnar i en minimeny med val av återgång till huvudmeny eller upprepad läsning av hjälptexten.

4. Sluta.
Detta är utgången ur programmet.

I huvudprogrammet finns det två olika "mod" A - manövermod och B - funktionsmod.

A. I **manövermod** visas följande kommandorad överst på skärmen.
-> <- ^ (=ASCII 94) <RET> $

Manöverkommandona betyder:

-->	Flytta markören en ruta åt höger.
<—	Flytta markören en ruta åt vänster.
^ (=ASCII 94)	Flyttar markören en ruta uppåt.
<RET>	Flyttar markören en ruta nedåt.
$	Övergång till funktionsläge.

Övriga tangenter ger text i rutan.

B. I **funktionsmod** visas följande kommandorad överst på skärmen som en rad.
A C D G L N P R S X + - * / H U K

Funktionskommmandona betyder:

A Average	Beräkna medelvärde.
C Calculate	Kalkylera matrisen.
D Delete	Radera värden och funktioner i den ruta som Du befinner Dig i.
G GOTO	Gå till ruta.
L Load	Ladda in den ursprungliga kalkylen.
N	Radera hela rutnätet.
R Repeat	Repetera rutans innehåll och tillhörande funktion i angivna rutor.
S Save	Spara kalkylen på skiva. Första tecknet av max 8 skall vara en bokstav. Extension sätts av programmet till "CA3".
X	Gå ur funktionsmod.
+	inleder "+"-funktion (addition).
-	inleder "-"-funktion (subtraktion).
*	inleder "*"-funktion (multiplikation).
/	inleder "/"-funktion (division).
H	Användes inte
U	Utskrift på printer av formler/ och eller text med 69 tkn/rad.
K	Avbryter och går tillbaks till menyfunktionen.

Det finns således 6 olika funktionstyper som kan definieras i rutorna.

A	Average
R	Repeat
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

Du kan använda en funktionstyp i varje ruta. Definiera en funktion genom att först gå till den ruta som funktionen skall vara i. Gå sedan över i funktionsmod med "$". Definieringen inleds därefter med något av tecknen"A R + - * /". Dessa 6 funktioner kan beräkna värdet av innehållet i enföljd av rutor. Om man avser att endast summera t ex rutorna AA00 och AA01 och lägga resultatet i AC00 så blir kommandosträngen:

AC00 = AA00 + AA01...AA01 D 2

Efter bokstaven D anger man antalet decimaler. Enda sättet att klara "dubbelfunktioner" är att mellanlagra den första funktionens värde i en ruta som sedan anropas av nästa funktion.

Ett tips är: lagra konstanter i rutor på första eller sista raden.

När rutnätet fyllts i med text och värden kan det sparas med kommandot "S". Före avslut med kommandot "K" kommer frågan SPARA PÅ FIL J/N. Även här kan matrisen sparas på disketten.

Före utskrift på printer bör lämplig printerkod sättas i CALC3.80K.

KALKYLERING:
Bygg först upp din kalkyl på ett rutat papper. Det ger Dig möjlighet att tänka igenom uppläggningen innan Du lägger in funktioner och konstanter i rutorna.

Matrisen kalkyleras i "ASCII-aritmetik". Tomma rutor ger "noll" vid funktionerna + - *. Vid division på tomma rutor blir det saklöst Error liksom vid division med noll.

Ett sätt att förhindra sådana fel är att göra en "noll-matris". Fyll alla rutor med 0 (noll) på följande sätt:

* Skriv in 0 i ruta AA00 i den tomma matrisen.
* Gå över i kommandomod med $ och ange "R".
* Ange sedan AA00 och efter det "R" som kommer upp AA00...AC02, om matrisen har 3 rader och 3 kolumner.

Om hela kolumn 0 skall reserveras för text, vilket är ganska vanligt byts AA00 ...AC02 ovan ut mot AA01...AC02. När så övriga rutor är fyllda med "0" raderas ruta AA00.

På motsvarande sätt kan man spara den översta raden för text genom att repetera AA00 till AB01...AC02. Kalkyleringen av matrisen sker i ordningen rader och sedan kolumner. Detta medför att kalkylerade värden som skall anropas av en ruta måste finnas i en ruta som redan passerats.

(HUR) LÄNGE LEVE ABC80!?

Ibland får man intrycket att ABC80 numera anses "ute". Det är en gammalmodig och förlegad apparat. Bäst göra sig av med sin så fort som möjligt och satsa på något annat, kanske ABC806 - eller rent av IBM PC..!

Visst är åttian gammal, enligt datormåttstock. D v s den utsprungliga ABC80, med 16K och kassettspelare som yttre minne. Men om man diskuterar ABC80 i dag måste man ta hänsyn till de mycket kraftfulla tillsatser som i dag finns: 32K arbetsminne, 80 tecken per rad möjlighet att kraftigt utöka kommandorepertoaren med Superbasic eller Smartaid, skivspelare med dubbel packningstäthet och diverse annat.

Naturligtvis kostar allt det där extra pengar, och om någon i dag skall köpa en dator helt från början kan man komma billigare undan än med ABC80. Men nu talar jag om alla de datorer som redan finns hos oss.

Frågan är: hur länge till kan vi hänga med? Jag tycker mycket talar för att det blir länge än. Vi kan göra det mesta som behövs i hackersammanhang, från modemkommunikation till dataspel - och det finns stora mängder program tillgängliga.

Apparaten är mycket väl genomforskad. Det finns möjligheter till åtskilligt, som på flertalet andra datorer är reserverat för de mystiska männen i vita rockar - och vad värdefullare är: genom ABC-klubben kan vi få kontakt med dem som kan tala om för oss hur man gör, även med sådant som inte står i instruktionböckerna.

På min apparat kan jag direkt läsa och skriva i arbetsminnet; jag kan ögonblickligen få tillbaka program som förlorats med RESET; jag kan läsa och vid behov skriva direkt på flexskivorna (och därmed i vissa fall laga trasiga skivor och förstörda filer).

Jag kan ta fram LIB på skärmen och jag kan dumpa skärminnehållet på printern utan att störa vad som finns i arbetsminnet. Jag kan leta rätt på alla förekomster i ett program av en viss variabel för att med en enkel skärmeditor ändra/rätta och veta att man inte har hoppat över något!

Jag kan skriva texter, redigera dem och spara dem utan att behöva uppta arbetsminnet med textprogrammet osv.

Det mesta av detta kan jag tyvärr INTE göra på någon 800-modell och jag har inte sett någon annan mindre dator där det går.

Visst finns det i dag BASICII, som är mycket bättre än ABC80-basic. Visst har man datorer med 16-bitars processor, färg, högupplösande grafik, flera ljudkanaler osv. Är dessa saker viktiga, då kan inte ABC80 hänga med.

Men hur viktiga är de? Är de verkligen viktigare än de andra saker jag räknat upp, där vår gamla traktor ändå ligger före?

Det som ABC80 var bra på för fem år sedan är den väl fortfarande (tillräckligt) bra på - ibland rent av bäst på - eller?

Vad anser läsekretsen? Vässa pennorna och skriv en rad!

Sven Wickberg

ABC-KASSETT 15

TRMHLP.TXT

Detta program definierar initialvärden för setup-parametrar i TERM100 terminal program (som emulerar VT100).

Inställningsförfarandet efterliknar 'setup'-moderna i TERM100.

Tryck R,T,Y eller U för att komma in i respektive setup-mode. De värden som visas kommer att bli defaultvärden för TERM100 vid uppstart och reset.

Ändra värdena genom att trycka på siffertangenterna 1 t o m 7.

TRMHLPU.TXT

I SETUP-U anger man defaultvärden för de definierbara strängarna. Efter att man tryck 5, 6 eller 7 så skriver man in strängen som man vill ha den. CTRL-

CTRL-tecken måste anges med understrykningstecken (_) t.ex. _M för RETURN. Vidare måste CTRL-SHIFT-O skrivas som "_ASCII(127)" och "_" måste skrivas som "__".

Inmatning av MELLANSLAG kan göras med framåtpil eller CTRL-S.

Max 25 tecken får ingå i en sträng.

Inmatning av sträng avslutas med RETURN eller mellanslag eller CTRL-X. CTRL-X tar inte bort strängen.

TRMHLPL.TXT

SETUP-I är en specialmode i detta setupprogram.

SETUP-I finns ej i TERM100. I denna mode kan man välja om tangentnedtryckningar ska ge klick i högtalaren.

Redaktionen har fått några frågor om litterarur speciellt avpassat för ABC-datorerna

Fråga:

Disassembler-rapporten, är den lämplig även för ABC-800-användare?

Svar:

"Disassemblern", ABC-klubbens rapport nr 1 är en redovisning av programvaran i ABC80, och närmare bestämt med checksumma 11 273. Skillnaden mot andra checksummor på ABC80 är inte större än att man kan hitta vad man är ute efter. Det skall dock sägas att det inte bara är att läsa den från pärm till pärm, man bör nog vara ganska slängd i assembler-programmering för att kunna nyttja den fullt ut. Och då kan man få ut mycket av bokens dryga 300 sidor.

Och vissa av BASIC II:s rutiner finns ju med redan i BASIC I, men ligger nog på andra ställen. Det här är ju möjligt att med mycket träget eget arbete känna igen och hitta rutiner som kan nyttjas genom anrop från program även i ABC800/806.

Fråga:

Om inte, vilken bok rekommenderas då om man vill lära sig assemblerprogrammering på 800:an från grunden?

Finns det någon litteratur i stil med BIT för BIT med 800, men i stället avseende ABC806?

Svar:

BIT för BIT med 800 är att rekommendera och den gäller i stort sett även för ABC806.

Redaktionen

Bruksanvisning till DRAW16K.BAC och DRAW32K.BAC på kassett 15

DRAW är en subrutin i maskinkod.

Koden finns i DRAW16K.BAC resp DRAW32K.BAC tar 618 byte och lagras under BOFA i minnet. (Den innehåller även rutinerna för SPRITE)

Vad den gör är att snabbt dra linjer på skärmen. Du kan få den att dra en linje eller tusen med ett CALL till rutinen.

Man kan skapa flera bilder med massor av linjer. Dessa bilder lagras i en matris i BASIC.

DRAW-rutinen som ritar linjerna är gjord så att det aldrig blir ERROR 62, utan då hörs bara ett tickande i högtalaren som upplysning och de delar av linjerna som finns inom skärmen ritas. Inte ens de tecken som ger grafmod förstörs av misstag.

DRAW fungerar på både 40- och 80-tecken skärmbredd.

DRAW tar värdena för linjerna från den plats i minnet där tolken lagrat dem.

DRAW kan både rita och sudda.

För att använda DRAW i egna program är det lämpligt att utgå från programmet DRAWDJUP.BAS .Där finns en färdig **subrutin INITIERA** som sköter lagring av linjer i resp bild.Den räknar även ut var maskinkoden ska hämta värdena till alla linjers ändpunkter.

Det du behöver göra för att skapa egna linjer/bilder är :

1) Bestäm antal bilder och linjer/bild (rad 110).
   Initiera variablerna (rad 120).
2) Sätt P%=bildnr du ska bearbeta eller skapa.
3) Lagra en linje:
   100 X1%=<kolumn>:Y1%=<rad>:X2%=<kolumn>:Y2%=<rad>:GOSUB 60021
   4) Upprepa 3) tills bilden är klar.
5) Sätt skärmen i grafmode:
   150 ;CHR$(12):FOR I=0 TO 23 : ;CUR(I,0);:NEXT I

När bilderna är skapade får man dom blixtsnabbt visade med följade satser.

```
(Rita bild 2)
POKE Q%,Q1% : REM setdot
Z%=FNP%(2) :REM execute PIC 2

(Sudda bild 3)
POKE Q%,Q0% : REM clrdot
Z%=FNP%(0) :REM execute PIC 3
```

SPRITE är en subrutin i maskinkod

Både DRAW- och SPRITE-rutinerna ligger i DRAW16K.BAC (för 16Kbyte minne) och DRAW32K.BAC.

Sprites är "gubbar" som kan förflyttas på skärmen utan att förstöra den befintliga bild som de rör sig över.

Lista programmet SPRITE.BAS och försök förstå det.

Här är lite hjälp för egen programmering

"Gubbarna", högst 4 st görs lättast i DATA-satserna. Men man kan även lagra dom själv byte för byte och POKA dom på plats.Det senare kostar mindre minne.

Om du gör (POKE Q3%,1) så fungerar gubbarna var för sig vid förflyttning. Om du däremot gör (POKE Q3%,0) så avlöser dom varandra vid förflyttning.

En sprite förflyttas sålunda:

```
100 Z%=FNS%(S,R,K)
      S= sprite nr 1,2,3 eller 4.
      R= till rad i grafmod.
      K= till kolumn i grafmod.
```

SUBRUTINER!

Två små subrutiner.

Jag har valt att genomgående använda I, I% eller I$ som parametervariabler och U, U% eller U$ som resultatvariabler. Övriga variabler är arbetsvariabler eller slask. I övrigt har jag redigerat om rutinerna i enlighet med Rune Mattsson's förslag i ABC-bladet. Detta innebär att rutinerna direkt körbara med exempel.

GOSUB leder till aktiv rad varför samtliga REM-rader kan avlägsnas om man så önskar.

1. FRAME.BAS gör en ram på skärmen av skrivbart ASCII-tecken. Vad det skall tjäna till får Du räkna ut själv. Rutinen behöver knappast förklaras.

Du sätter parametrar som styr storlek och läge på skärmen och vilket tecken som skall användas gör GOSUB och ramen finns där. Det går självklart att med cursorn styra in text i ramen.

2. POKEWHEN.BAS är en subrutin för datumangivelse vid utskrifter. Den här rutinen har jag lagt in bl a i mitt programlistningsprogram för att slippa göra input varje gång jag kör programmet men vill ha aktuellt datum på listan.

Rutinen är generellt användbar när man behöver dagens datum i olika program eller vid upprepade tillfällen och funkar om man inte pokar något annat högst upp i minnet eller slår av strömmen. Rutinen testar först om 65535 innehåller ASCII 42 ('*'). Om så är fallet är datum redan inläst och presenteras på rad 22 för godkännande. Input av datum sker annars på rad 22. Rimlighetskontroll sker av input men ej strikt kontroll av antal dagar i respektive månad. Endast februari kollas till max 29 dagar. Orimlig input raderas. Rutinen väntar därefter på ny input. Datum lämnas i variabeln U$.

På rad 120 står POKE 65535,0. Detta är endast för att testprogrammet skall bli meningsfullt.

Rutinen innehåller ett GOTO. Ta bort det den som kan.

Det förekommer vissa program där samma adresser används för något ändamål. I så fall får man leta reda på någon annan adress att lägga datum på.

<2776> Bengt Sandgren

```
1 REM LIST FRAME.BAS
2 REM Rutin för ram av text på skärmen
3 REM 19840414.1740 B Sandgren (C)
4 REM --------------------------------
100 REM Testprogram_____________________
110 FOR N%=0% TO 4% : READ I%(N%) : NEXT
    N%
120 DATA 39,21,0,0,36
130 ; CHR$(12%) : GOSUB 310
140 STOP
150 REM Slut testprogram _______________
160 REM ================================
170 REM
200 REM FRAME.BAS ++++++++++++++++++++
210 REM Rutin för ram av asciitecken på
220 REM skärmen.
230 REM För ABC80, ABC800
240 REM
250 REM
260 REM Insänt av Bengt Sandgren <2776>
270 REM ++++++++++++++++++++++++++++++++
280 REM I%(0)= ramens bredd
281 REM I%(1)= dito höjd
282 REM I%(2)= rad för övre vänstra hörn
284 REM I%(3)= kolumn för dito
285 REM I%(4)= tecknets ascii-värde
300 REM ++++++++++++++++++++++++++++++++
310 ; CUR(I%(2%),I%(3%))STRING$(I%(0%),
    I%(4%))
320 ; CUR(I%(2%)+I%(1%)-1%,I%(3%))STRING$
    (I%(0%),I%(4%))
330 FOR N%=1% TO I%(1%)-2% : I%(2%)=
    I%(2%)+1%
340 ; CUR(I%(2%),I%(3%))CHR$(I%(4%))
    CUR(I%(2%),I%(3%)+I%(0%)-1%)CHR$(I%(4%))
350 NEXT N%
370 RETURN : REM Åter till huvudprogram
380 REM


1 REM LIST POKEWHEN.BAS
2 REM Poka ÅÅMMDD från 65535-och neråt
3 REM 19850414.1758 (C) B Sandgren
100 REM ================================
110 REM Testprogram ____________________
120 POKE 65535%,0%
130 GOSUB 320
135 ; CUR(22,0)TAB(40);
140 ; CUR(22%,0%)U$ : STOP
150 REM Slut testprogram_______________
160 REM ================================
190 REM
200 REM POKEWHEN.BAS ++++++++++++++++++
210 REM POKAR ÅÅMMDD från 65535-och
220 REM nedåt.
230 REM För ABC80
240 REM Insänt av Bengt Sandgren <2776>
250 REM ++++++++++++++++++++++++++++++++
260 REM N% = lopräknare
270 REM Z$ = slasksträng
280 REM U$ = utsträng
290 REM ++++++++++++++++++++++++++++++++
300 REM -- Rutin för datum ÅÅMMDD -----
310 REM -- Rutinens styrdel -----------
320 IF PEEK(65535%)=42% GOSUB 390 ELSE
    GOSUB 470
330 ; CUR(22%,0%)TAB(40%);
340 ; CUR(22%,0%)CHR$(7%)U$+'  Är datum
    ok J/N (N) '; : GET Z$
350 IF Z$='J' OR Z$='j' Z$="" : N%=0% :
    RETURN
360 POKE 65535%,0% : GOTO 320
370 REM ------------------------------
380 REM -- Peek'a hem datum till U$ ---
390 U$="19"
400 FOR N%=1% TO 6%
410 IF N%=3% OR N%=5% U$=U$+'-'
420 U$=U$+CHR$(PEEK(65535%-N%))
430 NEXT N%
440 RETURN
450 REM ------------------------------
460 REM -- Input datum ÅÅMMDD ---------
470 ; CUR(22%,0%);TAB(40%)
480 ; CUR(22%,0%)'=> Datum ÅÅMMDD:';
490 ; CUR(22%,16%); : INPUT Z$
500 IF LEN(Z$)<6% 470
510 IF MID$(Z$,1%,2%)<'01' THEN 470
520 IF MID$(Z$,1%,2%)>'99' THEN 470
530 IF MID$(Z$,3%,2%)<'01' THEN 470
540 IF MID$(Z$,3%,2%)>'12' THEN 470
550 IF MID$(Z$,3%,2%)='02' AND
     MID$(Z$,5,2)>'29' THEN 470
560 IF MID$(Z$,5%,2%)<'01' THEN 470
570 IF MID$(Z$,5%,2%)>'31' THEN 470
580 REM ------------------------------
590 REM -- Poka datum och signal -----
600 FOR N%=1% TO LEN(Z$)
610 POKE 65535%-N%,ASC(MID$(Z$,N%,1%))
620 NEXT N%
630 POKE 65535%,42%
640 GOSUB 390
650 RETURN
660 REM
```

Vilken dator körs programmet på?

När man gör ett program så kan man behöva ta hänsyn till vilken dator ur 800-serien som programmet körs på. För att ta reda på detta finns det olika metoder. I senare versioner, troligen från ABC802 och framåt, finns det i adress 38 en uppgift om BASIC-version och i adress 39 en uppgift om BASIC-typ. Vad jag har erfarit så är innehåller i adress 39 följande:

ABC802	3
ABC806	4
DTC 2	6

Version av DOS kan man också ta reda på. I maskiner med UFD-DOS gäller att i adress 24687 står det en nolla om det är UFD-DOS och i adress 24688 står dostypen. För maskiner som inte har UFD-DOS kan man också ta reda på vilket DOS som finns, men det blir lite svårare då. Det framgår av sid 72 i Bit för Bit hur man gör.

Vill man ta reda på vilken typ av 800-maskin som programmet körs på och även klara gamla 800:or så kan man använda subrutinen enligt nedan. Den fungerar naturligtvis inte på DTC-maskiner.

Rutinen tilldelar variabeln Tolk$ innehållet i BASIC-tolken genom BASIC-instruktionerna POKE och VAROOT. Närmare detaljer om hur detta sker framgår av Bit för Bit sid 100-106. Sedan söker rutinen helt enklet efter "ABC80" i Tolk$ med instruktionen INSTR. Eftersom det alltid står antingen ABC800, ABC802 eller ABC806 där så hittar programmet det! Sedan är det bara och skilja på ABC800 C och M om svaret blir ABC800. Detta görs genom att kontrollera om instruktionen SET DOT ger felmeddelande. SET DOT finns inte i ABC800 M och ger därför ett felmeddelande.

Vad programmet inte klarar är att ta reda på om ABC800 C/M är utrustad med grafikkort. Det går inte att testa med någon FG-instruktion eftersom ABC802:ans BASIC inte känner till FG-instruktioner. Är det någon som har löst det?

Bo Kullmar

```
2 ! ************************************
3 ! Funktion          TYP.BAS
    Utgåva  1.0       1985-03-10
4 ! av Bo Kullmar
5 ! Insänd av Bo Kullmar <1789>
6 ! För ABC800M ABC800C ABC802 ABC806
7 ! Testad på ABC806, ABC802
8 ! Funktionen anger vilken datortyp som
    programmet körs på genom att läsa
9 ! tolken och se om det står ABC800,
    ABC802 eller ABC802 där. Står det
10 ! ABC800 så kollas det som CLR DOT
   ger error och därigen kan man skilja
11 ! på ABC800 C och ABC800 M.
12 ! ************************************
20 INTEGER : EXTEND
1000 ; FNTyp$
1010 DEF FNTyp$ LOCAL Adr,Tolk$=0,Typ$=6
1020   ! In: --
1030   ! Ut: Datortyp, PIC X(8)
1040   POKE VAROOT(Tolk$),0,96,0,0,0,96
1050   Adr=INSTR(1,Tolk$,"ABC80")
1060   Typ$=MID$(Tolk$,Adr,6)
1070   IF Typ$<>'ABC800' RETURN Typ$
1080   ON ERROR GOTO 1110
1090   CLR DOT 0,0
1100   RETURN 'ABC800 C'
1110   RETURN 'ABC800 M'
1120 FNEND
```

Vilken version?

När man utvecklar ett program, sparar man under tiden versionerna med namn som PROGRAM.BAS och därnäst PROGRAM2.BAS etc.

Sen kommer det en eller flera varianter ut på diskarna. Plötsligt upptäcker man att man har en uppsjö varianter och kopior på samma program. Då får man städa. För detta ändamålet har jag tillverkat detta nyttoprogram JÄMFÖR.PGM.

Du svarar med namnen på de två programmen som du skall jämföra. Nu tar maskinen fram programmen och lägger dem bredvid varandra i minnet.

Den jämför varje radnummer. Om innehållet i en rad inte är identiskt lika i de två programmen, skriv båda raderna ut. Detta gäller också om ett radnummer i den ena versionen saknas i den andra. Du kan välja att få avvikelserna utskrivna på skrivaren.

<3098> Egil Fjelddahl

```
10 REM LIST JÄMFÖR.PGM EGILS PROGRAMMER
20 DIM A$=128% : DIM A1$=128% : DIM A2$=
   128%
30 ; CHR$(12)'Programmet har handbroms
   - när du håller ner en tangent står
   det still' : ; : ;
40 ; 'Om du har samma namn på programmen
   på två diskar, skriv DR1: framför ena
   namnet '; : ; : ;
50 ; 'Första filen :'; : INPUT F1$
60 ; 'Andra filen :'; : INPUT F2$
70 GOTO 98
80 CLOSE 1 : CLOSE 2 : ; 'Samma filer en
   gång till'; : INPUT A$ : IF A$='j' 90
   ELSE STOP
90 ; 'På printern (n)'; : INPUT A$ : IF
   A$='j' OR A$='J' P%=3% : OPEN 'PR:'
   ASFILE 3%
100 ONERRORGOTO 80 : OPEN F1$ ASFILE 1 :
    OPEN F2$ ASFILE 2
110 ; $P%,'Jämför rader med samma
    radnummer för 'F1$' och 'F2$ : ; $P%
120 F1$=F1$+SPACE$(16%-LEN(F1$))
130 F2$=F2$+SPACE$(16%-LEN(F2$))
140 REM ...START LÄSNING
150 IF INP(56%)>127% THEN 150
160 IF A1%=A2% GOTO 220
170 IF A1%>A2% GOTO 190
180 GOSUB 200 : ; $P% : F%=1% : GOSUB
    260 : A1%=A% : A1$=A$ : GOTO 140
190 GOSUB 210 : ; $P% : F%=2% : GOSUB
    260 : A2%=A% : A2$=A$ : GOTO 140
200 ; $P%,F1$' : 'A1$' 'A1$ : RETURN
210 ; $P%,F2$' : 'A2%' 'A2$ : RETURN
220 IF A1$<>A2$ GOSUB 200 : GOSUB 210 :
    ; $P%
230 F%=1% : GOSUB 260 : A1%=A% : A1$=A$
240 F%=2% : GOSUB 260 : A2%=A% : A2$=A$
250 GOTO 140
260 INPUTLINE $F%,A$
270 A$=LEFT$(A$,LEN(A$)-2)
280 B%=INSTR(1%,A$,CHR$(32%))
290 A%=VAL(LEFT$(A$,B%-1%))
300 A$=RIGHT$(A$,B%+1%)
310 RETURN
```

Gör en ordbehandlare av TV-editorn

En viktig del av en ordbehandlare är att du kan vräka på med text utan någon RETURN för varje ny rad. Det skall maskinen själv sköta. Du skall också spara din text ofta, och då får ingen rad vara längre än 120 tecken. Eftersom TV-editorn inte är fullkomlig här, har jag plockat fram en lösning. Var så goda.

Följande snutt till TV-editorn delar alla för långa stycken till önskad längd. D v s du sätter en maximal radlängd och raden kapas efter det mellanslag som ligger närmast innan den maximala längden. Om du inte sätter någon maximal radlängd används 79 som blir vackrast på skärmen och passar till skrivare med 12 tecken per tum.

Om du önskar att koppla ihop en styckad sats igen, används ctrl-E ctrl-K upprepade gångar. Efter en dylik hopkoppling kan du editera stycket och sedan dela det editerade stycket igen.

Bruksanvisning

<$$ ger delning till maximalt 79 tecken.

nn<$$ ger delning till maximalt nn tecken.

<3098> Egil Fjelddahl

```
1070 M8%=0% : L8%=0% : L4%=79%
2660 IF C$<>'<' 2670 : REM RADBYTE
2662 IF N%>1% L4%=N%
2664 GOSUB 3900 : GOTO 1700
2670 IF C$<>'J' THEN 2681 : REM JUMP
3124 ; CUR(23%,0%)'Raden är för lång!
     <$$ ger längd';L4%; : RETURN
3900 REM ** RADBRYTA **
3910 P%=1% : P1%=P%
3920 IF P1%<LEN(M$) THEN GOSUB 3690 ELSE
     RETURN
3930 IF P1%-P%<L4%+3% THEN P%=P1% : GOTO
     3920
3940 FOR P1%=P%+L4% TO 1% STEP -1%
3950 IF MID$(M$,P1%,1%)=CHR$(32%) 3970
3960 NEXT P1% : P1%=0% : RETURN
3970 P%=P1% : E%=0% : M5%=2% : GOSUB
     3350 : IF E% THEN RETURN
3980 POKE M7%+P%,13%,10% : E%=-1% : GOTO
     3920
```

```
+------------------+
!  Typografi       !
!                  !
!  S-ELITE-hjul    !
!                  !
!  ! = ASCII  33   !
!  " = ASCII  34   !
!  § = ASCII  35   !
!  $ = ASCII  36   !
!  % = ASCII  37   !
!  & = ASCII  38   !
!  ' = ASCII  39   !
!  ( = ASCII  40   !
!  ) = ASCII  41   !
!  * = ASCII  42   !
!  + = ASCII  43   !
!  , = ASCII  44   !
!  ´ = ASCII  64   !
!  ^ = ASCII  94   !
!    = ASCII  95   !
!  ¯ = ASCII  96   !
!  ü = ASCII 126   !
+------------------+
```

ABC ÖST

Box 6129
580 06 Linköping
Postgirokonto 483 91 35-3

Från östra kanten av vårt land kan vi nu börja summera det som hänt den första tiden av vår verksamhet. Vi har haft en ganska stor mängd aktiviteter som tydligen även accepterats av våra medlemmar eftersom vi lyckas att en vanlig vardagskväll samla folk som har en körsträcka på ungefär 25 mil för att kunna infinna sig. Det gör att vi från styrelsens sida känner kravet att försöka ge dessa medlemmar, och givetvis även alla andra, ett programutbud som gör att vi även i fortsättningen kan vidmakthålla och om möjligt öka det otroliga gensvar vi tycker oss ha fått.

Vi vädjar också från styrelsens sida till alla andra inom vårt område att göra slag i saken och betala in 25:- och gå med i lokalavdelningen. Ni är välkomna och behövs för att ytterligare öka våra möjligheter.

De program vi hitills haft är bl a studiebesök på Luxor och SVT samt RR (Riksradion), demonstration av möjlighet att köra ABC80 mot dubbla bandspelare, hur man använder diskarean som RAM-floppy med batteribackup, mycket uppskattade demonstrationer av egna förmågor. Vi har även haft demonstration av DIS (Dator I Släktforsning), JSP-800, TOOLBOX 2, COMP-REGINA och mycket mer.

Även här är det delvis egna medlemmar som ställt upp. Det har varit givande och intressanta stunder och vi har all anledning att vara tacksamma mot dessa "demonstratörer" som gett oss så otroligt mycket.

ABC-Öst har även lyckats ordna vissa rabatter hos en del firmor. Den vanliga rabattsatsen är 10% men vissa undantag finns. De firmor som lovat rabatt är Data och Kontorskonsult AB, Linköping Material Clearing, Norrköping Eltema, Linköping. Överenskommelsen med dessa är att vi skall kunna visa att vi verkligen är medlemmar i ABC-klubben. Uppge därför medlemsnummret när du vill aktivera rabatterbjudandet för att firman enkelt skall kunna kontrollera medlemskapet.

Vårens sista träff kommer vi att ha onsdagen den 29 maj då vi är inbjudna till Data och Kontorskonsult AB för att studera nätverk och modemanvändning. Vi har i styrelsen mycket stora förväntningar.

Höstens upptakt kommer att bli den 4/9, med tema CAD/CAM. Även denna träff har vi anledning att rekommendera om vi får planerna att gå i lås.

Vårt första möte höst blir blir onsdagen den 4 september och sedan träffas vi första helgfria ondagen i varje månad. Träffadressen är normalt **Ånestadsgården, Ridhusgatan, Linköping.** Temat för den första kvällen blir CAD/CAM.

Vi försöker meddela våra medlemmar genom personlig kallelse.

I övrigt gäller att Ni bör kontakta oss om det är några frågor som Ni vill ställa. Säkerligen kan vi inte svara på allt men inom klubben finns en stor möjlighet att få svar på det mesta, och även viss hjälp.

Vill du ha ytterligare information kan du ringa någon av

Nils Larsson (ordf)	013-134565
Lars Kållner (sekr)	013-66297
Tommy Drotz (kassör)	013-151536

JOYSTICK

Bygga om en joystick till ABC80

Jag har använt en COMPETITION PRO, en joystick avsedd för Atari och VIC. Den kostade då 199:-.

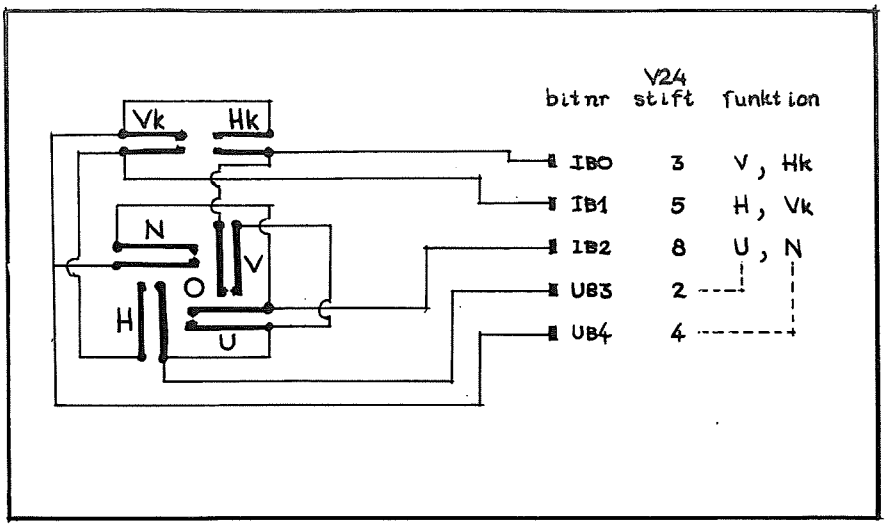
Befintlig sladd används, men V24-kontakten måste bytas mot en ny för att placeringen av stiften skall bli rätt.

I dosan löds alla trådar om enligt följande schema:

```
65525 REM §§ JOY.SUB §§
65526 REM §§§§§§§§§§§§§§
65528 OUT 58,16 : Ö%=INP(58) AND 7
OUT 58,8  : Ö%=Ö%+(INP(58) AND 7)*8
65529 IF NOT Ö% AND 1%  R$='Hk'
65530 IF NOT Ö% AND 2%  R$='Vk'
65531 IF NOT Ö% AND 4%  R$='N'
65532 IF NOT Ö% AND 8%  R$='V'
65533 IF NOT Ö% AND 16% R$='H'
65534 IF NOT Ö% AND 32% R$='U'
```

Kombinationer kan göras för diagonala förflyttningar

<2401> Tom Sjöberg



Radannonser

Säljes
Oanvänd TKN80 säljes till högstbjudande.

<5654> Göran Mattsson
Vitalisgatan 28
441 43 Alingsås

Säljes:
Radskrivare Scandia-Metric Itoh 8300P med paralellsnitt och engelsk beskrivning. Nästan ny.
Pris 3700:- eller högstbjudande

<5621> K-O Wiren
0586-82 653, dagtid
0550-19 386, kvällstid

SÄLJES

Obetydligt begagnad Diskettenhet ABC 830

Priside´ 3200 SEK

Lena Arkwall

Tel 08-715 47 82

Säljes
Programvara för ABC800, Facit DTC. Ord, Kalkyl, Databas, Grafik i engelsk version för diskdrive 320/640 bortslumpas för 1900:- per styck inklusive handledning och backup i orginalförpackning.

Tord Colton
08-93 29 32

Säljes
1. CP/M-tillsats för ABC800M. UNI 800G med CPM, nytt 1500:-.
2. Videokort 40 tkn färg till ABC800C med PROM, 600:-.
3. Flexskiveenhet 8 tum, 2 x 0.5 MB, med controllerkort 4500:-.
4. Matrisskrivare, liggande A4, Philips, 1000:-.
5. Flexskiveenhet ABC832, 80-spårs 2 x 640 kB, 7500:-.

<1416> Leif Åhlin
742 23 24, kvällstid

Säljes
ABC80 inklusive bandspelare och nummeriskt tangentbord, expansionsenhet med 16 kB och floppy 2 x 160 kB.
Program: Basregister, ABC ORD och bokföringssystem, ABC-klubbens kassetter 3 - 14 samt diverse litteratur.

<2827> Lars-Olle Gustavsson
Snapphanegatan 20
271 00 Ystad
0411-17 531

Säljes
ABC830 flexminne, använt endast 10 timmar säljes för 7500:-
"Ta med dig datorn" EPSON HX20. Helt ny. Säljes med printer, kassettminne och väska. Pris 5000:-

Anders Mattsson
044-12 95 43, kvällstid

Säljes
Flexskiveenhet ABC-FD2 och skrivare Centronics 730-4. Båda lite använda.

<2767> Ingvar Malm
Godemansvägen 10
372 00 Ronneby
0457-218 76, kvällar

Till salu
ABC80 med bandspelare, DataDisk 82 försedd med 2 x 8 kB minne och Centronics-interface, skrivare, Microline 80.

Pris tillsammans 12 000:-

<3330> Erik Nilsson
0152-12001

Till salu
Facit Floppy-drive 2 x 320 kB. Typ 190 92 06, årsmodell 83, nyskick.

<5704> Olle Andersson
0243-800 35 (9.00-17.00)

Köpes
Anbud önskas på begagnad ABC80, med eller utan extra utrustning. Ifall extra utrustning finns, lämna gärna förslag till pris utan utrustning ifall ni kan tänka er att sälja endast datorn.

<4642> Peter Anvin
Tegelvägen 33
723 48 Västerås
021 - 13 25 05

Förmånserbjudanden:

Rabatt på Jackie

Avtalet

System Innovation AB, som har utvecklat Jackie 1200, lämnar ett introduktionserbjudande exklusivt riktat till ABC-klubbens medlemmar för köp av Jackie 1200, med ABC-kabel, till ett starkt reducerat paketpris!

Paketet består av:
1 st modem Jackie 1200
1 st kabel för anslutning Jackie - ABC80/ ABC800 (9 polig D-sub)

Följande regler gäller för erbjudandet:

Beställningar skall vara System Innovation AB tillhanda senast den 24:e december 1985. Detta innebär att beställningar måste vara klubben tillhanda senast den 16:e december 1985. Beställningarna expedieras i inkommen turordning. Paketet levereras mot postförskott, totalt paketpris 1375 kronor inkl moms, porto, postförskottsavgift, emballage och expedition. Priset kan jämföras med ordinarie pris 1800 kronor inkl moms!

Detta introduktionserbjudande gäller inte försäljning av modemet utan kabel. Dock kan du i stället få en 25 polig kabel till bl a IBM PC till priset 1.560 kronor inkl moms och kabel.

Beställning skall göras skriftligen till
ABC-klubben
Vidängsvägen 1
161 33 BROMMA.

Medlemsnummer, namn och adress skall anges. Om du undrar något, ring Bo Kullmar, 08-751 15 18 eller direkt till System Innovation, Täby.

System Innovation lämnar ett års garanti mot materiel- och tillverkningsfel. Garantitiden räknas från leveranstillfället från System Innovation. Porto för insända modem med garantianspråk erlägges av medlem, returporto för utbytt respektive åtgärdat modem betals av leverantören. Returporto för felaktigt reklamerat Jackiemodem debiteras medlemmen.

Min provkörning

Jag har provkört modemet mot olika system, inte bara system inom Stockholmsregionen. Jag har provkört det på ABC806 och ABC802 med DIAB:s terminalrutin och Autocodes VT102 emulator samt på ABC80 med TERM100 och ABCV24. Modemet har vid dessa tester fungerat utan anmärkning.

På modemet finns autosvar, som aktiveras genom att sätta DTR hög. (Stift 1 i den 9 poliga kontakten). När DTR är satt, så svarar modemet om någon ringer. Med en terminalrutin till ABC800-serien är detta inget problem eftersom DTR tas ner när man avslutar terminalrutinen.

Kör man däremot ABC80 så är det tyvärr så att i ABC80 ligger DTR alltid hög, så länge ABC80 har strömmen påslagen. Det innebär att Jackie alltid kommer att svara. För att undvika detta bör man lämpligen dra ur kontakten mellan V24:an och Jackie eller bryta stift 1 med en strömbrytare. Man kan naturligtvis också stänga av ABC80, för då är inte DTR hög längre.

Tidigare har jag rekommenderat ett 1200/ 75 bps-modem för privat bruk och körningar mot t ex klubbens monitor. Från början är Jackie ett mycket billigt modem och detta introduktionserbjudande som System Innovation har varit vänliga att ge oss gör att modemet blir ännu billigare. Jag kan därför utan förbehåll rekommendera modemet.

I dagens läge kan du inte få tag på ett snabbt och billigare modem. Visst finns det billiga 300 bps modem, men du kan inte köra 1200 bps på sådana modem.

Slutligen vill jag nämna att modemet verkligen är litet. Tittar man på modemet på bild så har man svårt att föreställa sig hur litet modemet i verkligheten är.

Bo Kullmar

Mikrodatorn

ABC-medlemmar!

Det fanns en lycklig tid då MikroDatorn innehöll masssor av tips och nyheter kring ABC-datorena, som ju ett tag dominerade den svenska marknaden. Att vi nu inte längre skriver så mycket i MikroDatorn om ABC beror faktiskt inte på på att vi trolöst lämnat ABC-ägarna i sticket till förmån för IBM- och Macintoshägare, utan snarare på att MikroDatorn bytt stil. Vi skriver inte längre så mycket maskinspecifika artiklar, utan försöker i stället förmedla nyheter och erfarenheter och allmänna tips. I den mån tidningen har övervikt åt IBM-hållet så beror det på att det kommer så mycket NYHETER från IBM.

Allmän datortidning

Dagens tidning ska alltså ses som en allmän intressetidning för människor som arbetar med eller är intresserade av persondatorer eller personlig datoranvändning. Ja, överhuvudtaget de som är intresserade av datorn som tekniskt och samhälleligt fenomen bör kunna ha glädje av MikroDatorn. MikroDatorn kommer alltmer att ägna utrymme åt det allmängods som förenar persondatoranvändarna som helhet.

Vad kan detta allmängods bestå av då?. Ny teknologi, branschförändringar, programspråk, systemering, användarreportage, förhållandet människa - maskin, trender, prisjämförelser, service, utbildning m m. Visst kommer vi att fortsätta att testa utrustning men varje nummer kommer fortsättningsvis att innehålla en god portion allmängods.

Samarbete med ABC-klubben

Maskinspecifika artiklar passar bäst i klubbtidningar som denna. Vi tror därför att MikroDatorn och respektive klubbtidningar kan ha stor nytta av samarbete. MikroDatorn kommer ut regelbundet och oftare än ABC-bladet, vilket gör att MikroDatorn är en bra informationskanal för ABC-klubben när det gäller olika aktiviteter och erbjudanden. Därför upplåter MikroDatorn varje månad utrymme till ABC-klubbens redaktion som fritt kan förfoga över denna anslagstavla för interna meddelanden (normalt en halvsida).

Prenumerationserbjudande

För att detta skall vara en praktisk lösning förutsätts givetvis att de flesta medlemmarna läser MikroDatorn. Jag tror att många redan gör det men för att uppmuntra fler att läsa MikroDatorn erbjuder jag härmed alla ABC-klubbens medlemmar att prenumerera ett år till priset 145 kr, vilket är halva lösnummerpriset! Erbjudandet gäller tills vidare.

Att MikroDatorn blir mindre maskinorienterad innebär förstås inte att vi är ointresserade av alla som har en ABC-dator, tvärtom. Men det som intresserar redaktionen är information som kan intressera även icke ABC-ägare, t ex speciella branschlösningar eller liknande där själva tillämpningen i sig är intressant. Jag uppmanar därför både leverantörer och användare som känner igen sig här att höra av sig till MikroDatorns redaktion. Adressen är:

MikroDatorn
CW/Communications
Södra Hamnvägen 22
115 41 Stockholm

Namnet CW/Communications kräver kanske sin förklaring. Det är så att Nova Media, som tidigare ägde MikroDatorn, Computer Sweden och Svenska PC World numera har sålt sina datatidningar till det amerikanska jätteförlaget Computerworld. Det innebär att MikroDatorn ingår i världens största datatidningsförlag och har över 50 systertidningar runt om i världen. Givetvis utväxlar vi nyheter med varandra (över satelit) och därför kan du vara säker på att vi har god kontroll över vad som sker på den internationella datamarknaden.

Peter Elmlund
redaktör MikroDatorn

Mikrodatormässan

I år ägde mikrodatormässan rum ovanligt tidigt, redan i januari. Detta för att man inte ville ligga för nära Data Kontor Miljö 85 som äger rum den 28 september - 2 oktober på Stockholmsmässan.

Vissa priser kan vara inaktuella nu.

Bland nyheter fanns en 5 1/4 " superfloppy till ABC-datorer som lagrar 2,5 Mbyte per skiva och en mycket kraftfull supermikro från Myab. En del text har jag hämtat från olika broschyrer.

ARDO

Ardo presenterade superfloppyn RD312 som är en 5 1/4 floppy med två drive:ar. RD313 lagrar 2,5 Mbyte per 5 1/4 skiva eller 9600 sektorer! Drive:arna är av fabrikat Drivetec och flexskivorna är förformaterade och beräknas kosta cirka 100-150 kronor.

Pdatas program finns redan klara på floppys till RD312 och Luxors standardprogram kommer att finnas tillgängliga på dessa flexskivor.

Ardo har utvecklat ett diskoperativssytem som är kompatibelt med Diabs UFD-DOS. Ardo har också gjort ett så kallat turbokontrollerkort som gör att floppyn blir ca 8 gånger snabbare än en vanlig floppy. ABC80 skall också gå att köra!

RD312 kan läsa såväl 830 som 832 skivor, men inte skriva på dessa. Floppyn känner automatiskt av vilken typ som sätts i. Detta tar dock några sekunder, så om man vill så kan man slå om en switch på kontrollerkortet så att drive:en ställs in för en viss typ.

Data:

160 spår/sida, 2 sidor, stegtid 3 ms spår till spår, överföringshastighet 500 Kbits/sek, rotationshastighet 360 rpm, vikt 5 kg, dimensionener (b * h * d) 242 * 132 * 300 mm, kraftförsörjning 220V/110V, 50-60 Hz, effektförbrukning 30W-50W och arbetstemperatur 5 - 30 C. Pris 19805 exl moms.

ARDO har också winchestrar med fasta och löstagbara skivor. Drive:arna är av fabrikat Syquest. En löstagbar winchesteriva lagrar 5 Mbyte och själva skivan kostar 980 kronor. Fasta winchestrar finns i storlekar 5, 10, 20 och 30. Dessa kan kombineras med en löstagbar skiva.

Samma typ av DOS som till RD312 används, vilken innebär att man även kan köra den på ABC80. Dessutom kan man använda den mot ABC-NET och CAT-NET. Troligen kan man även använda RD312 mot dessa nätverk.

Data för fast och löstagbar 5 Mbyte:

306 spår/sida, 2 sidor, 19584 sektorer, 3 ms stegtid spår till spår, överföringshastighet, vikt 5 kg, dimensioner 218 * 132 * 300, kraftförsörjning 220 V 50 Hz, effektförbrukning 30-50 W och arbetstemperatur 5 - 30 C.

För 10, 20 och 30 Mbyes fasta winchesterdiskar gäller samma data, utom att dessa har 612 spår/sida och sektorerna är 39168, 78336 samt 117504.

Pris för en enhet med en löstagbar winchesterskiva är 19970 och för en fast 17330, exkl moms. Kombinationer kan göras. T ex så kostar en 30 Mbytes fast och 5 Mbyte lös 39890.

Tranfor och TDX

DataDisc 86 är en 8" slim line floppy i en låda som är något större än ABC806:s datorlåda, pris 19800. Flera 3" floppy finns nu, dels DataDisc32 som lagrar 2 * 160 Kbyte och kostar 9500, dels DataDisc32/1 med kapaciteten 1 * 160 Kbyte, pris 7600 samt DataDisc34 med en lagringskapacitet av 2 * 320 Kbyte, pris 10700.

Tranfor har också en DataStore 10/St som innehåller en fast 10 Mbyte plus en löstagbar 5 Mbyte, den kostar 40200. Ardos motsvarande kostar dock bara 35240! I Tranfors pris ingår dock en licens till Diab, vilket inte Ardo betalar eftersom deras är helt egenutvecklad vad gäller DOS och kontroller.

TDX och några andra visade ABC1600, dock var ingen applikationsprogramvara klar, förutom ordbehandling med LEX och kalkyl med Q-calc samt Mimers applikationsgenerator.

Luxor

Trots att Luxor inte var med på mässan skriver jag lite om om Luxors produkter.

I alla floppydiskar sitter nu ett nytt och snabbare kontrollerkort utvecklat av DIAB. Det gäller även Tranfors diskar. Normalt säljs dock inneliggande lager först ut, med gamla kort. Kortet är samma för ABC830, ABC832 och ABC838. Det är snabbt, det skriver inte i förväg som Myabs UNIDISK. Detta gör att det är långsammare vid skrivning än UNIDISK, men i övrigt är det ungefär lika snabbt, tror jag.

Biblioteket läses in så fort man sätter in en skiva. Detta är en finess som UNIDISK saknar.

Luxor gjorde en del prisändringar i december 1984. Normalt brukar man höja priserna, men denna gång sänkte man också. Detta gäller framför allt floppydiskar.

ABC80 kostar nu 2820:-. Produktionen av ABC80 är nedlagd, så att detta verkar vara utförsäljning. Dock räknar Luxor att med nuvarande orderingång ha ABC80 för leverans under hela 1985. Luxor fick lägga ner produktionen för att klara tillverkningen av ABC1600! Senare under sommaren har man börjat sälja ut ABC80 till ett pris som ligger betydligt under 2820!

ABC800 M kostar nu 7750. Oförändrat pris på ABC802, dvs 8150. ABC806 är sänkt något, till 10825. ABC1600 kostar 60680 (datorn).

ABC812 färgskärm kostar 7250 och ABC815 monkrom skärm kostar 4465. ABC1615, dvs bildskärmen till ABC1600, kostar 8715.

Tangentborden kostar 1075, 2350, 2860 och 3375. Priserna avser ABC22, 55, 77 och 99.

ABC830 disken kostar nu 7820, ABC832 11050 och ABC834 (nya slim line) 12550. ABC838 kostar 20930. ABC850 10 Mbyte winchester med en 640 kbytes drive och expansionslåda kostar 31680.

LUX-NET centralenhet kostar 13500 och nodkorten inkl linjeanslutning kostar 3800.

Summerar man kostnaden för ABC1600 så kostar hårdvaran 72770. Den programvara som finns nu är dyr. Nectar kostar 8000, Mimer OL/SH 9000, Lex-68 8900 och Q-calc 6000.

Liber

I Fritzes monter hittade jag två broschyrer om färgrafik och BASIC II till ABC80. Pris för färggrafkien är 2500-2600 och för BASIC II 2500 exkl moms.

På BASIC II kortet sitter det samma tolk som i ABC802, samma skrivarrutiner, samma skivhanteringssystem (UFD-DOS) samt 64 Kbytes RAM. BASIC II kortet är hårdvarumässigt förberedd för CPM.

CAT

CAT visade CAT-NET och Luxors nya skolpaket där CAT-NET ingår. Dessutom visade man ABC1600 som dock ännu inte kan köras mot CAT-NET.

Owoco

Owoco visade Smartaid800, se annons i ABC-Bladet nr 4, 1984. Smartaid800 levereras på kort. Saknar man expansionslåda så kan man få problem med nätaggregatet. Strömmen räcker om man har det nya diskkontrollerkortet, som är strömsnålare än det gamla.

Owoco visade också en 3 " floppy, specialgjord för ABC802. Pris är ännu inte fastställt.

Myab

Myab visade ABC16 och en prototyp till en supermikro utvecklad av Myab i Göteborg.

ABC16 beräknas bli klar för leverans i februari, vilket innebär att den bör vara klar när detta läses. Program för PC-DOS version 2.1 kan köras först när en ny version av Concurrent PC-DOS kommer, vilket beräknas komma mars 1985. Denna uppdatering av operativsystemet i ABC16 kommer att ske kostnadsfritt för kunden.

Både en ABC830 och ABC832 drive kan läsa IBM PC skivor. På ABC832, som har 80 spårs skivor, går detta till så att vart annat spår läses! Det är inga problem att läsa IBM PC skivor med ABC16, men att läsa ABC16 skivor på IBM PC kan vi vissa fall bli svårt. Detta beror på att spårinställningen kan vara lite olika. Skall man göra program för IBM PC på ABC16 och kopiera sina skivor för IBM PC på ABC16, så föreslår Myab att man i stället köper en IBM PC.

Överföring av filer mellan ABC och ABC16 är ännu inte löst, men Myab räknar med att lösa det.

Till ABC16 har Myab en hel del program, bl a Word Star, dBASE II, SuperCalc III, Lotus 1-2-3 och Multiplan samt flera programmeringspråk (kompilatorer).

Myab System 32000 (My-32) är namnet på Myabs supermikro. Det unika med systemet är att det kan arbeta med upp till 8-10 processorer parallellt! Systemet blir därmed mycket kraftfullare än ett enprocessorssystem. Programvaran är "äkta" UNIX system 5.2 från Bell.

Systemet är moduluppbyggt och består av följande:

Processorkort NS 32016 eller 32032 med NS-32082 (MMU) för virtuellt minne och för anslutning till ett fler-cpu-system. Varje processor har en egen flyttalsprocessor, NS-32081, för att snabba upp hanteringen av flyttal.

Minneskort med plats för 3 Mbyte minne per kort.

Hårddiskkort för anslutning av 1-8 st hårddiskar. I stället för skiva kan även en

minikassettstation eller floppy anslutas. Kortet är speciellt anpassat till My-32 och UNIX och använder bl a spårbuffringsteknik för att uppnå maximala prestanda.

Hårddiskar med storlekar från 65 Mbyte till 191 Mbyte. Även större skivor kommer inom kort.

Minikassettstation för lagring av data på 1/4 " band. Lagringskapaciteten beror på bandstorleken och är fn 45 respektive 60 Mbyte. Bandstation för 1/2 " band kan anslutas.

Asynkront seriekommunikationskort med 8 st V24 linjer. Sättbara överföringshastigheter på upp till 38400 bps. Terminalkortet använder DMA (Direct Memory Acess-skrivning direkt i minnet). En hel del av teckenhanteringen som normalt sköts av huvudprocessorn i vanliga datorsystem, sköts här av den lokala processorn på terminalkortet.

Anslutningskort för Ethernet för att koppla samman flera system.

Synkront seriekommunikationskort för BISYNC, SDLC och HDLC. Mjukvarustöd finns också för RJE.

I grundutförande består ett My-32 system av ett processorkort med NS-32016, ett minneskort med 768 KByte, ett terminalkort för 8 terminaler, ett diskkort med en hårddisk på 65 MByte och en minikassettstation avsedd för säkerhetskopiering. Allt detta beräknas kosta cirka 150000 SEK.

Hårdvaran är anpassad för multiprocessorhantering, dvs flera processorer kan anslutas till ett system. Dessa processorer kommer då att dela på den totala belastningen. För användaren är systemet helt transparent och en användare har ingen möjlighet att avgöra hur många processorer som är anslutna. Användaren tror sig se samma maskin oavsett om en, två, tre eller flera processorer är anslutna! 8-10 processorer är vad man optimalt kan använda i ett system.

My-32 bygger på chip från National Semiconductors serie NS-32000. NS-32000-serien har minidatorn VAX-11 som förebild. My-32 är en sann 32-bit maskin, alla register har längden 32 bitar. Alla adresser är 24 bitar långa, vilket innebär att varje processor kan adressera 16 Mbyte minne.

My-32 använder riktig UNIX och inte kopior som andra, bl a Luxor/Diab. Det innebär att My-32 direket kommer att få operativsystemet uppgraderat när nya versioner av UNIX kommer ut.

Med My-32 levereras också UNIPLEX, som är ett menyhanterinsprogram med texthanterare, spreadsheetkalkylator, databashanterare. Man kan dessutom lägga in egna program i menyerna.

EMH-DATA

EMH tillverkar bl a EMH802 som är en industriversion av ABC802. Det var EMH som gjorde det autostartsdos som tidigare användes till klubbens gamla monitor.

Nu har EMH ett utvecklingssystem för BASIC II program som heter DIADEMH, vilket står för DIAlog Design EMH.

DIADEMH är ett verktyg för den som programmerar BASIC på ABC800 seriens datorer dvs 800, 802, 806 etc. Det är utvecklat för att lösa programmeringsproblem på ett enkelt och enhetligt sätt och har använts regelbundet för programutveckling av EMH sedan 1983 och kontinuerligt förbättrats under tiden.

DIADEMH består av två huvuddelar:

Ett stort antal utprovade funktioner skrivna i BASIC, avsedda att användas som byggklossar vid programmering. Här återfinns t ex rutiner för att göra menyval, rita formulär på skärmen, låta användaren fylla i formulären, redigering av utskrifter och alla funktioner som behövs för att administrera ISAM-filer.

Ett menystyrt programsystem för redigering av skärm- och utskriftslayouter, definition av inmatningsformulär, menyer och datafiler och testkörning av menyer och formulär.

Grundiden är att man redan vid systemskissandet använder DIADEMH för att rita upp menyer, formulär och exempel på utskrifter. Dessa bilder används senare om mallar för det slutliga systemet. De kan också förses med förklarande texter, och på så sätt vara en del av dokumentationen.

Man kan snabbt få en ide om hur det blivande systemet kommer att se ut, och även till en viss del provköra menyerna och formulären, fylla i sina egna testdata och se att kontroller görs på rätt sätt.

Eftersom man ännu inte har börjat programmera är det enkelt att ändra i skissen tills alla är nöjda.

Så småningom är det dags att börja skriva program. För att förenkla denna del av jobbet finns dels de färdigskrivna funktionerna, dels ett antal programstommar. Dessa stommar är nästan färdiga program, som bara behöver några ändringar (kanske en massa tillägg också, men stommen är densamma) för att blir kompletta.

Stommar finns för menyvalsprogram, ajourhållning av register, utskrift av ett register och städning av register. DIADEMH kostar 10000.

Övrigt på mässan

En hel del programvara fanns. Bland hårdvaran märkes IBM AT och DG One. DG One är den första riktigta PC:en som kan drivas på batteri. I England utvecklas ett inbyggt kombimodem 300/300 och 1200/75 modem. Svenskt tangentbord är på väg till One.

Bo Kullmar

Att köra en mikrodator som en asynkron terminal

Vill man använda en mikrodator som terminal så måste man ha en terminalrutin i datorn. En sådan rutin efterliknar någon känd terminaltyp och kallas därför för emulator. En populär benämning som jag har sett i KOM är termulator.

En enkel, men användbar terminalemulator är ADM3A. En mer avancerad emulator är VT100. VT100 som egentligen är en av DEC:s terminaltyper. När man kommunicerar med en stordator så måste man normalt tala om för datorn vilken terminalrutin som man använder. Därmed vet värddatorn vilka kommandon som skall skickas för att terminalen skall bete sig på önskat sätt.

Kör man mot klubbens monitor så förutsätter vi inte en viss termulator. Vi använder därför inte några styrtecken som inte alla terminalemulatorer klarar av. Man behöver därför inte tala om för monitorn med vilken emulator man kör med.

Begreppet "asynkron" som används i rubriken syftar på överföringsättet. Även "synkron" överföring förekommer, men främst när man kommunicerar med stordatorer typ IBM. Kör man synkront måste man ha andra emulatorer än de som nämns här. T ex en 3270 emulator som efterliknar en av IBM:s terminaltyper. Dessutom måste man ha ett speciellt modem för asynkron kommunikation.

ABC80 som terminal

Kör man ABC80 som terminal är man i princip hänvisad till två terminalemulatorer, nämligen ABCV24 eller T80PRT och TERM100. Båda är medlemsförmåner i ABC-klubben. ABCV24 emulerar ingen känd terminaltyp, därför kan man inte köra program som kräver en viss emulator. TERM100 är en VT100 emulator, som naturligtvis har vissa begränsnigar, men det har många andra VT100 emulatorer också.

Kör man ABCV24 så skall man först ladda ABCV24. Det räcker med standardparameterar, förutom att man bör ange radlängden till 255 för annars klipps långa rader av vid överföring.

Observera att hastighet och pariet sättes i det program man använder, t ex FILTRANS eller ABCTRANS. Där står idag troligen "V24:KA.1" och "V24:KB.1" i början på programmet. Vill man köra mot monitorn så bör man ändra K mot H. Detta innebär att pariteten sättes till space eller noll. Vill man sedan köra 1200/75 så skall man ändra 1 till 6. För att köra 1200/1200 så sätter man siffran till 3. 1 betyder alltså 300 bps.

Normalt skall man köra med ekning mot monitorn ("full duplex"). A i parametern står för ekning. Observera att ABCV24 inte fungerar så bra i 1200/75. Detta beror på att programmet inte hinner att eka tillbaka tecken om man inte skriver mycket långsamt. Det beror på att man sänder långsamt och sedan mottager snabbt. Använd därför TERM100 i stället.

Du skall välja paritet 0 i TERM100. För övrigt så hänvisar jag till den utförliga dokumentationen som finns till TERM100. Den är publicerad i ABC-bladet nr 2, 1985. Den version av TERM100 som finns på kassett nr 15 kan även köras på en 16 KB maskin.

Kör du TERM100 mot vår monitor så kan du inte hämta eller skicka filer med GET/SEND. Detta kräver ett BASIC program av typen FILTRANS som sköter om handskaknigen. Du kan i stället skriva ut filen med "TYPE <filnamn>" och sedan dumpa den lokalt till fil. På motsvarande sätt kan du skicka in filer med WRITE, om de inte är alltför långa.

Vill man köra wiewdata (Datavisonen) med ABC80 så finns det två terminalrutiner. Dels en gjord av VINVENT AB, se annons i ABC-bladet nr 3 sid 27 och dels en som finns upptagen i Luxors prislista över ABC80 program. Luxors rutin verkar billigare, men den är då troligen enklare än Vinvents eftersom man med Vinvents program även kan göra teledatabilder.

Dator ur ABC800-serien som terminal

Har du en dator ur ABC800 serien så kan du välja mellan att köpa en terminalrutin eller använda den ADM3A emulator som levereras med. För ABC800 C/M och ABC802 finns ADM3A emulatorn i själva datorn och det är mycket enkelt att köra. För ABC806 finns ADM3A-rutinen på skiva och det är ganska komplicerat att preparera en skiva så att du kan köra som terminal. Detta beskrivs dock nedan, med utgångspunkt från systemprogram på Luxors systemskiva.

Vill du köpa en terminalrutin så kan du köpa ABCUTE eller någon VT100 emulator. Autocode säljer en VT100 emulator där man får KERMIT på köpet. Är man intresserad av Autocodes terminalrutin så ta kontakt med Torbjörn Alm på Autocode. Också DIAB har gjort en VT100 emulator.

Väljer du att använda en annan terminalrutin än den inbyggda ADM3A emulatorn så kan du inte hämta eller skicka program med GET/SEND till monitorn. Då bör du i stället skriva ut filen med "TYPE <filnamn>" och sedan dumpa den lokalt till skiva med hjälp av din emulator. På motsvarande sätt kan du skicka in filer med WRITE, om de inte är alltför långa. Man måste observera att även om man kör med ADM3A emulatorn så måste du använda ett speciellt BASIC-program typ FILTRANS för att kunna hämta och skicka in filer med GET/SEND.

Vill man köra wiewdata så går det bra med den inbyggda terminalrutinen, bara ändra T:et till ett W. Detta går dock inte på ABC800 M som i stället fordrar ett specialprogram som använder HR-grafiken för att rita bilderna. Det finns också kommersiella wiewdataprogram från bl a Luxor, med vilka man även kan skapa bilder.

ADM3A emulatorn till ABC800 M/C och ABC802

I dessa 800:or finns det en inbyggd terminalrutin som kan köras direkt. Därför är det mycket enkelt att köra ABC800 eller ABC802 som terminal. Hur du skall göra framgår av punkt 6 nedan, i beskrivnigen för ABC806.

ADM3A emulatorn till ABC806

Vill man använda ABC806 som terminal utan att köpa ett terminalprogram så kan man använda den rutin som normalt finns inbyggd i vanliga ABC800:or. I ABC806 finns dock denna rutin på systemskivan, eftersom den inte fick plats i optionsprommet i datorn.

Terminalrutinen skall laddas vid reset och därför skall du formatera en flexskiva som du skall använda när du skall köra terminal.

1 Formatera en ny skiva och sätt skivan i drive 0.

2 Kopiera filerna BASICINI.SYS, ADDOPT.ABS, DEVDES.REL och TERMOPT.REL till skivan från systemskivan. Saknar du några filer så kontakta din återförsäljare.

3 Ändra i filen ADDOPT.ABS på terminalskivan. Detta kan du göra genom att köra programmet CONFIG.UFD som fanns på kassett nr 13 eller genom att ändra direkt i ADDOPT.ABS med POSIT.
Med CONFIG:
Kör du CONFIG så är det enkelt, då skall du bara svara ja på frågorna om "Datavision/ADM3A emulering" och "Läsa in drivetabell".
Utan CONFIG:
Har du inte tillgång till CONFIG så kan du ta programmet SETOPT.BAC som finns på systemskivan. På sid 12 i Luxors manual till UFD-DOS:et hittar du lite anvisingar.
Vad du skall göra är att skriva in två filnamn. Det står troligen redan en rad med DEVDES, så låt den stå och lägg till en rad med TERMOPT.REL. Observera att filnamnet måste ha ett speciellt format! Längden på filnamnet måste vara 11 tecken. Inga gemena och ingen punkt får ingå. Filnamnet måste börja i position 1 och filtypen måste börja i position 9. Överblivet utrymme måste fyllas med blanktecken.
Exempel:

```
10 OPEN 'ADDOPT.ABS' AS FILE 1
20 PUT §1,CHR$(255)
30 PUT §1,CHR$(255)+'DEVDES  REL'
40 PUT §1,CHR$(255)+'TERMOPT REL'
50 PUT §1,CHR$(254)
60 CLOSE 1
70 END
```

4 Nu skall du skapa autostart på filen ADDOPT.ABS som du just har ändrat. Detta kan du göra med programmet PREABS.BAC eller ett program av samma typ. Jag vet inte om programmet har samma namn på systemskivor till diskar som är tillverkade av Tranfor.
Kör alltså programmet PREABS och svara ja på frågan om autostart på maskinkodsfil. Ange sedan "ADDOPT.ABS" som maskinkodsfil. Observera att denna gång anger vi det med punkt. Sedan frågar programmet efter en kommandosekvens som skall utföras efter att maskinkodsfilen har exekverats.
Normalt skiver du "RUN DRINI" där. Vill du också ha EXTEND och INTEGER så får du skriva "INTEGER : EXTEND : CHAIN 'DRINI'". Den typ av enhet som DRINI körs från tolkas efter att programmet har körts som DR enhet. DRINI läser också av batteriklockan och ställer datorns klocka.

5 I DRINI finns sedan ett anrop till programmet START.BAC. Du kan om du vill sedan ändra namnet på det program som du vill köra till START eller gå in i DRINI och ändra anropet till START så att det i stället anropar ett annat program. T ex FILTRANS.

6 För att du skall kunna hämta filer från programbanken med GET och skicka in filer med SEND så måste du använda ett speciellt BASIC-program av typen FILTRANS. FILTRANS fanns på kassett nr 9. Skall du köra en annan hastighet än 300 bps, så måste du dock ändra i programmet FILTRANS så att det där står rätt parameter för den hastigehet som du vill köra.

```
Parameter:
TSA70C72.22E     för 300/300 bps
          40     för 1200/75 bps
          44     för 1200/1200 bps
          55     för 2400/2400 bps
```

Vill du enbart köra som "dum" terminal så kan du göra "LOAD V24:...".

7 När du har gjort detta så skall du trycka reset och därmed kommer terminalrutinen att laddas. Om du får fel nr 200 när du kör FILTRANS så har du inte lyckats att ladda terminalrutinen.

Överföring av filer mellan två ABC80/800:or

Vill man föra över filer mellan två ABC-maskiner så kan man använda programmet FILTRANS. Det går också att föra över filer mellan en ABC80 och en dator ur ABC800-serien. Varje dator måste då naturligtvis använda FILTRANS för ABC80 respekitve ABC800-serien. Detta förutsätter att man för ABC80 använder ABCV24 och för ABC800-serien ADM3A emulatorn som levereras med datorn.

Man ringer först upp varandra i vanlig "pratmode" och sedan bestämmer man vem som skall ha A och vem som skall ha B-kanal. På en del modem kallas A för "ORIG" och B för "ANS". Det spelar ingen roll vem som väljer vad, bara inte båda kör med samma. Har inget av modemen B-kanal så går det inte. Automatiska modem av typen TGC, kan komplicera det hela. Jag har dock ingen erfarenhet av TGC-modem.

När man har bestämt vem som skall ha A och vem som skall ha B-kanal så slår man på dataknappen och väljer halv duplex i FILTRANS. Nu kan man skriva text till varandra, men kom bara ihåg att inte skriva samtidigt. Man får också se till så att man efter varje RETURN även skickar ett CTRL-J för ny rad.

För att sedan föra över filer måste en av datorerna gå över i passiv monitormode. Detta kan man göra genom att välja det i menyn på en av datorerna eller genom att från den andra datorn skicka CTRL-B+ "MONITOR". Eftersom man på en dator ur ABC800-serien inte kan skicka CTRL-B i halv duplex med ADM3A emulatorn så finns texten inlagd på en Pf-knapp.

När en av datorerna har gått över i monitormode så är det bara för den aktiva att hämta eller skicka in de filer som önskas. Filerna måste vara i textformat (BAS-format). När den aktiva datorn sedan avslutar så kan man gå tillbaka till att skriva text till varandra eller slå om modemet till "TALK" och prata på människors sätt!

Selics modem

Använder man något av Selics moderna modem så skall man se till så att att switch 7 inne i modemet är på (on) för annars kan modemet koppla ner förbindelsen i oönskat läge.

Vill man styra nedkoppligen av V24:an från datorn så skall switch 7 vara off, men då får man se till så att man inte stänger V24:an eller att man sätter den filen permanent. Gör man detta så stängs enbart filen om man gör CLOSE <filnr>. Hur man sätter en fil permanent framgår av "Bit för Bit" sid 98. Det är byte 3, bit 1 i filparameterblocket som skall sättas.

Televerkets Folkmodem

Signalen CCITT 105 skall alltid vara på (on). Detta kan man göra i sladden genom att koppla stift nr 1 i den 25 poliga kontakten till stift 1 i den 9 poliga kontakten. Se ABC-bladet nr 2, 1983 sid 24, där det också finns en skiss.

Bo Kullmar

Lite om Datel, Datex och Datapak

För överföring av data via ledningar finns det flera olika nät. Kör man "data" med vanliga telefonlinjer så kallas detta numera för Datel. Datel kan köras som Datel Uppringt eller Datel Fast. Datel Fast innebär att man hyr en fast linje från Televerket. Din vanliga telefonlinje kan du använda för att köra Datel Uppringt.

För att minska belastningen på telefonlinjerna så har Televerket under senare år satsat mycket på speciella nät för dataöverföring. Först och främst har man satsat på Datexnätet, men senare även på Datapak. Dock kommer Datel Uppringt att även i fortstättningen ha hand om en hel del av datatrafiken, eftersom det under vissa förutsättningar är billigt.

Datex

Datex är liksom det vanliga telefonnätet ett kretskopplat nät (eng circuit switching). Detta innebär att linjen är uppkopplad hela tiden medan samtal pågår. Upp och nedkoppling går dock mycket snabbt. För att köra Datex krävs det speciella modem.

Den mest kända användaren av datexnätet är nog de blå Bankomaterna. Vid starten 1981 fanns det 700 anslutningar, av vilka de flesta var Bankomater. Som kuriosum kan jag nämna att Bankomaterna kommunicerar med så låg hastighet som 600 bps. Redan tre år efter starten har man passerat 10 000 anslutningar och man räknar med en dubblering om cirka 1 år.

Datex eller NPDN (Nordiska Publika Data Nätet) täcker idag hela Norden och ger även möjlighet till kommunikation med Västtyskland. I Österrike, Japan och Kanada finns också kretskopplade publika datanät.

Datapak

I de flesta andra länder har man dock satsat på datapaketförmedlande nät (eng. packet switching). I Sverige infördes ett litet paketförmedlingsnät i form av en växel (nod) i Stockholm. Detta var främst till för att man i Sverige skulle kunna nå databaser i USA. Utrustningen köptes begagnad från Tymnet i USA. Tymnet är en av de stora paketförmedlande näten i USA. Telenet är det andra stora nätet i USA.

I ett paketförmedlande nät samlas data som skall sändas i paket. Varje paket innehåller adress och användardata och sänds sedan genom nätet med optimal hastighet. Sändare och mottagare är alltså inte som i ett kretskopplat nät förbunda utan paketen skickas som "godsvagnar på räls". Paketen kan till och med gå olika vägar och hos mottagaren sammankopplas "tåget" ihop igen.

CCITTs rekommendation X.25 beskriver hur paketen skall se ut. I televerkets prislista kallar man ett paket för ett segment. Normalt är ett segment på 64 oktetter och en okett definieras som 8 bitar. Används mindre paket än 64 oktetter så baseras volymavgiften per påbörjat paket.

Kan man inte själv kommunicera med X.25 så kan man använda en sk PAD (eng Packet Assembly, Disassembly). En PAD samlar asynkrona tecken från användaren till ett paket för vidare transport i nätet och omvänt. En värddator måste dock kommunicera enligt X.25 vilket innebär i praktiken att man aldrig kan nå klubbens monitor via Datapak.

Dock kan man nå Datapak från en vanlig asynkron terminal i hastigheterna 300, 75/1200 eller 1200/1200, dvs Datel uppringt. Det går också att nå Datapak via Datex och Datel Fast.

Nu bygger Televerket ut Datapak med ny utrustning från Tymnet i USA. I samband med detta har avgifterna sänkts. För att skaffa konto för Datel Uppring eller Datex får man betala 500 kronor och sedan 180 kronor i abonnemangsavgift per kvartal.

Om man går in via Datel Uppring och Datex är trafikavgiften följade:

öre/påbörjat segment öre/påbörjad min Öre/anrop			
Inom Sverige	20	45	0.5
Till Norden	20	65	2
Till övriga Europa	20	80	2.5
Till Nordamerika.	20	120	5
Till övriga länder	20	190	8

Dessutom tillkommer för Datel Uppring avgift för en telefonmarkering och för Datex anropsavgift. Tid och volymavgifterna sänks med 25% vardagar 18.00-08.00 samt lördagar och söndagar 00.00-24.00. Dessutom finns mängdrabatt vid stora volymer.

Men som bekant kan det kosta betydligt mer att ringa riks över det vanliga telefonnätet. Jämför 173 öre med 45 och 87 öre med 34 öre under lågpristid. Därutöver får man dock betala lite per paket vilket man ju slipper om man ringer på telefonnätet. QZ:s datorer i Stockholm tillhör dem som man kan nå via Datapak. Inom högskoleområdet finns det ett speciellt nät kallat SUNET. SUNET använder sig av Datapaknätet för kommunikation. Taxorna är dock annorlunda såvitt jag förstår. Allmänheten kan dock inte få ett SUNET-konto.

Skall man köra på en dator i USA, så är nästan Datapak ett måste ur kostnadssynpunkt. Att ringa till USA kan kosta cirka 10 kronor i minuten medan att köra Datpak kostar bara drygt 1 krona!

Satellitkommunikation

Alla dessa nät kräver ett lokalnät, dvs förbindelser ut till abonnenterna. Det är detta lokalnät som är dyrast. För framtiden finns det dock ett kommunikationsmedium som kan hoppa över detta lokalnät och även lång- och kortdistansnäten, nämligen satellitkommunikation! Genom att sätta upp en egen parabolantenn på taket så kan man kommunicera direkt med en Satellit.

En nackdel med Satellitkommunikation är dock den fördröjning på cirka 1/2 sekund som uppstår på grund av att satelliterna befinner så långt ut från jorden. Detta måste de göra för annars blir de inte geostationära, dvs de stannar inte kvar på en punkt på himmeln. Att de ligger så långt ut gör att radiovågerna måste ha relativt lång tid på sig att först komma upp till satelliten och sedan ner till mottagaren.

Om man bortser från fördröjningen så är det fullt tänktbart att kommunicera med ett grannhus via satellit utan att blanda in televerket. Om sedan telemonopolet tillåter det, är en helt annan sak.

Visste du förresten att du inte får dra en kabel till din granne och använda den för telefontrafik! Det är ett brott mot Televerkets monopol! Internt i en fastighet har dock ett företag rätt att dra egna linjer för kommunikation.

Bo Kullmar

En smartaid® födelse

Då ABC80 utvecklades lades vikten på en kraftfull och snabb BASIC-interpretator vilket dock innebar att redigering och andra hjälpmedel vid programmering inte kom i första ledet. I och med detta föddes iden till att göra ett program som skulle fungera om ett verktyg vid utveckling och provning av program i BASIC.

Det första programmet i denna genre skrevs av Henrik och Felix Burton och kallades AUTO och blev färdigt hösten 1980. Grundiden med AUTO var att alltid ha de nya kommandona tillgängliga vid programmering vilket löstes genom att utnyttja atorns tangentbordsinterupt på ett elegant sätt.

Hårdvaran utvecklades av Bengt Sjöqvist och Bernt Lindgren, "ett orange munspel" sattes bakpå ABC80:n vilket innehöll 1 kilobyte programvara. Den hade funktioner som bildskärmseditor, AUTO-numrering, START, DELETE, och VAR (variabellistning). Nackdelen med SMARTAID, som den kom att kallas hos OWOCO, var att den bara fungerade på en version av BASIC-tolken, vilket föranledde utveckling av en helt ny programvara gestaltad i SMARTAID II. Denna programvara och senare versioner av den är skrivna av David Andersson (som för övrigt gjort och skänkt ABC-klubben programmet TERM-100) och Ulf Söderman. SMARTAID II hade bl.a. en ny och förbättrad bildskärmseditor och en rad nya kommandon samt fungerade på alla versioner av BASIC-tolkar till ABC80.

Våren -82 kom SMARTAID III samt SCHOOLAID på begäran av Luxor och Liber, den senare en bantad version av SMARTAID III och såldes av liber till skolorna. Dessa två varianter byggde på en helt ny hårdvara gjord av Christer Ekman och blev den för Smartaidarna karakteristiska svarta lådan bakpå bussen och innehöll 5 respektive 4 kilobyte programvara.

Nu gav målsättningen att inte inkräkta på användarens minnesutrymme eller andra periferienheter stora problem. Mycket arete lades ner på att finna var olika programvaror såsom printerrutiner etc. utnyttjade de "fria" bitarna i bildminnet (SMARTAID III lagrar bl.a. senaste radnummer för AUTO-numrering i dessa minnesceller) samt anpassning till fyra olika DOS och hänsynstagande till TKN-80 och andra tillbehör. Detta för att undvika att produkterna störde varandra. En lätt lösning på många av dessa problem hade varit att göra ingrepp i datorn, men vi valde att göra den dyrare mjukvarulösningen som gav en bättre produkt för användaren.

Nyheterna i dessa versioner av SMARTAID var bl.a. LIST fram och baklänges samt möjlighet att lista valda delar till printer, OLD framtagning av programmet efter RESET med fullständig syntaxkontroll, inbyggd printerrutin, HELP, CHANGE byte av variabelnamn, PEEK listning av minnesinnehåll i decimalt och hexadecimalt format, LIB visning av innehållet på skivor, SYS visning av alla viktiga systemvariabler och status m.m.

En annan nyhet var att SMARTAID III automatiskt initieras vid spänningstillslag eller reset. SMARTAID III har dessutom FOR-NEXT loop i kommandomod, Hardcopy (dumpning av bildskärmen till skrivaren), TRACE-funktion för single-step av program, m.m.

Flaggskeppet i SMARTAID-serien, SUPERSMARTAID, innehåller 10 kilobyte programvara och 2 kilobyte CMOS-minne för att bibehålla viktiga systemparametrar som printerparametrar och användarens KEY-definitioner även när strömmen slås av. För att få plats med denna programvara var det nödvändigt att använda mappningsteknik för minnet (s.k. bankswitchning).

För att lösa detta problem och bibehålla SMARTAID som en fristående enhet och slippa ingrepp i datorn utvecklades en ny förnämlig hårdvara av Christer Ekman som bl.a möjliggjorde auto-start utan ingrepp i datorn eller byte av datorns DOS.

Mjukvaran gjordes av David Andersson och innehöll nya funktioner som DISP visning av filinnehåll på skärmen, JOB exekvering av kommandon från fil, KEY definiering av valfri text på tangent, TRACE med singlestep som visar aktuell BASIC-rad, TIME kontinuerlig visning av tiden på skärmen, SPOOL utskrift av en fil på printer samtidigt som programmering pågår, CONTinue och RESUME, editering av textfiler med TLOAD och TSAVE m.fl. Dessutom finns många andra finesser som man snabbt vänjer sig vid och sedan inte kan undvara, t.ex. den ytterligare förbättrade BILDSKÄRMSEDITORN.

När ABC800 kom gjordes en SMARTAID även för denna maskin med mjukvaran skriven av undertecknad och Håkan Jonsson samt Mikael Larsson. Hårdvaran utvecklad av Christer Ekman innehåller 8 kilobyte program och 2 kilobyte CMOS-minne.

Att vidhålla iden med en Smartaid som inte inkräktade på användarens minnesutrymme beredde stora svårigheter p.g.a. att ABC800 redan hade en full minneskarta samt redan använde mappning av vissa delar i tolken. Dessutom fanns inte alla önskvärda signaler tillgängliga på ABC800-bussen.

SMARTAID 800 har de flesta av SMARTAID III:s funktioner omgjorda för ABC800 såsom en förbättrad BILDSKÄRMSEDITOR, KEY-funktion, förbättrad LIB, PEEK listning av minnesinnehåll i HEX och decimalt, VAR variabellistning, FIND, EXC utbyte av kommando eller textsekvens mot en annan, LIST fram och baklänges och från senaste radnummer samt Hardcopy m.m.

En helt ny SMARTAID kommer att introduceras under hösten; SMARTAID MAGNUM. SMARTAID MAGNUM kommer att vara byggd för ABC80 och innehålla en förbättrad SUPERSMARTAID samt en helt ny programvara; en maskinkods-SMARTAID eller vad man kallar debugger.

Denna debugger gör det väldigt enkelt att lära sig maskinkodsprogrammering samt att felsöka i maskinkod då man hela tiden ser aktuell instruktion disassemblerad och status på alla flaggor i flaggregistret samt innehållet i samtliga register. Man har dessutom möjlighet att ändra status samt innehållet i alla register

För att snabbt kunna prova nya lösningar innehåller SMARTAID MAGNUM en miniassembler som assemblerar direkt i minnet. Debuggern är utvecklad av Magnus Stråle som tillsammans med David Andersson har implementerat den på SUPERSMARTAID. Även hårdvaran har uppdaterats och byggts ut för att klara det större minnesbehovet utan att stjäla minnesutrymme från användaren. Vi hoppas att i något senare nummer av ABC-bladet få möjlighet att beskriva SMARTAID MAGNUM närmare.

Som framgår av ovanstående sker hela tiden en utveckling av våra produkter i SMARTAID-programmet.

Vi anser att en beskrivning av bakgrunden till Smartaid och ideerna bakom Smartaid kan vara värdefull som komplement till olika jämförelser som gjorts; alla Smartaidar är osynliga från program och användarens synpunkt, innehåller en kraftfull bildskärmseditor samt är automatiskt initierande samt begränsar inte datorns användning för andra tillämpningar.

OWOCO AB
Bjarne Borg

Hur man kan använda ABC80 och SUPER-SMARTAID att datummärka filer

Att kunna datummärka sina filer är naturligtvis av stort värde. Om man har sin ABC80 med Super Smart Aid är det ganska enkelt.

Det hela bygger på att ha maskinkod + data för detta i CMOS RAMet. Då man startar ABC80 eller gör reset kommer SSA att leta efter filen SUPER.JOB och exekvera denna. Jag har gjort så att den först visar textfilen SUPER.TXT och gör RUN SETDATE. SETDATE visar det i CMOSRAMet lagrade datumet (Adr 22525-22527 som YY,MM,DD). Om det är dagens datum skriv endast <CR>, annars ange detta. Datumet pokas in i sina adresser i CMOS RAMet.

Nu är det så att filer som nyskapas på disken och gamla filer som överskrivs inte behandlas lika av DOSet. Därför behöver vi två olika procedurer. Vi börjar med nyskapade filer (SAVE;LIST;PREPARE) För att DOSet ska veta var på disken det finns ledigt utrymme, kopierar DOSet bitmappen, sektor 6, till DOS buffert 0, för kopiering av den egentliga filen användes buffert 1. Hur det går till i detalj struntar vi i nu, men DOSet ändrar buffert 0 till att bli det nya filhuvudet och skriver detta på skivan. Detta innebär att om dagens datum fanns på sektor 6, efter den egentliga bitmappen som inte tar hela sektorn i anspråk, så kommer även filhuvudet att innehålla detta datum. Detta kallar vi automatisk datummärkning.

Hur får vi nu datumet till bitmappen? I mitt fall genom att trycka Ctrl-É som startar en rutin på adress 21359 (DATESEC6.ASM). Rutinen sätter first drive 0, läser in bitmappen i buffert 0, kopierar dagens datum, som finns i CMOS RAMet till de tre sista adresserna (62973-62975) och skriver tillbaka bitmappen. Samma sak upprepas med drive 1. Denna datummärkning fungerar även med filer som nyskapas inuti ett program.

Här användes KEY-def. Ctrl-Å som gör CALL till maskinkod i CMOS RAMet på adr 21287 (DATEFILE.ASM). Maskinkoden gör följande : Avläs vilken drive som filen skrev till och sätt denna drive. Beräkna sektoradress genom byte4*256+(byte5 AND 224), i buffer 0, och lägg detta i DE reg, (argument till skrivrutinen) : Kontrollera att andra och tredje byte = 0, och den fjärde = 255 (då och endast då är det ett filhuvud i bufferten). Om detta stämde skriv buffert 0 till disken, sätt HL reg = 0 (OK) och gör return till KEY-def. Om det inte stämde sätt HL reg = 1, (skriv EJ) och gör return. Detta för att skydda mig själv om jag av misstag slår Ctrl-Å. Nu fortsätter KEY def, on HL (=Z%)=0 skriv 'OK'. Det är klart! Med denna metod spelar datumet i bitmappen ingen roll, den används aldrig. Rutinerna är assemblerade med ASM från ABC-klubben, den producerar ett BASIC program med poksatser.

Nu återstår problemet med filer som skrivs över (befintligt filnamn används). Innan SAVE, LIST eller PREPARE göres, kan man naturligtvis göra UNSAVE eller KILL, så är problemet löst. I annat fall får man göra en halvautomatisk datummärkning. Denna bygger på att då man har gjort SAVE eller LIST (eller LOAD), så finns filhuvudet för filen kvar i buffert 0.

Utility rutiner för att avläsa fildatum. Efter laddning av en fil (LOAD, TLOAD) kan Ctrl-O användas, denna KEY läser av de tre sista byten i buffert 0 (YY,MM,DD) och visar detta på skärmen. BASIC programmet CAT (en variant an LIB) är ändrat så att det förutom att visa dagens datum, klockslag, drive nr, skivnamn, filstorlekar även visar datumet för resp. fil. Filer utan datummärkning har datumet 00-00-00.

Bengt Larsson <2729>

```
10 REM setdate
15 ONERRORGOTO 70
20 Y%=1900+PEEK(22525%)
30 M%=PEEK(22526%)
40 D%=PEEK(22527%)
50 D$=NUM$(Y%)+'-'
55 M$=RIGHT$(NUM$(M%),2%) : IF LEN(M$)=
   1% THEN M$='0'+M$
56 D1$=RIGHT$(NUM$(D%),2%) : IF LEN(D1$
   )=1% THEN D1$='0'+D1$
57 D$=D$+M$+'-'+D1$
60 ; 'Date is: ';D$
70 ; 'Give date (YY,MM,DD): '; : INPUTL
   INE Y$
80 IF LEN(Y$)=2% THEN 180
90 Y%=VAL(LEFT$(Y$,2%))
100 M%=VAL(MID$(Y$,4%,2%))
110 D%=VAL(MID$(Y$,7%,2%))
120 IF Y%>99% OR Y%<1% GOTO 70
130 IF M%<1% OR M%>12% GOTO 70
140 IF D%<1% OR D%>31% GOTO 70
150 POKE 22525%,Y%
160 POKE 22526%,M%
170 POKE 22527%,D%
180 END
```

```
; TSAVE DATESEC6.ASM
; program that puts date on bit map,
; sector 6 on both drives
        ORG 21359
        LD HL,64769
        LD (HL),0     ; SET DRIVE to 0
        CALL DO
        LD HL,64769
        LD (HL),1     ; SET DRIVE TO 1
        CALL DO
        RET           ; END
DO:     LD DE,192     ; SECTOR 6 * 32
        CALL 24678    ; READ SECTOR
        CALL LOOP
        LD A,(22525)
        LD (62973),A ; PUT YY
        LD A,(22526)
        LD (62974),A ; PUT MM
        LD A,(22527)
        LD (62975),A ; PUT DD
        LD DE,192
        CALL 24675    ; WRITE BACK
        CALL LOOP
        RET
LOOP:   LD A,255      ; DELAY LOOP
LOOP1:  SUB 1
        JR NZ,LOOP1
        RET
END
```

```
; TSAVE DATEFILE.ASM
; program that puts date on file header
; used when old file is overwritten
; after saving use KEY function which
; make call to this routine checks if
; writing was ok and updates same drive.
; key def to 1
        ORG 21287
        LD A,(64769)
        BIT 5,A       ; disk select code
        JR NZ,UT      ; Writing failed
        RES 5,A       ; reset space is
                      ; to zero(drive no
                      ; is still there)
        LD (64769),A ; and write back
        LD A,(62724) ; GET SECTOR ADRESS
                      ; FROM
        LD D,A        ; DISKBUFFER 0
                      ; (62720)
        LD A,(62725)
        AND 224
        LD E,A        ; SECTOR NO IN DE
        LD A,(22525) ; GET YEAR (YY)
        LD (62973),A ; PUT YY IN BUFFO
        LD A,(22526) ; GET MM
        LD (62974),A ; PUT MM
        LD A,(22527) ; GET DD
        LD (62975),A ; PUT DD
        LD A,(62721) ; CHECK IF A FILE
                      ; HEADER
        CP 0
        JR NZ,UT
        LD A,(62722)
        CP 0
        JR NZ,UT
        LD A,(62723)
        CP 255
        JR NZ,UT
        CALL 24675
        LD HL,0       ; RETURN 0 =
                      ; NO ERROR
        RET
UT:     LD HL,1       ; RETURN 1 =
                      ; ERROR (NO FILE
                      ; HEAD)
        RET
END
```

```
rem KEY definition file KEYS.JOB M
KEY ^ = "  I  Q  L  Åü-  M" M
rem puts date in fileheader (not first
   time!) M
KEY  Å = "IF NOT call(21287) ;
   'ok'  M" M
JOB KEYS2.JOB M
rem datemark sector 6 M
KEY  ´ = "Z%=CALL(21359)  M" M
KEY  T = "RUN TERM100  M" M
KEY  O = ";'Datum:';peek(62973);
   peek(62974);peek(62975)  M" M
REM loading date save into CMOS RAM M
RUN DATEFILE.BAS M
RUN DATESEC6.BAS M
```

```
 100 REM MOD. VER. 4.00 * Flexnamn upp
     till 32 tkn & date *** För 80 och
     med SUPERSMARTAID
 110 REM Modifierad av Bengt Larsson, S
     OLLENTUNA 83-01-11
 200 ; '** ABC80 CATATOG Ver. 4.00 with
     file dates **'
 250 ; "Fil: "; : INPUTLINE F$ : IF LEN
     (F$)=2% THEN F8%=-1% ELSE F8%=0% :
     F$=LEFT$(F$,LEN(F$)-2%)
 255 IF NOT F8% GOSUB 8000
 260 F1%=INSTR(1%,F$,".") : IF F1%=0% T
     HEN F1$=F$ ELSE F1$=LEFT$(F$,F1%-1
     %) : F2$=RIGHT$(F$,F1%+1%)
 300 ; : ; "P - Printer" : ; "N - Fysis
     ka filnummer"
 400 ; "Optioner ......(Nej) "; : INPUT
     LINE O$ : ;
 500 L%=FNO%('P')
 550 IF L% ; 'Listfil ........(PR:) ';
     : INPUTLINE D9$ : D9$=LEFT$(D9$,LE
     N(D9$)-2%) : IF D9$='' D9$='PR:'
 600 PREPARE D9$ ASFILE L%
 700 REM F%=FNO%('S')
 800 N%=FNO%('N') : IF N%=0% THEN F%=1%
 900 W1%=40%
1000 IF PEEK(885%)=88% THEN W2%=81% ELS
     E W2%=41%
1050 GOSUB 7000 : REM GET DATE
1100 FOR D%=0% TO 6%
1200 G8%=0%
1300 POKE -767%,D%
1400 Z%=CALL(24678%,224%)
1500 IF PEEK(-747%) AND 128% GOTO 5500
1600 B%=-2577% : FOR I%=0% TO 7%
1700 M%(I%)=PEEK(B%+I%) : NEXT I%
1800 G%=0% : FOR P%=-2816% TO P%+160%
1900 IF PEEK(P%)=255% THEN 2300 ELSE IF
     PEEK(P%)=0% THEN G%=G%+8% : GOTO 2
     300
2000 Q%=1%
2100 G%=G%-((PEEK(P%) AND Q%)=0%)
2200 Q%=Q%+Q% : IF Q%<256% THEN 2100
2300 NEXT P%
2400 IF G8%=0% THEN G8%=G% : Z%=CALL(24
     678%,192%) : GOTO 1600
2500 X$="" : Z%=CALL(24678%,0%) : IF PE
     EK(-747%) THEN 5700
2600 FOR I%=-2592% TO -2561% : X$=X$+CH
     R$(PEEK(I%)) : NEXT I%
2700 GOSUB 5800 : ; §L%,D$'   'H$':'M$'
     :'S1$;'    Access: RWD = Read, Wri
     te and or Delete permitted'
2750 IF (ASC(X$)>31% AND ASC(X$)<128%)
     X$='   Flexskiva: '+X$ ELSE X$=''
2800 GOSUB 5800 : ; §L%'=== Drive §'CHR
     $(D%+48%)X$; : GOSUB 5800 : GOSUB
     5800
2804 IF F%=1% AND N%=0% THEN 2820
2805 ; §L%,'File          Fno    Access'
     ; : ; §L%,TAB(30%),'File       Fno
        Access' : GOTO 2890
2820 FOR I%=1% TO 2%
2830 ; §L%,'File          Date     Size
     Access      '; : NEXT I%
2890 ; §L% : ; §L%
2900 K%=0% : FOR S%=0% TO 7%
```

Det där med HJÄLPARE...

De flesta hjälpare har två typer av kommandon, de som aktiveras av en CTRL-sekvens och de som skrivs in direkt, som vilket annat kommando som helst. De förstnämnda är lättast att fixa, så vi börjar med dem.

När man trycker ner någon tangent på ABC80 genereras ett interrupt. Därvid hoppar processorn till en rutin som kollar om det var CTRL-C som trycktes, och sätter en flagga om så var fallet. Därefter sätts en interruptflagga för att indikera att en tangent är nertryckt.

Detta interrupt kan vi utnyttja för att få datorn att reagera på ett speciellt sätt på vissa tangenter, t.ex. en CTRL-sekvens. För att åstadkomma detta måste vi skriva en ny interruptrutin, och få datorn att gå till den, i stället för den vanliga. Rutinen måste vara skriven i maskinkod, och kan se ut som följande:

```
INTRUT  PUSH AF       ;RÄDDA FLAGGOR m.m
        IN   A,(56)   ;LÄS TANGENT
        CP   0        ;CTRL-É ?
        JP   NZ,801   ;OM INTE, HOPPA
                      ;TILL ORD. RUTIN
                      ;GÖR VAD SOM SKALL
                      ;GÖRAS VID CTRL-É
```

Rutinen här ovan reagerar på CTRL-É d.v.s ASCII-kod 0 (noll) från tangentbordet. Genom att lägga in fler CP-satser kan man få fler CTRL-funktioner.

En del CTRL-funktioner, t.ex en bildskärmseditor är det dock lämpligare att lägga in på ett annat vis, nämligen i inlinerutinen i basictolken. Detta kräver att vi skriver en helt ny inline-rutin, och skriver om kommandotolken. Den vanliga kommandotolken kommer ju att hoppa till den gamla inlinerutinen, och därför är den oanvändbar.

Kommandotolken

Den del av ABC80s basictolk som behandlar kommandon ligger på adress 204 i en ABC80 med checksumma 11273. För den som är intresserad av att göra en egen hjälpare kan jag rekommendera att titta på den adressen och framåt i t.ex ABC-rapport 1.

För att kunna lägga egna kommandon till de vanliga måste man skriva en helt ny kommandotolk. Samtliga kommandon finns upptagna i en lista med början på adress 2592 och deras adresser i en annan med början på 2650.

Om man har en 64K-utbyggnad och möjlighet att lägga tolken i RAM-minnet kan man göra om dessa listor och på det viset ordna en länk till några extrakommandon. Annars måste man göra helt nya listor, och en ny kommandotolk som använder de nya listorna.

En enkel kommandotolk kan ha följande uppbyggnad:

1 Hämta en rad från tangentbordet
2 Försök kompilera raden
3 Om det gick att kompilera, så sortera in raden i programmet och hoppa till 1

4 Kolla om raden innehåller ett kommando

5 Om inte kommando är raden felaktig. Ge ERROR och hoppa till 1
6 Leta upp kommandots adress i listan och exekvera det.
7 Hoppa till 1

Till sist: är du intresserad av att studera ABC80s basic närmare, eller av att göra en egen SMARTAID, så är det värt att lägga en hundralapp på ABC-rapport 1.

Lycka till med programmerandet och trevlig höst, eller vad man nu ska kalla det önskar

Ture Pålsson <5169>
Villagatan 20
940 45 VIDSEL
Tel. 0929-30109

```
3000 IF M%(S%)<1 GOTO 5200
3100 Z%=CALL(24678%,512%+S%*32%)
3200 IF PEEK(-747%) THEN 5700
3300 FOR B%=-2816% TO -2576% STEP 16%
3400 IF PEEK(B%)=255% OR PEEK(B%+4%)=0%
     GOTO 5100
3500 R%=PEEK(B%)*256%+PEEK(B%+1%)
3600 X$='' : FOR I%=4% TO 14%
3700 IF I%=12% THEN X$=X$+"."
3800 X$=X$+CHR$(PEEK(B%+I%)) : NEXT I%
     : Ö%=0%
3900 IF F8% THEN ; §L%,TAB(K%)X$; : Ö%=
     -1% : GOTO 4000
3910 IF F1%=0% AND LEFT$(X$,LEN(F1$))=F
     1$ THEN ; §L%,TAB(K%)X$; : Ö%=-1%
     : GOTO 4000
3920 IF F1%=1% AND MID$(X$,10%,LEN(F2$)
     )=F2$ THEN ; §L%,TAB(K%)X$; : Ö%=-
     1% : GOTO 4000
3930 IF LEFT$(X$,LEN(F1$))=F1$ AND MID$
     (X$,10%,LEN(F2$))=F2$ THEN ; §L%,T
     AB(K%)X$; : Ö%=-1%
4000 IF F%=0% AND N% AND Ö% ; §L%,CHR$(
     32,(B%+2816%)/64%+48%,(B%+2816% AN
     D 56%)/8%+48%,48%+S%);'     ';
4100 IF F%=0% THEN 4900
4200 POKE -1024%,6%,112%,195%,15%,96%
4300 Z%=CALL(-1024%,R%) : IF PEEK(-747%
     ) THEN 5700
4350 GOSUB 15000
4400 Y%=0% : FOR J%=0% TO 254% STEP 2%
4500 IF PEEK(J%-1020%)=255% GOTO 4800
4600 V%=PEEK(J%-1019%)
4700 Y%=Y%+(V% AND 31%)+1% : NEXT J%
4800 IF Ö% Y9$=NUM$(Y%) : Y9$=SPACE$(5%
     -LEN(Y9$))+Y9$ : ; §L%,Y9$;' ';
4900 IF (R% AND 1% AND Ö%) A9$='R D' EL
     SE IF (R% AND 2% AND Ö%) A9$='R  '
     ELSE A9$='RWD'
4910 ; §L%,TAB(K%+12%+(4%*F%))A9$;
5000 IF Ö% THEN K%=K%+W1% : IF K%+W1%>=
     W2% GOSUB 5800 : K%=0%
5100 NEXT B%
5200 NEXT S%
5300 IF K% GOSUB 5800
5400 GOSUB 5800 : ; §L%,RIGHT$(NUM$(G%)
     ,2%)' out of'G8%' sectors free'; :
     GOSUB 5800 : GOSUB 5800
5500 NEXT D% : GOTO 6100
5600 DEFFNO%(X$)=-(INSTR(1%,O$,X$)<>0%
     OR INSTR(1%,O$,CHR$(ASC(X$)+32%))<
     >0%)
5700 ; : ; "Diskerror DRIVE"D% : GOTO 5
     500
5800 ; §L% : IF L% RETURN
5900 F9%=F9%+1% : IF F9%>23% GET Z9$ :
     F9%=0%
6000 IF ASC(Z9$)<>13% RETURN
6100 END
7000 REM GET DATE AND TIME  YY-MM-DD an
     d HH:MM:SS ******************
7010 DIM H$=5%,M$=5%,S1$=5%,D$=10%
7020 D%=0% : IF (PEEK(65008%) AND 4%)=0
     % THEN 7020
7030 FOR I%=0% TO 2% : Z%(I%)=255% XOR
     PEEK(65008+I%) : NEXT I%
7040 Z$=ADD$(MUL$(NUM$(256%*Z%(2%)+Z%(1
     %)),'512',0%),NUM$(Z%(0%)*2%),0%)
7050 IF COMP%(Z$,'8640000')>-1% D%=D%+1
     % : Z$=SUB$(Z$,'8640000',0%) : GOT
     O 7050
7060 Z=INT(VAL(Z$)/100%)
7070 Y%=PEEK(65400) : M1%=PEEK(65401) :
     D%=PEEK(65402)
7080 H%=Z/3600% : Z=Z-3600*H%
7090 M%=Z/60% : S%=Z-60%*M%
7100 H$=RIGHT$(NUM$(100%+H%),3%)
7110 M$=RIGHT$(NUM$(100%+M%),3%)
7120 S1$=RIGHT$(NUM$(100%+S%),3%)
7130 REM READ DATE
7140 Y1%=PEEK(22525%)
7150 M1%=PEEK(22526%)
7160 D%=PEEK(22527%)
7170 D$='19'+RIGHT$(NUM$(Y1%),2%)+'-' :
     IF LEN(NUM$(M1%))<3 THEN D$=D$+'0'
7180 D$=D$+RIGHT$(NUM$(M1%),2%)+'-' : I
     F LEN(NUM$(D%))<3% THEN D$=D$+'0'
7190 D$=D$+RIGHT$(NUM$(D%),2%)
7200 RETURN
8000 REM set uppercase in file n ame (F
     $)
8010 FOR I%=1% TO LEN(F$)
8020 A%=ASC(MID$(F$,I%,1%)) : IF A%>95%
     THEN GOSUB 8100
8030 NEXT I%
8040 RETURN
8100 F$=LEFT$(F$,I%-1%)+CHR$(A%-32%)+RI
     GHT$(F$,I%+1%)
8110 RETURN
15000 REM get the date and print it
15001 A0$=RIGHT$(NUM$(PEEK(64765%)),2%)
     : IF LEN(A0$)=1% THEN A0$='0'+A0$
15002 A1$=RIGHT$(NUM$(PEEK(64766%)),2%)
     : IF LEN(A1$)=1% THEN A1$='0'+A1$
15003 A2$=RIGHT$(NUM$(PEEK(64767%)),2%)
     : IF LEN(A2$)=1% THEN A2$='0'+A2$
15010 A9$='  '+A0$+'-'+A1$+'-'+A2$
15020 ; §L%,A9$;
15030 RETURN
```

Nya kommandon på ABC80

Av Anders Franzen <5258>

Detta är ett assemblerprogram som länkar in några nya kommandon på ABC80. Det är avsett för dem som vill skapa egna kommandon eller helt enkelt tycker att det vore roligt att veta hur man gör. Programmet placeras på höga adresser i minnet så det passar både för 16K och 32K. Dessutom spelar det ingen roll vilken checksumma datorn har, det känner programmet själv av. Programmet går även att köra om man bara har kassettspelare eller om man dessutom har flexskiveenhet. Rutinen startas med hjälp av CALL(61440) från BASIC, dvs anrop av START.

Om man anropar en subrutin i assembler måste datorn veta vart den ska ta vägen när subrutinen är utförd. Då instruktionen CALL exekveras placerar Z80 adressen till instruktionen efter CALL'et på stacken. Då en subrutin är klar görs retur till adressen på stacken.

Det kan liknas vid en irrfärd i en labyrint. Vid varje vägskäl skriver vi upp vårt vägval, antingen vänster eller höger, på en lista. Med tiden växer listan nedåt och vi blir trötta och vill gå hem. Hur vet vi hur vi ska gå? Jo, vi läser bara listan underifrån och med ledning av detta vet vi hur vi ska svänga. Antag att vi lägger oss ner och sover då vi kommit halvvägs hem. Antag vidare att någon kommer och ändrar i listan, byter ut ett höger mot ett vänster! Detta medför naturligtvis att vi går vilse när vi vaknar. Genom att ändra på listan kan man få oss att gå en annan väg.

Ett sätt att länka in nya kommandon i ABC80 är att använda interupt. När datorn ligger i sina loopar i ROM och väntar på kommandon kan vi få processorn att hoppa till en egen rutin då en rad är inmatad genom att låta en ny interuptrutin ändra returadresserna på stacken. Vi ändrar den adress som medför retur tillbaka till komandotolkningen i ROM så att den istället pekar på en egen rutin. Denna rutin kan sedan tolka våra kommandon på ett godtyckligt sätt.

För att förstå det hela måste vi veta hur BASIC-tolken jobbar. Den anropar till att börja med en rutin som läser in en rad från tangentbordet. Denna subrutin anropar i sin tur en snutt som väntar på att en tangent ska tryckas ned. Snutten ligger och loopar ända tills en flagga sätts. Flaggan sätts av en interuptrutin så fort en tangent trycks ned. Det betyder att följande 16-bitars tal kan ligga på stacken då interuptrutinen arbetar:

240	Returadress då "läs en rad" exekverats.
IX	Register som "läs en rad" sparar på stacken.
700	Returadress då "vänta på tangenttryck" är utfört.
okänd	Returadress då interupt är färdigbehandlat.
AF	Register AF som vi sparar i interuptrutinen.
HL	Register HL som vi sparar i interuptrutinen.
DE	Register DE som vi sparar i interuptrutinen.

Till DE pekar SP (stackpekaren)

Naturligtvis kommer andra värden att ligga på stacken om vi t ex just nu kör ett BASIC-program. Allra först i interuptrutinen känner vi därför av om datorn jobbar med BASIC-program eller om den väntar på kommando. Dessutom kontrollerar vi att den första returadressen är 240. Om ett program körs just nu eller adressen är felaktig fortsätter vi i BASIC-tolkens interuptrutin.

När man använder interuptrutiner programmeras ett chips i datorn som kallas PIO. Det måste veta när och hur det ska agera. En vektor är en adress som pekar ut två bytes i minnet. Dessa bytes innehåller adressen till interuptrutinen. Vi skickar den låga byten av vektorn till PIO och den höga byten till CPU'ns I-register. BASIC-tolken har redan programmerat PIO'n med värdet 52. Detta värde utnyttjas även av kassettrutinerna så det låter vi vara oförändrat. Det enda vi behöver göra är alltså att låta I-registret få den höga byten av vektorn.

```
        ORG  61440          ; Placera programmet på lämplig
                            ; plats.
START   LD   HL,START       ; Adress till programmets
                            ; början.
        LD   (65063),HL     ; Sänk värdet på stackpekaren,
                            ; vi reserverar härigenom arean
                            ; som detta program ligger i.
        LD   HL,VEKTOR      ; Adress till nya interuptvek-
                            ; torer.
        LD   A,H            ; Höga byten i adressen
        LD   I,A            ; till CPU'ns I-register.
        LD   HL,TEXT1       ; Adress till text att skriva.
        LD   BC,TEXT2-TEXT1 ; Längd på texten att skriva.
        CALL 11             ; Skriv en liten inlednings-
                            ; text genom att anropa en sub-
                            ; rutin i ROM.
        JP   198            ; Hoppa till ROM och vänta på
                            ; kommando.
;
        ORG  61440+52       ; Observera att låga byten är
                            ; 52.
VEKTOR  DEFW INTERUPT       ; Adress till interuptrutin för
                            ; tangentbordet.
        DEFW (1328)         ; Adress till interuptrutin för
                            ; kassettläsning.
TEXT1   DEFB 13
        DEFB 10
        DEFM "MINIHJÄLPARE   A"
        DEFM "BC-klubben 1985-"
        DEFM "08-21"
        DEFB 13
        DEFB 10
        DEFB 13
TEXT2   EQU  $
;
```

Här kommer nu den nya interuptrutinen som exekveras varje gång en tangent trycks ned på tangentbordet.

```
;
INTERUPT PUSH AF            ; Spara A-registret på stacken.
        LD   A,(IY+14)      ; Är datorn i RUN-mode,
        AND  A              ; dvs exekveras ett BASIC-pro-
                            ; gram just nu?
        JP   Z,799          ; Ja, fortsätt i så fall i ROM.
;
```

Datorn ligger just nu, när interuptet inträffat, i en slinga som läser in en rad från tangentbordet. Där väntar den på interupt, dvs tangenttryck, ända tills dess hela raden är inmatad och RETURN tryckts ned.

```
;
        PUSH HL             ; Spara register.
        PUSH DE
        IN   A,(56)         ; Läs av tangentbordsporten.
        AND  127            ; Släck paritetsbiten (bit 7).
        CP   12             ; CTRL-L?
        JR   Z,INT1         ; Ja, hoppa och töm bildskärmen.
        CP   9              ; Högerpil?
        JR   NZ,INT2        ; Nej, hoppa.
;
```

Här följer en specialfiness. Om datorn väntar på att vi ska knappa in första tecknet i en rad och finner att vi trycker på högerpil, visas den senast inmatade raden igen som nu kan editeras. Mycket praktiskt då man skrivit in en lång rad men som visar sig vara felaktig. ABC80 får sålunda en finess som finns på ABC800!

```
;
        LD   A,B            ; Är detta det första tecknet
        CP   C              ; som datorn får in till raden?
        JR   NZ,INT2        ; Nej, hoppa.
;
```

Om ABC80 ligger och väntar på att vi ska skriva in ett kommando har register BC följande värde då interuptrutinen anropas:

C Max antal tecken kvar att mata in på raden.
B Max antal tecken totalt i rad att mata in (=120).
Om inget tecken är inmatat till raden är C=B. Tack vare detta kan vi enkelt kolla om ett tecken är det första på raden.

```
        LD   HL,(65019)     ; ED-kommandot sparar här adress
                            ; till rad som ska editeras, tag
                            ; denna adress.
        LD   A,(HL)         ; Tag första tecknet i rad att
                            ; editera.
        CP   13             ; Är raden tom, dvs är första
                            ; tecknet RETURN?
        JR   NZ,INT2        ; Nej, editering pågår. Hoppa!
        LD   HL,BUFFERT     ; Adressen till specialbuffer.
        CP   (HL)           ; Är bufferten tom?
        JR   Z,INT2         ; Ja, hoppa.
        LD   (65019),HL     ; Texten i specialbufferten
                            ; kommer att editeras.
        PUSH BC             ; Spara register som används av
                            ; "mata in rad".
        PUSH HL             ; Tag tecken i specialbufferten.
        LD   A,13           ; ASCII-kod för RETURN.
        LD   BC,120         ; Specialbufferten är 120 tecken
                            ; lång.
        CPIR                ; Leta slutet på raden, hittar
                            ; alltid CR.
        LD   (HL),10        ; Lägg ner "Line Feed" sist i
                            ; raden.
        LD   A,121
        SUB  C              ; Beräkna längden på texten i
                            ; specialbufferten.
        LD   C,A
        POP  HL             ; Adress till specialbufferten.
        CALL 11             ; Skriv ut texten att editera
                            ; på skärmen.
        EX   DE,HL          ; HL=adress i bildminnet först
                            ; på nästa rad.
        SET  7,(HL)         ; Tänd markören!!!
        POP  BC             ; Återställ registret som "mata
                            ; in rad" använder.
        POP  DE             ; Återställ DE-registret.
        POP  AF             ; Återställ inte HL, HL pekar
                            ; på aktuell adress i bildminnet
                            ; så att markören släcks då
                            ; nästa tangent tryckts ned.
        POP  AF             ; Återställ A-registret.
        EI                  ; Tillåt interupt igen.
        RETI                ; Retur från interuptet.
;
```

Jaha, här slutar specialfinessen med högerpil. Nu följer en rutin som tömmer bildskärmen om CTRL-L tryckts ned. Detta ger endast om markören står i kolumn noll, dvs vi har ej börjat skriva in någon rad. Det vore ju pinsamt om skärmen rensades då vi hunnit halvvägs i en programrad!

```
;
INT1    LD   A,B            ; Har vi skrivit in några andra
                            ; tecken på
        CP   C              ; raden?
        PUSH IX             ; Spara register.
        CALL Z,630          ; Nej, töm bildskärmen, rutinen
                            ; finns i ROM.
        POP  IX             ; Återställ register.
;
```

Nu ska vi kontrollera vad returadressen på stacken har för värde. Vi vet precis var den ska ligga i förhållande till stackpekaren SP, nämligen 6 heltal upp vilket är detsamma som tolv bytes.

```
;
INT2    LD   HL,12          ; Nu ska vi kolla om den första
                            ; returadressen
        ADD  HL,SP          ; är okey. Låt HL peka på rätt
                            ; ställe i stacken.
        LD   DE,240         ; Den här adressen bör ligga där.
        CALL TEST           ; Kolla om adressen är riktig.
        LD   DE,CMD         ; Ny returadress till vår
                            ; kommandoavtolkning.
        JR   Z,INT4         ; Hoppa om adressen var riktig.
;
```

Om vi använt oss av ED-kommandot men vill skriva in ett nytt kommando utan att editera är returadressen en annan. Kolla om ED använts och byt i så fall ut returadressen mot en som medför hopp till en ny ED-snutt.

```
;
        LD   DE,(2777)      ; Om ED använts bör denna adress
                            ; ha använts.
        LD   A,(DE)         ; Bäst att kolla vad som finns
                            ; i ROM här,
        CP   33             ; denna adress är lite olika
                            ; beroende på
        JR   Z,INT3         ; datorns checksumma. Hoppa om
                            ; checksumma 9913.
                            ; Notera att checksumman 10042
                            ; är identisk med 9913 frånsett
                            ; en byte men den är ointressant
                            ; i detta fall.
        LD   DE,(2775)      ; Denna adress används om check-
                            ; summan är 11273.
INT3    CALL TEST           ; Kontrollera om returadressen
                            ; är från ED.
        JR   NZ,INT5        ; Nej, hoppa. Tolken var någon
                            ; annanstans eller så har vi
                            ; redan ändrat den!
        LD   DE,ED          ; Adress till vår egen ED-rutin.
INT4    LD   (HL),E         ; Lägg ner den nya returadressen
                            ; i stacken.
        INC  HL             ; (Här sker alltså det som hela
                            ; detta program
        LD   (HL),D         ; bygger på, att lura tolken!)
INT5    POP  DE             ; Återställ några register.
        POP  HL
        JP   799            ; Fortsätt i ROM. Där läses vil-
                            ; ken tangent som tryckts ned
                            ; av, och tangentflaggan sätts.
;
TEST    LD   A,(HL)         ; Tag låg byte i adressen på
                            ; stacken.
        CP   E              ; Jämför med vad det bör vara.
        RET  NZ             ; Retur om olika med Z-flaggan
                            ; satt.
        INC  HL
        LD   A,(HL)         ; Tag hög byte i adressen på
                            ; stacken.
        DEC  HL             ; Återställ HL.
        CP   D              ; Kolla om också höga byten är
                            ; lika.
        RET                 ; Om olika är Z-flaggan satt.
;
```

Det var allt som har med interupt att göra. Det som har uträttats just nu är alltså att en returadress på stacken har förändrats. När vi har matat en rad på tangentbordet avslutad med CR kommer retur att ske till kommandoavtolkningsrutinerna nedan som tolkar raden.

Här kommer rutinen som utförs då vi editerat klart en rad. Den återställer stacken som ED använt sig av. Om vi alltså skriver "ED 10" och return får vi upp rad 10 klar att editeras. Nu skriver vi ett kommando istället för att editera raden varvid vi hamnar i denna rutin.

```
;
ED      LD   HL,128         ; ED-kommandot har reserverat
                            ; 128 bytes på
        ADD  HL,SP          ; stacken.
        POP  DE             ; Adress till radbufferten.
        LD   SP,HL          ; Återställ stacken i skick som
                            ; innan ED.
        EX   DE,HL          ; HL pekar på radbufferten.
        JR   CMD2           ; Fortsätt i nya kommandoavtolk-
                            ; ningen.
;
```

Här börjar kommandoavtolkningen. Raden som vi knappat in ligger nu i en radbuffert någonstans i RAM. Innan BASIC-tolken anropar rutinen "läs in rad" sparar den adressen till radbufferten på stacken. Om vi gör POP HL kommer adressen därför att hamna i HL. Om den inmatade raden är tom hoppar vi till ROM för att hämta en ny.

```
;
CMD     CALL 830            ; Återställ CTRL-C-flaggan.
        POP  HL             ; HL pekar på radbufferten.
CMD2    PUSH HL             ; Spara adressen.
        RST  32             ; Skippa eventuella mellanslag,
                            ; A=första tecknet.
        EX   (SP),HL        ; HL pekar på radbuffert, spara
                            ; adress till tkn.
        CP   13             ; Är den inmatade raden tom?
        JP   Z,219          ; Ja, hoppa till ROM och hämta
                            ; en ny.
;
```

Nu sparar vi raden i vår specialbuffert. Det gör vi för att kunna ta fram den igen för editering då vi trycker på högerpil. En tom rad tömmer alltså inte specialbufferten!

```
;
        LD   DE,BUFFERT     ; Adress till vår specialbuffert.
        LD   BC,120         ; Raden är 120 tecken lång.
        LDIR                ; Spara inmatad rad i special-
                            ; buffert.
;
```

Det finns en rutin i ROM som letar efter ord i en tabell. Tabellen ska se ut så här:

```
;
                            ;    128,ORD1,129,ORD2,.. ..255
;
```

Orden åtskiljs med en byte där bit 7 är satt. Tabellens slut markeras med en byte 255. Om ordet återfinns placeras värdet på byten före ordet i A-registret. Om ORD1 hittats får A sålunda värdet 128. Med hjälp av detta värde kan en annan rutin leta fram en adress i en tabell och hoppa till den adressen.

```
;
        POP  HL             ; Adress till bufferten.
        LD   DE,CMDTAB      ; Adress till vår egen kommando-
                            ; tabell.
        CALL 68             ; Leta inmatat kommando i
                            ; tabellen.
        JP   NZ,249         ; Hoppa om kommandot ej finns i
                            ; tabellen,
                            ; i ROM kollas då om det är ett
                            ; annat kommando.
        LD   BC,204         ; Adress att göra retur till
                            ; efter kommadot.
        PUSH BC             ; Lägg returadressen på stacken.
        LD   DE,JPTAB       ; Tabell med adresser till våra
                            ; kommandon.
        JP   314            ; I ROM finns en rutin som
                            ; hoppar till den adress som
                            ; motsvarar det kommando som
                            ; hittats i kommandotabellen.
;
CMDTAB  DEFB 128            ; Här är tabellen med våra nya
                            ; kommandon.
        DEFM "EXIT"         ; Maximalt får 127 kommandon
                            ; plats i tabellen.
        DEFB 129
        DEFM "^"            ; Observera att det är samma
                            ; "kod" för
        DEFB 129            ; gement och versalt "^", de
        DEFB "ü"            ; har ju samma funktion.
        DEFB 130
        DEFM "RAM"
        DEFB 255            ; Slut på tabellen.
;
JPTAB   DEFW 0              ; EXIT medför hopp till adress
                            ; noll, dvs RESET.
        DEFW LIST           ; Adress till listrutinen.
        DEFW RAM            ; Adress till RAM-kommandot.
;
```

LIST-rutinen gör att vi bara behöver skriva "U" (tyskt y) för att lista programmet. Både gement och versalt "U" (tyskt y) ligger i kommandotabellen därför att BASIC-tolken inte kan skilja på dessa. Endast 15 programrader visas på skärmen innan tolken väntar på tangenttryck. Detta medför att de första programraderna aldrig försvinner i bildskärmens övre ände.

```
;
LIST    LD   IX,-16         ; Reservera 16 bytes på stacken,
                            ; det är
        ADD  IX,SP          ; precis vad som sker i ROM-
                            ; LIST-rutinen.
        LD   SP,IX
        LD   (IX+0),15      ; Scrolla ej så att översta
                            ; raderna försvinner.
        LD   A,(2811)       ; Kontrollera datorns checksumma.
        CP   221
        JP   Z,2811         ; Hoppa om checksumma 11273.
        JP   2809           ; Hoppa om checksumma 9913.
;
```

RAM är ett kommando som skriver ut hur stort BASIC-programmet som finns i minnet är, hur stor plats variablerna tar samt hur mycket ledigt utrymme som finns kvar.

```
;
RAM     LD   HL,RAM1        ; Adress till text att skriva.
        LD   BC,RAM2-RAM1   ; Längd på text att skriva.
        CALL 11             ; Skriv text på bildskärmen.
        LD   HL,(65054)     ; Värde på EOFA, adress till
                            ; programmets slut.
        LD   DE,(65052)     ; Värde på BOFA, adress till
                            ; programmet.
        CALL SKRIVDIF       ; Skriv ut programmets storlek.
        LD   HL,RAM2
        LD   BC,RAM3-RAM2
        CALL 11             ; Skriv text på skärmen.
        LD   HL,(65056)     ; Värde på HEAP, här tar vari-
                            ; ablerna slut.
        LD   DE,(65054)     ; Värde på EOFA, adress till
                            ; programmets slut.
        INC  DE             ; Pekar på variabelutrymmet.
        CALL SKRIVDIF       ; Skriv ut variablernas minnes-
                            ; utrymme.
        LD   HL,RAM3
        LD   BC,RAM4-RAM3
        CALL 11             ; Skriv text.
        LD   HL,(65063)     ; Värde på STACK, här slutar
                            ; användbart RAM-minne.
        LD   DE,(65056)     ; Värde på HEAP, här tar vari-
                            ; ablerna slut.
        CALL SKRIVDIF       ; Skriv ut hur mycket ledigt
                            ; minne som finns.
        LD   HL,RAM4
        LD   BC,RAM5-RAM4
        CALL 11             ; Skriv text.
        RET
;
RAM1    DEFB 13,10              ; Här ligger textrader som
        DEFM "MINNE     BYTES"  ; gör att detta visas på
        DEFB 13                 ; skärmen om vi skriver in
                                ; kommandot RAM:
        DEFB 10
        DEFM "==============="  ;
        DEFB 13                 ; MINNE     BYTES
        DEFB 10
        DEFM "PROGRAM   "       ; ===============
RAM2    DEFB 13                 ; PROGRAM   0
        DEFB 10
        DEFM "VARIABLER "       ; VARIABLER 0
RAM3    DEFB 13                 ; LEDIGT    16000
        DEFB 13
        DEFM "LEDIGT    "       ; ===============
RAM4    DEFB 13                 ;
        DEFB 10
        DEFM "==============="
        DEFB 13
        DEFB 10
RAM5    EQU  $
;
```

SKRIVDIF är en subrutin som skriver ut skillnaden mellan talet HL och talet DE på bildskärmen.

```
;
SKRIVDIF AND  A
         SBC  HL,DE
         EX   DE,HL         ; DE är nu talet att skriva.
         LD   HL,-10        ; Reservera tio bytes på stacken.
         ADD  HL,SP
         LD   SP,HL
         EX   DE,HL         ; HL är talet igen.
         LD   A,(9031)      ; Kolla datorns checksumma.
         CP   221
         JP   Z,9031        ; Hoppa om checksumma 11273.
         JP   9029          ; Hoppa om checksumma 9913.
;
```

Här ligger den specialbuffert där vi sparar varje inmatad rad.

```
;
BUFFERT  DEFB 13            ; Plats för vår specialbuffert.
                            ; Ett CR ligger
         DEFS 120           ; först efter initieringen.
;
```

Sammanfattningsvis kan sägas att kommandon bestående av ord tolkas i en speciell tolkningsrutin efter det att vi lurat BASIC-tolken att hoppa dit. Kommandon som nås med ett tryck på en CTRL-tangent bearbetas direkt i interuptrutinen. Det som återstår nu för att få en avancerad hjälpare är att konstruera fler nya kommandon men det överlåter jag åt var och en. Med lite fantasi kan man säkert åstadkomma de mest häpnadsväckande kommandon.

ABC-kassett 16

LIB-lista kassett/diskett § 16

PROGRAM	STL	BESKRIVNING
CASDISK3.BAS	14	Programmet för ABC80 som för över till disk
VISA .BAS	3	Används för att läsa textfiler
SPIDER .BAS	31	GAME&WATCH-inspirerat spel
DJUR .INF	7	Anvisningar till DJUR.BAS läses med VISA
DJUR .BAS	22	Ett intressant program där datorn lär sig
LABYRINT.BAS	10	Datorn ritar olika labyrinter 40 och 80 tkn
FASTCAS .INF	21	Anvisningar programmet som ökar hastigheten vid läsning/skrivning på banspelaren.(VISA)
FASTCAS .16K	14	För 16 Kb-minne
FASTCAS .32K	15	För 32 Kb-minne
VÄXLAPGM.UTL	12	Maskinkodsrutin som växlar mellan två basicprogram i minnet samtitigt.
SORTERA .BAS	27	Kappkörning mellan 6 olika sorteringsalgoritmer.
DIMFIX .SUB	9	Fixar med DIM
SWAPER .UTL	14	Programmet sparar utrymme på disketten genom att ta bort blanksteg utom inom citattecken.
REMERASE.BAS	10	Avlägsnar REM-rader i ett program på flexskiva.
ASTART .BAS	13	Ordnar autostart på ett program på flexskiva. Fungerar på senare versioner av DOS.
***** ABC800 *****		
CONFIG .BAS	15	System-program, se vidstående text.
HARDCOPY.REL	8	Skriver ut texten på bildskärmen på en skrivare. Hur den används beskrivs här bredvid.
		Det kommersiella SCHACK-paketet för ABC800-serien som är en gåva från TDX.
SCHACK .BAC	8	Starta med RUN SCHACK
SCHACK0 .ABS	4	
SCHACK6 .ABS	4	
SCHACK0 .BAC	32	
SCHACK2 .BAC	32	((Programmet känner själv vilken
SCHACK6 .BAC	32	dator Du har och laddar sedan
SCHACK40.SHP	16	rätt fil.
SCHACK6 .SHP	28	OBS. Om Du ej har grafik i datorn
SCHACK80.SHP	16	kan Du använda ett vanligt schack-
CHAR800 .Z80	4	bräde vid sidan om men köra pro-
FIGUR0 .Z80	4	grammet på datorn.))
FIGUR6 .Z80	4	
GRAPTEXT.Z80	4	
RENUM .Z80	4	
SCHACK0 .Z80	24	
SCHACK6 .Z80	24	
		REMOTE-rutiner. Se artikel bredvid.
INIUSERS.BAS	8	Skapar en tom fil för CRERUSER.BAS
REMOTE .BAS	4	Start-rutinen
REMOTE .DOC	16	Anvisningar till REMOTE och dess filer.
REMOTASM.BAS	20	Innehåller maskinkoden.
CREPASS .BAS	8	Skapar lösenord för systemet m.m.
CRERUSER.BAS	20	Skapar användares lösen och loginnamn.
RLOGIN .BAS	12	Inloggnings-rutinen.
RTERM .BAS	8	Ett terminal-program.
PASSWORD.SYS	4	Filen som skapas av CREPASS.BAS
USERS .SYS	4	Filen som skapas av INIUSERS.BAS

Vi har valt att publicera innehållet och kommentarer till kassett 16 som service åt dem som ej har egen skrivare.
Redaktionerna.

ABC kassett nr 16

Producerad 1985-09-02

FEL PÅ KASSETTEN ? Observera att samma innehåll finns på kassettens båda sidor som vanligt. Om du får error 11/16 betyder det att filen är en textfil och går således inte att köra med RUN! Kör programmet VISA.BAS svara med filnamnet och stega fram texten med mellanslag.

DJUR

Ett intressant program som kan ses som ett embryo till artificiell intelligens. Genom dialog lär sig datorn mer och mer och kan sedan utnyttja denna kunskap. Programmet behöver inte handla om djur. Genom att ändra lite text i programmet och sedan "lära" datorn rätt saker, kan det t.ex. användas till att sammanställa en perfekt meny till middag. Finessen är att strukturera frågor och svar på rätt sätt innan man kör igång. Börja med förrätt, varmrätt, dessert. Utveckla varje gren vidare. Varmrätt ger kött, fisk osv. Fortsätt med lämpliga frågor tills en speciell rätt valts ut. Vartefter man kommer på nya goda rätter kan man vid senare tillfälle komplettera datafilen.
Se vidare DJUR.INF använd då VISA.BAS eftersom det är en textfil.

CONFIG-800

Program för att ändra optioner som laddas in med hjälp av filen ADDOPT.ABS. Programmet är självinstruerande. Det är samma program som fanns på kassett nr 13, men denna version har även HARDCOPY.REL med samt förbättrade felutskrifter.

HARDCOPY-800

Hardcopyrutinen fungerar på ABC800 C, ABC800 M, ABC802, ABC802 och DTC 1. Hardcopy är engelska för utskrift av texten på bildskärmen på en skrivare. Programmet/rutinen har skänkts till klubben av DIAB.

För att ladda rutinen så måste man ändra systemfilen ADDOPT.ABS. Detta kan enklast göras med programmet CONFIG som finns på kassetten/disketten. Preparera sedan autostart på ADDOPT.ABS med PREABS. När detta är gjort så skall systemet resettas, varvid hardcopyrutinen läses in. Detta är alltså samma förfarande som krävs för att ladda in andra REL-filer, t ex ISAM eller TERMOPT för ABC806. Saknar du filen ADDOPT.ABS så be att få den av din återförsäljare.
För att få en utskrift av bildskärmen på skrivaren, tryck på SHIFT + CTRL + Pf 8. Program som använder denna Pf-tangent kan då inte användas, eftersom hardcopyrutinen "går före". Krävs det andra än defaultparameter för printer, initiera då skrivaren med rätt parameter genom att öppna en printerfil och stäng sedan den direkt.
Hardcopyrutinen fungerar även på ABC806 i WIDTH 40! Däremot tas ingen hänsyn till attribute, dvs text med dubbel bredd och eller dubbel höjd skrivs ej ut korrekt på printern.

SCHACK-800

SCHACK kan köras på alla maskiner ur ABC800-serien och FACIT DTC/DTC 2. Kör man programmet på en maskin som saknar HR-grafik så får man dock inte spelplanen på skärmen. Startprogrammet är listbart. Startas med "RUN SCHACK". Har man ABC806 så frågar programmet om man har färg eller monokrom skärm.

REMOTE-800

REMOTE800 är ett programpaket som kan användas för att köra en ABC800-maskin från en terminal på samma sätt som om man satt vid tangentbordet för datorn.
Vanliga BASIC-program går att köra från terminal, men program som skriver i bildminnet och assemblerprogram som inte skriver ut text via BASIC:en kan inte köras på detta sätt.
För att få cursoradresseringen och blankning av skärmen att fungera måste man köra det speciella terminalprogrammet RTERM som medföljer. Programpaketet är avsett för ABC800/802/806.
Källkoden till assemblerrutinen i REMOT-ASM finns tillgänglig i klubbens programbank, biblioteket ABC800/ASMKOD.

INSTRUKTIONER TILL DJUR

Programmet går ut på att man ska tänka på ett djur, som sedan datorn ska gissa.

För varje felaktigt svar frågar programmet efter svaret och hur den kan skilja mellan det nya djuret och djuret den redan känner till. Datorn kommer följaktligen öka sina kunskaper alltefter som den lär sig flera djur. Man får svara med obestämd artikel framför djuret i de fall där det behövs.
När man är klar svarar man 'N' på frågan 'TÄNKER DU PÅ ETT DJUR?, och 'EXIT' för att avsluta. Då sparas alla inslag- na djur på skiva i filen 'DJURBIBL.DAT'.

Lycka till.
<863>
Per Lundin
08-36 16 08

Programredaktionens kommentarer :

Om du inte har floppy
1) Ändra rad 1160 PREPARE P1$ ASFILE 2%
Ny rad
1155 ;:;"Kassett för Recording!";:GET A$
Ny rad
1235 CLOSE 2% :END

2) Spara den nya versionen på ett band som har plats för datafilen efter.
3) Första gången man nu gör RUN måste bandspelaren vara avstängd !
4) När du avslutar en körning och kommer till "Kassett för recording!" sätt på bandspelaren för inspelning och tryck på valfri tangent.(För smidig start vid nästa körning bör bandet stå direkt efter DJUR.BAS vid inspelnigen)

Anvisningar till REMOTE-rutiner för ABC800-serien

Mikael Lidn har gjort några remoterutiner för ABC800-serien för att pröva remotefunktionen innan den integeras i ett annat program. Då kommer Mikael trolgen att förbättra dem. Nu har Mikael skänkt programmen till klubben i nuvarande skick. De är listbara och går att ändra i.
En remoterutin är till för att man skall kunna köra en dator från en terminal som om man satt vid datorns tangentbord. Då fungerar dock inte program som adresserar direkt i bildminnet. I denna versionen tas inte heller hand om när ett program blankar skärmen så att styrtecken enligt ADM3A för att blanka skrämen sänds. Cursoradressering enligt ADM3A är saknas troligen också. Bell (ASCII 7) ger ej signal på terminalen. Använder du RTERM som terminalprogram på någon ABC800 så fungerar både blankning av skärmen och cursoradressering.
Prova gärna genom att koppla upp två 800:or mot varandra med kabel mellan V24:orna (GND-GND, TxC-RxC,RxC-TxC,bygla DCD & CTS). Programmet saknar timeout vid tid vid DCD bortfall eftersom det är konstruerat för TGC modem som har 1 s time out vid DCD bortfall.
När man kör remote så kommer allt som man skriver på terminalen upp på värddatorns skärm. Man kan också köra samtidigt på båda tangentborden, dvs på värddatorn och terminalen. Allt man skriver på någon av tangentborden kommer upp på båda skärmarna. Givitvis behöver man inte använda en ABC800-maskin som terminal.

Mikael Lindn vill gärna ha synpunkter på programmet.

INIUSERS.BAS

Är ett program som jag har gjort. Det skapar en tom fil som måste finnas när du kör programmet CRERUSER.BAS.

CREPASS.BAS

Används för att skapa ett extra lösenord som systemet av säkerhetsskäl begär innan själva inloggningen startas. Man anger också hastighet och ett ev program som skall köras efter själva inloggningen.
CREPASS skapar alltid filen PASSWORD.SYS, så man behöver inte skapa den separat.

CRERUSER.BAS

Används för att skapa loginnamn och lösenord för varje användare. Det är detta programmet som kräver att filen USERS.SYS finns. Skapa alltså den först med INIUSERS.BAS.

REMOTE.BAS

Är statrutinen för remoterutinen, som sedan anropar REMOTEASM.BAS.

REMOTASM.BAS

Innehåller själva maskinspråksrutinerna, som laddas med POKE-satser. Programmet ligger och väntar på bärvåg. Följande sker:
Indikeras bärvåg skickas FF (form feed, ASCII 12) och "*>" till den uppringande terminalen och inloggningsförfarandetet påbörjas. Innan systemet skickar "*>" behövs nog en lite fördröjning, så jag har lagt in en sådan.

Systemet förväntar sig nu det lösenord som har angivits med hjälp av programmet CREPASS. Programmet börjar lösenordskontrollen från och med det att 1:a tecknet i lösenordet har blivit rätt, sedan får ej resterande tecken skrivas fel för då loggas man ut. När resterande tecken i lösenordet är rätt inskrivna körs programmet RLOGIN för själva inloggningen. Det första inloggningsskedet är till för att öka säkerheten i systemet.

RLOGIN.BAS
Här sker normal inloggning med tre försök tillåtna. Login och lösenord läggas upp med programmet CRERUSER (Create Remote User). Programmet startar sedan det program som har angivits med CREPASS.

RTERM.BAS
RTERM är ett terminalprogram. Använder man det så fungerar nog blankningen av skärmen och cursorstyrningen.

<1789>
Bo Kullmar

MANUAL TILL FASTCAS
INLEDNING
Med detta program kan man ställa om överföringshastigheten i 6 steg från 700 till 2400 baud. På grund av att programmet FASTCAS endast är mjukvarubaserat kan överförings-fel ibland uppstå vid de högsta hastigheterna. Man har därför möjlig het att välja mellan flera över- föringshastigheter. När man använder sig av högre hastigheter ställs större krav på vad man använder sig av för kassetter. Kassetter av typ SONY CHF är en kassett som ger god säkerhet, men kassetter av typ chromdioxid har jag funnit olämpliga till ABC80. Fastcas funge- rar på alla checksummor som för närvarande finns till ABC80 (d.v.s till checksummorna 11273, 10042 & 9913). Det upptar något mer än 1 Kbyte av användarminnet och är anpassat till både 16K och 32K ABC80. OBS att Smartaid eller dylikt ej kan användas tillsammans med fast- cas utan ändringar i FASTCAS.

Fastcas finns i två versioner, en för 16K- och en för 32K-ABC80.

START AV FASTCAS
När man startar upp FASTCAS frågar datorn först efter om kassetten skall vara prioriterad. Den enhet (ex CAS: DR0:) som prioriterad, behöver man ej ange något enhetsnamn till. T.ex. LOAD START-datorn laddar då från den enhet som är prioriterad.

Sedan frågas efter om man vill ha bandstopp mellan varje block. Det kan vara värdefullt vid t ex laddning eller lagring av data m.m. på kassett då ej allting laddas/lagras med en gång. Bandet stannar då mellan varje block.

Därefter frågar datorn efter hur långt mellanrum i sekunder man vill ha mellan varje block. Normalt används 0.02 sekunder, men om man vill ha plats för BANDSTOPP ENLIGT OVAN får man välja en längre paustid, t ex 1.0 - 1.5 sekunder.

Välj nu en av av de sex hastigheterna genom att trycka ned önskad siffra (1-6).

1. 700 Baud (Standard ABC80 hastighet.)
2. 1200 Baud
3. 1400 Baud
4. 1800 Baud (Rekommenderad hastighet.)
5. 2100 Baud
6. 2400 Baud

Sedan är FASTCAS initierad !
Det rekommenderas att man på kassetten noterar använd hastighet.

ÖVERFÖRINGSFEL
Programmet FASTCAS är ej helt säkert vid överföring på de högsta hastigheterna. Därför är rekommenderad hastighet 1800 baud. För att kassett-bandspelaren skall bli säkrare vid höga hastigheter krävs hårdvarumässiga ändringar i den. Får du mycket fel kan du prova att trimma på tidskonstanterna för läsning (se nedan). Om ett program spelas in på en bandspelare och av på en annan,är risken för fel större.

BYTE AV ÖVERFÖRINGSHASTIGHET
Om du någon gång vill ändra överföringshastigheten,utan att förlora det program du har i minnet, så skriv så här: ";CALL (49962)", eller om du har 32k-minne, så skriv så här: ";CALL(33578)". Du får då upp en meny där du kan välja överföringshastighet. Tryck siffra (1-6) för önskad hastighet.

BYTE AV ÖVERFÖRINGSHASTIGHET UTAN MENY
Om du vill byta överföringshastighet utan att få upp en meny på skärmen skriver du: ";CALL(50176,n)", eller om du har 32K-minne skriver du: ";CALL(33792,n)", där n är en siffra 1-6 för de olika överföringshastigheterna.

BANDSTOPP AV/PÅ
Om du någon gång vill sätta på bandstopp under pågående programme- ring/ körning av datorn, så skriv så här: ";CALL (50185)", och om du har 32K-minne: ";CALL (33801)". Vill du stänga av bandstopp, så skriv så här: ";CALL(50203)" och om du har 32K-minne: ";CALL(33819)".

TIDSKONSTANTER
Om du tar bort ordet REM i början av rad 57 innan du kör FASTCAS ut- ökas programmet med en rutin som kan användas i stället för att välja en av sex överföringshastigheter. Genom att besvara datorns frågor ändrar man tidskonstanterna för läsning och skrivning.På så vis skapar man egna överföringshastigheter.

Om du vill justera överföringshastigheten själv, eller trimma in den bättre, så finns tidskonstanterna på dessa adresser: Pauser vid skrivning: 49435 och 49444. (32K-minne= 33051 och 33060). Pauser vid läsning : 49471 och 49475. (32K-minn3= 33087 och 33091). OBS! Vid 700 baud's inställning existerar bara den första tidskons- tanten vid läsning resp skrivning.

MELLANRUM MELLAN BLOCKEN
Vill du under körning ändra storleken på mellanrummen mellan blocken så kan du göra det med detta kommando: "POKE 49321,tid i sek * .02", eller om du har 32K-minne så skriv: "POKE 32937,tid i sek * .02". Vill du t.ex. ha 1.0 sek uppehåll skriver du med 16K-minne så här: "POKE 49321,50".

Har du några problem eller synpunkter på FASTCAS, eller om du är mer djupgående intresserad av FASTCAS så ta kontakt med mig.

Hälsningar:

Per Magnusson <2340>

MSG

Hämta filer med TERM100

Text 1809 - Johan Struwe <3338>

Hjälp !
Jag sitter och kör TERM100 och har ett 300 bauds akustiskt modem. När jag vill hämta filer från monitorn med GET <namn> så får jag den föga upplysande skriften SYNK. Sedan händer absolut ingenting. Som grädde på moset kastar Monitorn ut mig. Jag har provat, med samma resultat, ett flertal gånger.
Vad ska jag göra ?

Text 1811 - Arne Hartelius <2731>
Kommentar till text 1809

GET används för att hämta program med FILTRANS och ABCTRANS osv.
Använder man TERM100 gör man så här:
Skriv TYPE FILNAMN.EXT men vänte med att trycka return. Tryck CTRL-É (ASCII 64), sedan Y och sen 1. Nu har du fått upp REC: i övre högra hörnet. Skriv sedan FILNAMN.EXT och return. Nu står det WAIT medan en fil öppnas på din diskett. När WAIT försvunnit trycker du CTRL-É (ASCII 64) och sedan return.
Nu spelas allt som kommer upp på skärmen in på den fil du öppnat. När filen är slut kommer >-promten och du stänger då filen med CTRL-É (ASCII 64), sedan Y och sist 4.
När du laddar in programet där hemma kommer >-promten att slinka med och försöka bli godkänd som BASIC-sats, vilket i sin tur ger ett felmeddelande. Spara då filen under samma namn så slipper du felmeddelandet nästa gång.

MSG

Miniterminalrutin

Text 368 - Bo Kullmar <1789>

Min disk är inne på service fn, vilket innebär att jag inte kan läsa in terminalrutinen till min ABC806:a. Skam den som ger sej ! Jag kör med följande "terminalrutin" nu:

```
10  OPEN 'V24:VSA70C72.40E' AS FILE
1
20  IF PEEK2(PEEK2(65500)+6)<>0 GET
§1,Tkn$ : ;Tkn$;
30  IF SYS(5)=128 GET Tkn$ : ;§1,Tkn$;

40  GOTO 20
```

Visserligen visas inte cursorn, men det sparar både kodningsarbete och tid !
Det fungerar och jag knappa in det varje gång jag slår på maskinen ! I och för sig finns inte VÄXLA för permobas, vilket är ett problem men man kan ju inte få allt. Man kan väl ställa om manuellt då. Kanske lägger jag in en test på det också, när jag tror att jag kommer in där. Observera att det fungerar lite annorlunda när man inte kör med "T".

MSG

Blockläsning

Text 353 - Harald Nautsch <4479>

Finns det något anrop till DOSet så man kan läsa ett block på skivan, vilket som helst?

Text 354 - Matts Kallioniemi <2332>
Kommentar till text 353

CALL(24675,block*32) skriver och CALL(24678,block*32) läser ett block. Block= Track*8+sector. DosBuf 0 används.

Text 357 - Bengt Österholm <2077>
Kommentar till text 354

Tilläggas bör att innan man gör call så gör man POKE -767,X där X står för vilken drive man vill läsa/skriva på.

Text 381 - Benny Löfgren <2615>
Kommentar till text 357

Ett tillägg till:
Efter läsning/skrivning kan man läsa av statusen, dvs hur det gick i byte -747. Då betyder:
0 att det gick bra,
2 att man försökte läsa/skriva utanför skivan (end of media),
8 att det är CRC-fel på sektorn,
16 att det är AM-fel (Adress Mark) på sektorn,
64 att skivan är skrivskyddad och
128 att den är offline, dvs man har inte stängt luckan eller det sitter ingen skiva i driven eller man har ingen drive ansluten.

Text 382 - Peter Öhlen <4659>
Kommentar till text 381

CRC ?

Text 383 - Benny Löfgren <2615>
Kommentar till text 382

CRC är en metod att checksummera data. Det står för Cyclic Redundancy Check. Med andra ord checksummeringsfel.

Sektorinformationsblocken

Text 450 - Thomas Hall <5232>

När man formaterar en skiva "försvinner" ett antal sektorer. Jag har kommit fram till att dessa går åt till 'sektorinformationsblock'.
Det går ju uppenbarligen inte att läsa dessa med 'vanlig sektorläsning' eftersom det inte rör sig om sektorer i vanlig bemärkning.
Enligt beräkningar skulle ett sektorinfoblock innehålla ungefär 13 byte. I dessa borde finnas information om spår, sektor och ID.

Text 451 - Bo Kullmar <1789>
Kommentar till text 450

På en 830 är det frågan om 6 byte enligt följande:

1: CYL
2: SID är till för vilken sida, men används ej utan det är bara någon som lägger ett ID här för skydd.
3: SEC
4: 01H
5-6: EDC
De är de uppgifter man jag har. Skall man läsa eller skriva detta enkla skydd i byte 2 så får man skicka kod till kontrollern.

Text 452 - Benny Löfgren <2615>
Kommentar till text 451

Side-byten används visst, men bara bit 0! De övriga sju bitarna ignoreras (maskas) vid sidcheck.

Informationen mellan sektorerna

Text 468 - Peter Öhlen <4659>

Vad är det egentligen som står mellan sektorerna, och hur kan man komma åt den informationen ?

Text 469 - Bo Kullmar <1789>
Kommentar till text 468

Det är information som controllern behöver för att så att säga hitta rätt på disken. Man måste skicka kod till controllern för att läsa dem. Hur man gör är det lite hysh hysh om, eftersom Pdata utnyttar eller utnyttjade några bitar där som inte används av DOS:et för kopieringskydd.

Text 472 - Peter Öhlen <4659>
Kommentar till text 469

Men om det är så att Pdata utnyttjade (och alltså inte utnyttjar det längre) varför ska det behöva vara en massa hysh hysh om det hela ? I ett inlägg i mötet ABC800 finns ju ett inlägg som beskriver kommandona till diskkontrollern, men det står inte till vilken port man ska skicka dem.
Ps. Jag *AVSKYR* hysh hysh

Text 474 - Bo Kullmar <1789>
Kommentar till text 472

Jag bara gissar att de har bytt, men de kanske har kvar samma. Den version av ORD800 (2.2) är skyddad så. Jag bara tycker att det borde ha gått över till ett bättre skydd, men jag vet inte.

Text 475 - Bo Kullmar <1789>
Kommentar till text 473

Jag tycker att jag är väldigt öppen ändå. Här är lite info.
1 : CYL
2 : SID
3 : SEC
4 : 01H
5-6 : EDC
Det är i SID-byten man lägger ett ID för skydd. Det är bara bit 1 som används där. Vad den andra informationen betyder får du lista ut själv.

Text 483 - Peter Öhlen <4659>
Kommentar till text 475

Jaha, nu vet man det. Men vad har man för nytta att veta vad som finns bakom dörren när man inte kommer in ?
Också dessa förkortningar, man kan ju gissa, men tänk om man gissar fel...

Text 484 - Bo Kullmar <1789>
Kommentar till text 483

Jag har lämat den informationen som har fått. Sedan har jag gissat vad det betyder. Mina gissningar är väl inte bättre än dina !

Text 487 - Peter Öhlen <4659>
Kommentar till text 484

Men den lyckliga person du fått informationen ifrån borde väl veta ? Och sen så detta hur man läser den där informationen. Jag vet att det finns dom som kör här som vet, men varför är de så tystlåtna ? Orkar de inte förklara ? Beror det på ren egoism att de inte vill dela med sig av sina hemligheter ?

MSG

Felaktigheter i ASM

Text 295 - Lars Frej <2260>
Kommentar till text 292

Nu har jag kikat igenom programet i helgen, och det här är vad jag kom fram till:

1. Ta bort följande rader:

```
30 GOTO 100
80 ; S1$;S1%(0%)
3390 Z%=CALL(63488)
3470 DIM B$=80%,L$=S9%+1%,S$=S9%+1%,L1$=80%,F1$=20%,B1%(20%)
```

2. Ändra rad 3310 från:

```
3310 S8%=70% : S9%=6% : P9%=5% : DIM B$=80%,L$=S9%+1%,S$=S9%+1% ,L1$=80%,P5%(P9%,1%)
```

till följande 2 rader:

```
3310 S8%=70% : S9%=6% :REM max antal lablar-max labellängd
3315 DIM L$=S9%+1%,S$=S9%+1%,S1$=S8%*S9%,S1%(S8%)
```

3. Komplettera slutligen rad 3460 till:

```
3460 DIM B1%(20%),E1%(10%,1%),B$=20%,L1$=80%,F1$=20%
```

Var inte orolig för att P9% och P5%(,) försvunnit, de används inte i programmet.

S8% (max antal lablar) och S9% (max labellängd) kan ändras efter behov. Jag har kört med 250 lablar med 7 tkn och det har gått bra!

Lablarna och deras lägen finns (redan efter första passet) i S1$ resp S1%(); antalet lablar finns i S1%. Detta kan utnyttjas för att få en label-tabell-utskrift!

När vi ändå är igång, varför inte ta och ändra så att "EI" också funkar. Det gör det ju inte i grundutförandet.

Börja med av 'städa av' inläsningen. Ändra rad 3280 från:

```
3280 INPUT §1%,T1%,T1%,T2%,C5$,C6$,C3$ : T2%=T2%+1% : DIM T1$=4%*(T1%+1%),T1%(T1%),T2%(T2%)
```

till:

```
3280 INPUT §1%,T1%,T2%,C5$,C6$,C3$ : DIM T1$=4%*(T1%+1%),T1%(T1%),T2%(T2%)
```

2. Därefter måste ASMCON.DAT ändras på två ställen:

a. i början ser det ut så här:

```
95
95,61
ABCDEFG.....(o s v)
...
```

ändra till:

```
96
62
ABCDEFG..... (oförändrat)
```

b. senare i filen ser det ut så här:

```
.
DAA,10239
DI,62463
EXX,55807
HALT,30463
.
```

ändra till:

```
.
DAA,10239
DI,62463
EI,64511
EXX,55807
HALT,30463
.
```

Text 300 - Lars Frej <2260>

När jag kikade igenom ASM och dess data-fil ASMCON reagerade jag för att op-koderna för OTIR (EDB3) och OTDR (EDBB) föll ur mönstret i jämförelse med INIR och INDR.

Igår fick jag tillgång till ABC Örebro:s tidning DATABITEN §2 84. Där nämner David Andersson samma iaktagelse - ASM ger felaktiga opkoder för OTIR och OTDR!!! Han ger också exempel på felaktigheter i viss litteratur (jag tror det var Zaks bok om Z80) och ytterligare någon programvara med samma eller liknande felaktigheter !

Det är alltså dags för en till ändring i ASMCON:

Så här är det det vara	Så här skall
.	.
NOP,255	NOP,255
OTDR,60808	OTDR,60859
OTIR,60803	OTIR,60851
OUTD,60843	OUTD,60843
.	.

Text 318 - Lars Frej <2260>

Jag jobbar inte ofta i assembler, och när jag gör det brukar jag inte använda ASM. Nu har det aktualiserats, och jag tycker att jag ramlar på fel efter fel.

ASM tar inte binära tal som operand/argument, om talet har för många siffror.

På rad 2800 samlas talet, tecken för tecken i S$. S$ används normalt för ev label i akt rad, och den är dimensionerad för detta (DIM S$= S9%+1%).

För att det hela skall gå bra måste man se till att S$ dimensioneras till labellängd + 1, DOCK MINST 9 TKN.

MSG

Cachminne

Text 1195 - Benny Löfgren <2615>
Kommentar till tidigare text

Någon central ram-floppy i Lux-net lär inte behövas heller, en cachad winchester är ju nästan lika snabb, och den är ju både större och datasäkrare !

Text 1196 - Bo Kullmar <1789>
Kommentar till text 1195

Vad menar du med större och datasäkrare ? Den blir väl så stor som man gör den, dvs hur många kort som man har. Datasäker vad menar du med det, bugsäker ?

Text 1197 - Benny Löfgren <2615>
Kommentar till text 1196

En ramfloppy är datasäker så länge det inte blir strömavbrott, det var så jag menade. Vad beträffar storleken så blir den givetvis lika med winchesterns kapacitet. Observera att jag inte talade om ramfloppy utan om ramcache (eller egentligen diskcache), vilket är en helt annan sak. En central ramfloppy är egentligen rätt löjligt, man tappar ju helt finessen med en dylik skapelse, dvs att det skall gå *snabbt* att accessa, det skall näst intill vara en utbyggnad av primärminnet.

Text 1198 - Johan Olofsson <5809>
Kommentar till text 1197

Vill någon vara snäll och undervisa oss mindre hårdvarukunniga om vad cashe är för något?

Om samma någon samtidigt kompletterar med en jämförelse av tider för läsning respektive spårbyte på floppy och winchester, överföringshastighet inom nät samt mellan disk-controlers och RAM vore jag tacksam. Jag har lite svårt att förstå skepticismen mot diskemulatorer.

Jag skulle föreställa mig att man med sådana kommer undan olägenheten av långa diskaccess-tider för de filer, som i administrativa program med förkärlek är flera till antalet, indexerade, spridda över hela disken och accessas stup i ett.

Det som irriterar är tiden för förflyttning av läs/skriv-huvudet !

Text 1199 - Bo Kullmar <1789>
Kommentar till text 1198

är en slags buffert mot disken. Redan läsa sektorer lagras där och hämts ifrån denna buffert om det finns där i stället från disken.

Vad gäller jämförelser i övrigt så har jag lite svårt att ge en sådan. Vad gäller winchesterdiskar så är dessa olika snabba. Det har också betydlese hur man når den, t ev via nät. Näten är det också skillnad på. ABC-NET är mycket långsamt, CAT-NET snabbt och LUX-NET sanabbast. CAT-NET och LUX-NET fungerar dock lite olika så det är svårt att jämföra dem.

Text 1200 - Benny Löfgren <2615>
Kommentar till text 1198

En cache (observera stavningen!) är en snabb "buffert" mellan t ex en disk och en dator (det kan lika gärna vara en s k instruktionscache internt i en CPU). Om man ska läsa en sektor från en disk som är försedd med cache kontrollerar man först om sektorn redan finns i cacheminnet, i så fall tas den därifrån (=snabbt). Annars läser man in sektorn från disken och lägger den först i cachen. Detta innebär att de sektorer som oftast accessas (bibliotek, indexfiler o dyl) oftast ligger i cachen, medan de mer sällan använda sektorerna så småningom ramlar ur. Det finns givetvis mer eller mindre smarta algoritmer för att bestämma hur länge en given sektor skall vara kvar i cachen, men det är grund-

principen. Vad du kallar "diskemulatorer", dvs i vardagligt tal RAM-floppy, är en utmärkt företeelse för att MYCKET snabbt byta program och läsa in data, eller att lagra temporärdata på. Däremot är användningsområdet för en central RAM-floppy rätt begränsat, då man i det fallet endast vinner tiden det tar för disken att flytta huvudet och läsa sektorn, och det ändamålet vinner man med en cache. Kör du ramfloppyn lokalt går det nästan lika fort som att skriva direkt i ramminnet. Med en cache får man också den fördelen att man inte förlorar data om man t ex får strömavbrott, då man hela tiden skriver BÅDE i cachen och på disken. Några tider för överföring är svårt att ge, då det är väldigt beroende av använt media och vald drivetyp.

MSG

TKN80 - TKN40 - TKN80

Text 89 - Börje Gustavsson <3374>

Supersmartaid innehåller ju inget kommando eller ctrl-tecken för att kunna skifta mellan 40 och 80 tecken (med TKN80).

Här är ett förslag hur man kan lägga till det:

KEY T="Z=CALL(21286)_O_L_M

POKE 21286,58,78,2,254,80,40,4,219,24,2,219,3,195,118,2

Lägg också upp en JOB-fil som kan utföra ovanstående om CMOSminne skulle dyka av någon anledning. Adressen där rutinen läggs in kan naturligtvis ändras eftersom rutinen bara har en fast hoppadress. Det är skärmtömningen i basicen (alla checksummor) som den hoppar till i avslutningen av rutinen.

MSG

SMARTIAID 800 jämförd med EXTEND800

Text 139 - Bo Kullmar <1789>

Jag har nu fått tillgång till en manula av ver 1.4 av Smartaid800. Extend800 har vi här på CAT-NET systemet och jag har det även på skiva. Jag kan därför göra lite jämförelser. Sammafattningsvis så tycker jag att XTEND800 är klart bättre. Smartaid800 verkar vara en besvikelse!

Nedan kallar jag Smartaid800 för S och XTEND800 för X. Owoco har gjort S och CAT Ing byrå har gjort X.

S finns på ett speciellt kort och tar 2-300 mA ström. Har man ett gammalt kontrollerkort så räcker inte strömmen utan man måste ha en expansionslåda med IO plats. Databoard buss kan ej användas (Tranfor?) Programvaran tar inte i anspråk något av datorns RAM-minne.

X finns på skiva för stand alone system och CAT-NET. Har man CAT-NET så räcker det med en X som alla kan köra. Systemet finns på REL-filer och laddas i RAM, vilket innebär att lite RAM-minne tas i anspråk (2 K för själva XTEND).

S har en editor, förmodligen fungerar den som motsvarande för ABC80 smartaidarna. X saknar editor.

FIND har både fast CAT kallar sin för CROSS. VAR finns i båda, med X har även VAR FN som ger en lista över funktioner. CROSS på ett namn på en funktion listar alla anrop till funktionen och funktionen i dess helhet.

TYPE, utskrift av textfil saknas i S! Lika så finns inte CHANGE dvs byte av variabelnamn i S, men möjligen kan man använda EXC för detta. EXC byter valfri text mot något. Som ni förstår så har X TYPE och CHANGE.

LIB finns i båda, men CAT kallar sin lib för DIR och den listar default filerna med små bokstäver. Den kan man dock ändra med switch. KEY finns i båda.

HELP, ?, SYS, PEEK och baklängeslistning och listning med ü finns bara i S. Däremot finns BIN$, DEC, DUMP$, DUMPH$ i X för omvanling av tal och dumpning av strängar.

Tangentbordsbuffert finns i en tilläggsrutin till X, men verkar saknas i S! Jobfiler finns i X, men ej i S. Dock har jag inte fått detta att fungera på min XTEND800 till min egen 806:a.

Till X finns dessutom ett program OPTION med vilket man kan ladda in REL filer med utan att använda ADDOPT.

Jag har haft mycket nytta av XTEND800 vid arbete på klubbens CAT-NET system. Främst värdesätter jag CROSSE <funktionsnamn>, CHANGE och TYPE. Dessutom är DUMP$ praktiskt vid debuggning eftersom man då får ut ASCII värden för strängen.

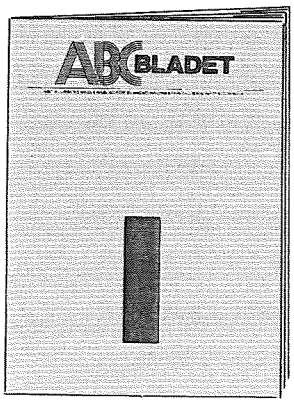
S har jag inte haft tillfälle att prova alls. Eftersom den kräver speciell hårdvara med strömförsöring så blir den ju dyrare än CAT:s XTEND, fast X ju har RAM-minne i anspråk.

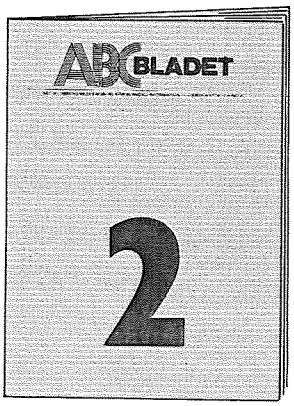
Min slutsats är att XTEND är klart mera användarbart än SMARTAID800. Synd att man inte kan köra XTEND800 på LUX-NET (någon får väl lösa upp skyddet så vi kan använda programvran där!).

Vad gäller priser så kostande XTEND800 för något år sedan 2000 SEK och SMARTAID800 kostade 2850 inl moms i slutet på förra året. Obs att priset för XTEND är exlusive moms!

UTGIVNINGSPLAN

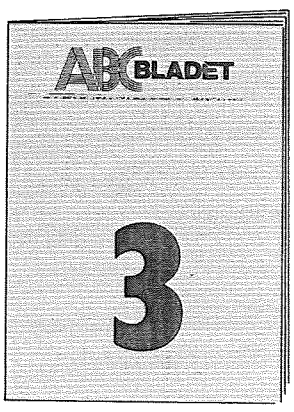
Nr 1, 1985
Manusstopp: 1 februari
Annonsbokning: 4 februari
Materialdag: 15 februari
Till tryck: 1 mars
Medlemmarna: 27 mars

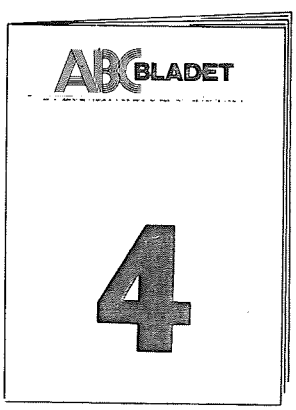




Nr 2, 1985
Manusstopp: 22 april
Annonsbokning: 29 april
Materialdag: 6 maj
Till tryck: 28 maj
Medlemmarna: 19 juni

Nr 3, 1985
Manusstopp: 15 augusti
Annonsbokning: 20 augusti
Materialdag: 29 augusti
Till tryck: 16 september
Medlemmarna: 11 oktober





Nr 4, 1985
Manusstopp: 21 oktober
Annonsbokning: 28 oktober
Materialdag: 4 november
Till tryck: 18 november
Medlemmarna: före jul

MSG

VAROOT(Variabel$) i assembler.

Text 390 - Torsten Frank <5452>

Hur gör man för att få VAROOT för en variabel i assembler ? Det enda man vet är variabelnamnet.

Text 391 - Matts Kallioniemi <2332>
Kommentar till text 390

En assembler-programmerare utan Bit För Bit i vänster näve ? Ska det vara långa variabelnamn så får du nog köpa Bit För Bit och göra det själv, för det är rätt jobbigt. Det bästa sättet är att göra varoot() i basic, och skicka nuffran till assembler-prg'et. Du kan alltså när du pokar ner prg'et skriva tex :

POKE Adr,33,VAROOT(Tjosan$),SWAP% (VAROOT(Tjosan$)) Det funkar alltså så att du skapar koden "LD HL,VAROOT(Tjosan$)" och det är vad jag antar att du vill. Om du absolut vill leta reda på variabeln i assembler så kan jag visa hur det går till för korta namn.

Text 392 - Torsten Frank <5452>
Kommentar till text 391

Ja tack, jag vill gärna att du visar hur man gör!

Text 393 - Matts Kallioniemi <2332>
Kommentar till text 392

Varför det? Men här kommer det i alla fall:

```
    LD  C,Typ
; varabeltyp & "nummer".
; bitarna : Osssstttt
; där ssss är siffan,
; tex 7 i A7% och ttt
; är typ enligt : A.=000
;                 A%=001
;                 A$=010
; ssss är alltså 1111
; om ingen siffa.
    LD  B,Namn  ; A=1, A=29
    LD  HL,(65326)
A0: LD  E,(HL)
    INC HL
    LD  D,(HL)
    INC HL
    LD  A,(HL)
    CP  B
    JR  NZ,A1   ; fel namn
    INC HL
    LD  A,(HL)
    CP  C
    JR  NZ,A1  ; fel typ
    INC HL ; HL=Varoot!
    JP  Klar
A1: EX  DE,HL
    LD  A,(HL)
    JR  NZ,A0
    INC HL
    LD  A,(HL)
    DEC HL
    JR  NZ,A0
    ; Den fanns inte, snyft
```

Det ska funka tror jag...
Allt enligt Bit För Bit, s. 100-104
Mycket elementärt

MSG

Det är ingen bug, det är en effekt

Text 580 - Patric Ljung <5455>

Jag vet inte hur det kom sig men jag hade ett programm inskrivet på en ABC806. Programmet ser ut så som nedan:

```
10 FLOAT
20 FOR I=0 TO 10 STEP .1
30   ; I
40   GET W$
50 NEXT I
```

Och alla som har en blekaste aning om basicprog. bör ju förstå vad detta program gör, eller ? Man får en utskrift av följande karaktär:

```
0
.1
.2
.3
.4
.5 osv...
```

Men det roliga är när den kom till 7.9 hände en liten kul pryl. Kolla!

```
7.8
7.9
7.9999(9)
8.0999(9) ! Tror jag att jag vill dra mig till minnes
8.2
8.3
```

Vad tycker ni om detta ? Om man gör en riktig avrundning stämmer det men... Jag har testat på 2 olika 806or och det blir likadant. Är det nån som kanske har en förklaring ?

Text 581 - Bo Kullmar <1789>
Kommentar till text 580

Det är ingen bug alls. Det är bara en effekt av flyttalshanteraren. Du vet man har en viss begränsning i sättet att lagra flyttal binärt. Sätt DOUBLE så får du fler "fel". I SINGLE så lagras ett flyttal i ABC800 i 4 byte och i 8 byte i DOUBLE.

Hade fler bytes använts för att lagra siffrorna, som det brukar vara i en stordator, så hade inte denna effekten uppträtt. Noteras kan också att flyttalshanteraren inte är samma ABC800 och ABC80.

Man skulle rätteligen själv ha avrundat i sådana här fall, t ex med PRINT USING '§§.§§' I. Hoppas det är begripligt. I några läroböcker för ABC80 har jag också sett något ditåt, som kanske förklarade det bättre än vad jag kan göra.

MSG

ABC800s kassettinspelningshastighet

Text 498 - John Stagg <5741>

Vet ni att man kan höja ABC800s kassettinspelningshastighet med nästan det dubbla ? Skriv POKE 65433,25. Det fina är, att det går att läsa in "snabba" program även om defaultvärdet (40) ligger i systemet. På min bandspelare går det till och med att minska bitlängden till värde 22, men då blir det visserligen något inläsningsfel då och då. Mycket beror på bandkvaliten.

Ni som har (som jag) fått vara utan skivenheten, testa detta ! Glöm ej förstås använda CASDEF i ditt START-program !

MSG

ABC80:s checksummor

Text 700 - Anders Franzen <5258>
Kommentar till text 696

Tack ! Tydligen är checksummorna 9913 och 10042 lika i det här fallet. Det vore kul att veta vad som skiljer dem åt. Vilken av dem är den senaste versionen ? Det finns tydligen tre olika checksummor. Vad jag vet som skiljer 11273 från 9913 är att den senare ej tystar ev ljud då tangent trycks ned.

Ett komplement till "Programvaran i ABC80" (rapport 1) vore intressant.

Text 705 - Benny Löfgren <2615>
Kommentar till text 700

Skillnaden mellan 10042 och 9913 är bara att de sista, oprogrammerade bytarna i det ena fallet är 255 och i det andra fallet 0, det påverkar alltså inte funktionen. Det finns lite fler skillnader mellan 10042/9913 och 11273, t ex så är den klassiska "DIM A$"-buggen rättad. Skillnaderna är inte så stora att de motiverar ett komplement till Rapport 1, men däremot kanske en artikel i bladet vore på sin plats.

Text 710 - Anders Franzen <5258>
Kommentar till text 705

Jag kollade slutet på min 9913 och där står det i ascii-form "DIAB '79". Just nu är det riktigt trevligt att ha en 9913. Skillnaderna mellan 11273 och 9913/10042 kan jag ju ta reda på själv genom att kolla i rapport 1 om det behövs. Jag har föresten hittat en bugg i 11273:an. På adress 60:89 står det 37, borde vara 38 !! Löjligt kanske, det är visst en konstant till rutinerna som räknar funktionen EXP som har sista siffran fel.

Förresten, kanske en till bugg:
Det går ej att definiera flera funktioner med DEFFN på samma programrad !

```
10 DEFFNA=1 : DEFFNB=2
20 ;FNB
```

funkar inte...

Text 712 - Hans Sjöberg <1196>
Kommentar till text 705

Nej, de sista byten i minnet är precis lika i 9913 och i 10042. Det enda som skiljer dem åt är en byte (minns inte vilken, men kan kolla upp det) som är NOP i 9913 och nå't annat (. tro det eller ej .) i 10042 och 11273...

Text 713 - Arne Hartelius <2731>
Kommentar till text 712

Adressen är 14412 och byten 10042-9913= 129.

MSG

Lever ABC80

Text 385 - Sven Wickberg <1384>

Det har varit intressant att se alla inlägg om gamla ABC80. Visst är det så att man inte skall välja dator, utan system etc. Och det är väl knappast troligt att nytillkomna professionella användare (företag o likn) skaffar en 80. MEN:

a) det finns massor av ABC80 ute

b) troligen kommer många nybörjare även i fortsättningen lättare över en 80 än en 800

c) tillbehören är vad jag förstår billigare till 80 än 800

d) 80 är fortfarande en mycket bra och användbar dator, inte minst att lära sig på

e) alla som redan har 80 och jobbar för fullt med den (och inte kan vill eller råd att byta) behöver stimulans och hjälp

DÄRFÖR tycker jag det är viktigt att klubben inte lägger av ABC80, utan håller i gång programutveckling för i varje fall hobbyisterna och de som är nya och håller på att lära sig.

Jag tvivlar på att 800 någonsin kommer att kunna bli på samma sätt "nästan var mans dator" som 80 har varit.

OM HEMBYGGE: Det är intressant att ta del av olika hembyggen och visst skall vi sprida information om vi kan. Men nackdelen med sådana saker är vanligen att det ena hembygget krockar med det andra. Därför begränsar de sin egen marknad.

Text 386 - Bo Kullmar <1789>
Kommentar till text 385

Naturligtvis finns det inga som helst planer på att "lägga av med ABC80" inom klubben! Orsaken till att det under en tid har stått lite om ABC80 i ABC-Bladet är helt enkelt bara att det har kommit in lite artiklar om ABC80. Jag har själv skrivit en hel del och förmedlat annat. Detta matrial som jag har förmedlat eller skrivit har naturligtvis inte handlat om den för mig ganska okända datorn ABC80.

Alltså fram för fler program och artiklar om ABC80 !!! Skriv på !!!

SVENSK STANDARD SS 01 16 01

Utgåva 3

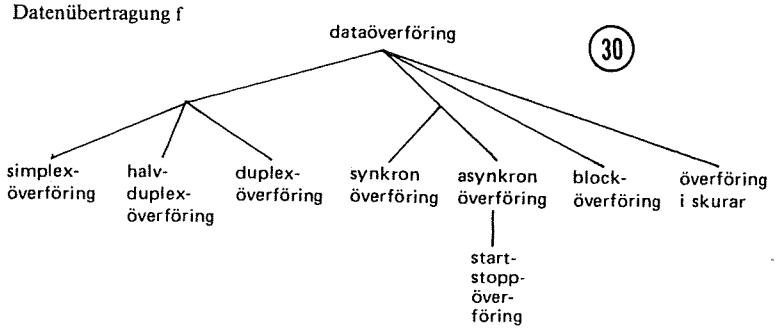
dataöverföring, överföring →30

överföring av data från en plats till en annan med hjälp av teleteknisk utrustning; jfr *datakommunikation*

E data transmission

F transmission f de données

D Datenübertragung f



8 kbyte extra CHAINARE

8 kbyte extra PROM-minne i tangentbordet på ABC80

Utöka PROM-minnet i tangentbordet från 16 kbyte till 24 kbyte.

Den som vill bygga om sin ABC80 till att få 3*8kb EPROM-minne av typ 2764 i stället för de fyra om vardera 4kbyte av typ 2532 eller 4732 som innehåller BASIC-tolken i originalutförande, skall ändra sin ABC80 enligt nedan. Man skall sedan använda 2764 EPROM. Man kan då använda det 3:e promet, dvs 8 kbyte för egna rutiner.

Arbetsbeskrivning.

Skaffa dig 2 EPROM med BASIC-tolken först. Montera av kåpan till tangentbordet. Skruva sedan bort tangentbordet och kylflänsen från bottenplattan. Börja med att kapa bort tre byglingar i folien på kortet. Bygling 11,18,20 kapas.

Bygling	sitter i koordinat
11	E5-6
18	B1-2
20	E8

Använd blanktråd och löd in tre nya byglingar på nr 10, 12 och 16.

Bygling	sitter i koordinat
10	E8
12	E5-6
16	B1-2

Ta bort alla byglingar på A1-B1 och löd sedan tillbaka dem men med början från första positionen från vänster.

Jaha nu är alla modifieringar på kretskortet gjorda. Nu gäller det bara att ha två EPROM med den BASIC-tolk du vill använda att stoppa i socklarna. Jag har EPROM förABC80 med checksummorna 11273 och 9913. Montera nu i de två nya EPROMmen i de bortersta socklarna och det tredje som du har programmerat själv i den tredje positionen. Den fjärde sockeln kan inte användas nu. Montera ihop datorn så skall du se att du kan använda dina 8kbyte PROM-rutiner.

<1005>
John-Erik Näslund

En rutin som tar över tangentbordet i kommandomod för ABC 800-serien.

Hämta hem CHAINARE.HEX (eller CHAINARE.BAC med H-optionen om någon konverterat tillbaks filen i monitorn) och konvertera till CHAINARE.BAC med programmet HEXTOFIL (finns i ABC800/UTILITY).

När man kör CHAINARE läggs en liten maskinkods-snutt i minnet. När datorn kommer i kommando-mod, dvs inte exekverar något program, tar denna snutt över kontrollen genom att starta ett förinställt BASIC-kommando.

Detta är särskilt lämpligt att använda i menysystem med många program. Det BASIC-kommando som används får då vara "RUN <menyprogrammets namn>", så startar alltid menyn så fort något program har slutat exekvera. Även om exekveringen bryts med CTRL-C startar menyn. Starta det hela med CHAINARE, så startar det menyprogrammet.

Programmet innehåller från början kommandot "RUN SPELMENY"+CHR$(13). För att ändra, gör så här:

```
OPEN "CHAINARE.BAC" AS FILE 1
POSIT §1,27
PUT §1 "nytt kommando..."+CHR$(13,0)
CLOSE 1
```

Kommandot får vara högst 21 tecken långt inklusive CHR$(13,0). Det är koden 0 som markerar kommandots slut. Koden 13 kan utelämnas om så önskas, men eftersom datorn då inte kommer ur kommando-mod, så skickas kommandotexten om och om igen, i all oändlighet.

Konstruerat och inskickat av Kristoffer Eriksson <5357>.

MSG

Vilken veckodag är det ?

Text 227 - Bo Kullmar <1789>

Kan man få reda på veckodag när man läser av systemklockan i LUX-NET centralen ? Rimligtvis måste det väl gå för man måste ju ha det om man skall ställa klockan i en 806:a.

Text 229 - Matts Kallioniemi <2332>
Kommentar till text 227

Kan du väl räkna ut själv ? Här kommer ett färdigt program för alla lata:

```
10 INPUT YY,MM,DD
20 IF MM>2 THEN 50
30 FAK=365*YY+DD+31*(MM-1)+INT((YY-1)/4)
   -INT(.75*(INT(((YY-1)/100)+1)))
40 GOTO 60
50 FAK=365*YY+DD+31*(MM-1)-INT(.4*MM+2.3
   )+INT(YY/4)-INT(.75*(INT(YY/100)+1))
60 DAY=FAK-7*INT(FAK/7)
70 DAY$=MID$("Lördag Söndag Måndag Tisd
   ag Onsdag TorsdagFredag ",7*DAY+1,7)
80 PRINT "Det är dag nummer" FAK ". En
    " DAY$
```

Luxor har såvitt jag förstår beslutat att enbart tillverka tangentbordet ABC99. Tangentborden ABC77, ABC55 och ABC22 kommer alltså inte längre att tillverkas.

ABC99 har lite fler tangenter än de tidigare tangentborden, bl a så finns det en ESC tangent. Det finns också stor fyrkantig markörförflyttningstangent samt möjlighet att ansluta mus. ABC99 togs fram till ABC1600, men kommer nu alltså att säljas även till ABC806/802.

Hitintills har det inte funnits program till ABC806/802 som har varit anpassade till ABC99. Luxors beslut tvingar fram sådana programversioner.

I ett MSG-inlägg som publiceras på annan plats finns en förteckning över vilken standard i programmen som Luxor skall använda för systemprogram. De rekommenderar att program görs så att de olika tangentbordens funktioner dubbleras så att programmen fungerar på olika tangentbord utan att programmen behöver känna av vilken typ av tangentbord som används.

Bo Kullmar

MSG

Tangentbordens liv

Text 1146 - Bernt Johansson <3384>

Det har diskuterats om tangentbordets, ABC77, vara eller icke vara. Det kommer ju att finnas många ABC77 i användning i fortsättningen också för alla vill nog inte slänga bort sina bara för att ABC99 finns nu.

Jag tror att det inte är nödvändigt att känna av, i varje program, om det är 77 eller 99. Det är enklare att dubblera funktionerna så att dom är åtkomliga med olika tangenter. Så här har vi tänkt oss att göra med dom systemprogram vi gör på Luxor Datorer.

Utökad standard för funktionstangenter för program med menyer och formulär.

Denna standard är inte helt tillämplig på ordbehandlings- och CAD- och bild-program.

Följande specialtangenter används:

Tangent	HEX	Decimal
BS	08	8
<-	08	8
TAB	09	9
->	09	9
RETURN	0D	13
CTRL+P	10	16
CE	18	24
CTRL+X	18	24
DEL	7F	127
HELP	80	128
STOP	81	129
PRINT	82	130
I<-	88	136
->I	89	137
PAD-UPP	A1	161
PAD-NER	A3	163
PAD-HÖGER	A4	164
PAD-VÄNSTER	AC	172
SHIFT+PAD-UPP	B1	177
SHIFT+PAD-NER	B3	179
SHIFT+PAD-HÖGER	B4	180
SHIFT+PAD-VÄNSTER	BC	188
PF1	C0	192
PF5	C4	196
PF7	C6	198
PF8	C7	199
SHIFT+PF5	D4	212
SHIFT+PF7	D6	214
CTRL+PF4	E3	227

Tangentbord

Cursorstyrning under fältinmatning (FNIn$)

Funktion	Tangent
Radera ett tecken	PF8, DEL
Radera fältet	CE, CTRL+X
Cursor vänster	PAD-VÄN, BS, <-
Cursor höger	PAD-HÖG, TAB, ->
Verkställ inmatning	RETURN
Skriv (skärmdump)	PRINT, CTRL+P

Funk$ värde vid avslutning av fältinmatning i fältets första position (FNIn$)

Funktionsvärde vid flygande tangentbordsavläsning (FNInkey$)

Funktion	Tangent	Funktionsvärde
Hjälp	HELP, CTRL+PF4	HE
Uthopp, upp i menyträd, stoppa utskrift	PF1, STOP	EX
Cursor i toppen	SH+PAD-UPP, SH+PF5	CH
Cursor upp	PAD-UPP, PF5	CU
Cursor vänster	I<-, SH+PAD-VÄN	CL
Cursor höger	->I, SH+PAD-HÖG	CR
Cursor ner	PAD-NER, PF7	CD
Cursor i botten	SH+PAD-NER, SH+PF7	CE

MSG

Luxors servicemanualer

Text 329 - Göran Altius <123>
Kommentar till text 328

Jag tror att det står mycket intressant i dessa servicemanualer. Efter att jag bläddrade lite i en sådan på ABC-dagen i klubblokalen tror jag att de kontinuerligt uppdateras av Luxor (upptäckta fel mm). Finns det något liknande för Luxors programvara ?

Kan man på något sätt prenumerera på denna värdefulla information som servicemanualen uppdateras med ?

Text 333 - Bo Kullmar <1789>
Kommentar till text 329

Ja, du kan prenumerera på servicemanualerna om jag inte minns fel. Kontakta Luxor för information. (Vi får dem gratis)

Fildump

Här kommer ett tips till de, som jobbar med assemblerprogrammering på ABC 80 och som vill använda befintliga rutiner från ABC 80, men med egna justeringar eller tillägg.

På ABC-kassett nr 2,fanns en disassembler, som heter 'NYMEMDMP.BAS'. Ett program,som enligt min åsikt är en liten pärla, bland många andra. Nåväl - det finns dock en sak i detta program, som jag saknar och det är en fildumprutin av den källkod, man får ut av programmet.

För att råda på detta kan man göra följande tillägg/komplettering av programmet:

```
 305 ; 'D  = Fildump'
 550 ; : ; 'F/B/A/+/-/1-7/R/N/" "/C/I/M/
     K/P/D';
 560 GET Ö$
 561 ON FNB('FfBb++--NnAa  IiKkPpMmRrCcD
     d') GOTO 650,570,580,590,600,320,61
     0,620,1630,1670,1750,1960,630,640,2
     010
1650 IF Y$="PR:" OR Y$="V24:" ; §3A$;LEF
     T$(Ä2$,L9%*3%);SPACE$(3%*(5%-L9%))
     ELSE ; §3"              ";P$
1810 IF Y$="PR:" OR Y$="V24:" ; 'CTRL-J=
     lf/L=ff/Ä=esc/T=cond/S=ejcond'; : G
     ET Ö$ : ; §3Ö$;
1820 IF Y$="PR:" OR Y$="V24:" ; §3SPACE$
     (V9%);
1830 IF Y$="PR:" OR Y$="V24:" FOR Ö%=1%
     TO 25% : ; §3'- '; : NEXT Ö%
1840 IF Y$="PR:" OR Y$="V24:" ; §3 : GOT
     O 1770
1860 IF Y$="PR:" OR Y$="V24:" ; §3SPACE$
     (V9%);Ä$;
1885 GOSUB 1930 : ; "Ange devicenamn(tex
     PR:)" : INPUT Y$
1890 GOSUB 1930 : ; 'Parametrar för open
     print/filnamn'; : INPUT Ö$
1895 IF (Y$<>"PR:" AND Y$<>"V24:") AND Ö
     $="" GOTO 1890
1900 OPEN Y$+Ö$ ASFILE 3%
2000 REM * in fildump *
2010 GOSUB 1930 : ; "Dump t.o.m. "; : IN
     PUTLINE Ö$
2020 M=VAL(LEFT$(Ö$,LEN(Ö$)-2%))
2030 ONERRORGOTO 3000
2040 GOTO 1630
3000 REM * errortreat *
3010 ONERRORGOTO 0
3020 IF ERRCODE<>21% THEN END
3030 PREPARE Y$+Ö$ ASFILE 3%
3040 GOTO 1900
```

'Merga' in dessa rader i det befintliga programmet och spara det. Sedan är det bara att tuta och köra. Byt ut absolut adresser mot lablar och lägg in kommentarer etc i källkoden. Detta görs lämpligen med TV-editorn. Tänk på att källkoden ska få plats i editorn för justeringar/ändringar.

LYCKA TILL

Bo Sehlberg <3027>

Om TV-editorn spårar ur

Vid ett flertal tillfällen har jag haft "otur" med TV-editorn på så sätt att den spårar ut. Förföljd av otur har jag naturligtvis använt en version av TV-editorn som inte har haft den förträffliga raden 10. Texten som jag har knappat in har förstås varit uppåt tiotusen tecken och då drar man sig ju för att behöva göra om det.

Efter en del funderingar, tester och rundfrågningar har följande visat sig att rädda texten till ett skick som gör den möjlig att bearbeta igen.

Om Du skriver RUN <RETURN> startar TV-editorn igen så detta bör Du undvika i detta skede.

Utan att rensa skärmen eller göra något annat, skriv som kommando på en rad:

PREPARE 'DRX:<FILNAMN.EXT>' AS FILE 9 : ; §9,M$ <RETURN>

X skall ange det drivenummer vart Du vill ha texten lagrad. Filnummer 9 kan väljas till något annat om Du har den upptagen redan, det viktiga är att man skriver på den fil Du öppnar med PREPARE.

<FILNAMN.TXT> får anges efter eget skön, det ändrar Du med RENAME. Vanliga regler för benämning av filer gäller även här.

Varför det fungerar?

Jo, Tv-editorn har hela texten i en sträng, M$. Vi skapar en fil och skriver in strängen på denna fil på skivan.

Därefter kan Du skrivar RUN <RETURN> och läsa in filen igen och fortsätta. Det som ligger i inmatningsbufferten går däremot förlorat med denna medtod, men det rör sig i normalfallet om högst femhundra tecken.

För att undvika dessa malörer i fortsättningen har jag lagt in red 10 i alla uppsättningar av Tv-editorn som finns på mina skivor.

Ulf Sjöstrand <1208>

Pressläggningsmeddelande:

Vi har under pressläggningen fått påpekande från Konsumentverket att våra annonsörer skall ange sina priser inklusive moms. Vi har ännu ej nått alla med detta meddelande varför priser i detta nummer anges exklusive moms.

MSG

VDO-trubbel

Text 224 - Anders Andersson <5519>

Gediget innehåll på kassetten, må jag säga!

Lite trubbel dock: Jag anser mig inte vara i behov av printerrutiner till VDO, varför jag (helt enligt anvisningarna) ersatte skrivarflaggan "-1" med "0" (rad 960 i den aktuella utgåvan). Resultatet av detta blev dock att enhetslistan var kvaddad när jag lämnade VDO. Pekaren ansåg att listan började på 0015H, vilket knappast kan vara lämpligt, eller?

Om det kan ha någon betydelse kan jag upplysa om att jag har ABC80 med checksumma 9913.

Text 244 - Kjell Svensson <5318>
Kommentar till text 224

För att få VDO att fungera utan dess egen printerrutin, måste man ändra detta.

I basicprogrammet VDO2.32k ändra rad 500 till:

500 READ Ö$: IF Ö$="SLUT" 545 ELSE 460

En rad 545 måste läggas till:

545 IF NOT S9% 880

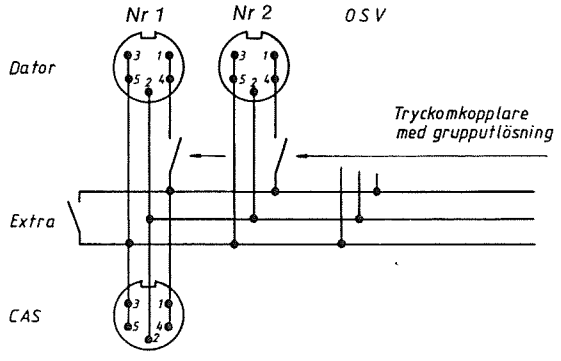
Med dessa ändringar klarar sig pekaren till enhetslistan helskinnad.

Tips

Om man har flera ABC80, t ex i skolor, och bara en bandspelare kan man parallellkoppla utgången med datorernas ingångar, så att flera datorer laddas samtidigt med samma program. Jag har provat med 8 stycken. Låt en valfri dator styra motorkontakten, och skriv RUN CAS: på samtliga maskiner. Tryck till sist på RETURN samtidigt på alla datorerna. Om man vill spara program eller filer får endast en dator inkopplas åt gången. Detta ordnas enklast med tryckomkopplare med grupplösning. (se kopplingsschema). Har man tvåpoliga strömbrytare är det lämpligt att även bandspelarens motorstyrning sker med omkopplaren.

Om den extra brytaren sluts kan program eller filer flyttas från en dator till en annan. Skriv SAVE Prog på den ena och run CAS: på den andra datorn och tryck samtidigit på RETURN, så överförs programmet.

<1014>
Sven Olby
Ragnaröksgatan 12
021-204 94



VDO-editor igen!

Först = 1: Per Melöy i Drammen, Norge.

Har skickat in en ändrad rutin till C-kolon vilken återhoppar in i VDO-editorn efter LIB-läsning. Har man fått felet ERR 21 vid hämtning av fil som inte finns får man dock inget återhopp längre. RunOnly-flaggan är borta då.

```
1 REM save    VCa
58 POKE 63968%,61%,200%,24%,242%,58%,98
   %,24%,254%,175%,202%,98%,24%,195%,96
   %,24%
59 POKE 63983%,205%,22%,249%,213%,221%,
   229%,205%,2%,0%,205%,118%,2%,221%,22
   5%,209%,201%
60 POKE 63757%,195%,239%,249%,195%,239%
   ,249%
63 A%=CALL(63744%) : GOTO 42
```

Andra = 2: Jag har pysslat med den rutin jag skrev om i 2-85. Glad i hågen satte jag mig efter semestern för att fylla på mina tabuleringsrader 414 - 419 och 434 - 439 samt 1054 - 1059. Men TJI! Där hade jag fingrarna med i munnen när jag bet i mackan! Jag fick ERR 35 Checksummafel vid läsning direkt vid en enda ytterligare tabuleringsrad 414, 434, 1054. När jag tog bort dessa gick det igen. Således går det för närvarande endast med fyra olika tabuleringsrader. Se nedan:

```
 410 ONERRORGOTO 410 : ; CHR$(7%,12%)CU
     R(6%,0%)'TAB 10  ?
     TRYCK 0'
 411 ; 'AC 1 ?  AC-TABELL 10 TPT  TRYCK
     1'
 412 ; 'AC 2 ?  AC-TABELL 12 TPT  TRYCK
     2'
 413 ; 'AC 3 ?  AC-NO-TABELL      TRYCK
     3'
 414 REM
 422 GET A$ : Ö%=VAL(A$) : IF Ö%<0% OR
     Ö%>3% 410
 423 IF Ö%=0% 430
 425 ON Ö% GOTO 431,432,433
 430 RESTORE 1050 : GOTO 444
 431 RESTORE 1051 : GOTO 444
 432 RESTORE 1052 : GOTO 444
 433 RESTORE 1053 : GOTO 444
 444 FOR I%=0% TO 8% : READ Ö% : POKE A
     %+36%+I%,Ö% : IF Ö%<>-1% NEXT I%
1050 DATA 9,19,29,39,49,59,69,79,-1,"Ta
     bulatorinställningar"
1051 DATA 5,12,28,32,41,68,72,84,-1,"Ta
     bulatorinställningar"
1052 DATA 6,15,34,38,48,81,87,100,-1,"T
     abulatorinställningar"
1053 DATA 14,47,87,-1,"Tabulatorinställ
     ningar"
1060 DATA 20,6238,10,175,1,240,216,SLUT
1070 DATA VR,80,4
```

Claes

Brevlådan

Det har varit en glädjande respons från läsekretsen kring flera av de artiklar som publicerades i förra numret. Det är stimulerande med tvåvägskommunikation (även utanför monitorn).

Per Erik Holmberg från Katrineholm fäste min uppmärksamhet på att programmet PRIMTAL släpper igenom tal som 1011 och 1101. Man ser ju genast att siffersumman är 3 och att dessa tal därför inte kan vara primtal utan är delbara med 3.

När jag tittade närmare på saken fann jag en rad andra, inte fullt så synliga 3-produkter och drog slutsatsen att sållningen med 3 inte fungerade. Förklaringen är nog så enkel, men motiverar ytterligare några artikelrader.

Den grundläggande primtalstabell som behövs för operationerna i programmet vaskas fram med Erastotenes såll och placeras (rad 1210) i vektorn T(N). De visas samtidigt på skärmen, och man ser att 3 är med.

(Bara ett ord om vektorer. Vektor kallar man en indexerad variabel med ett index, alltså T(1), T(2), T(3) osv. Om det är två index kallas den matris, alltså T(X,Y). Fördelen med en indexerad variabel är att man i en slinga kan köra med samma beteckning T() och bara ändra index.)

Från vardagslivet är vi vana vid att räkna 1, 2, 3 osv precis som i exemplet ovan, och så har också skett i programmet. Tror man. Men då tror man fel! Jag upptäckte när jag tittade efter att T(1)=5 och inte alls =3 som den borde vara.

Vart tog den lilla trean vägen? Jo, T(0)=3. Anledningen till detta är lärorik. Första gången rad 1210 genomlöps har variabeln N ännu inte nämnts i programmet, och enligt reglerna för Basic är då N=0.

Saken åtgärdas enklast genom att man accepterar T(0) och i ändrar rad 460 till

```
460 FOR J=0 TO N+1
```

Så har jag gjort i den rättade versionen i programbanken - ju simplare ju enklare.

Men en bättre metod - tror jag - är att i stället definiera N=1, antingen på rad 1195 (omedelbart före den slinga där N används) eller i samband med DIM-raden i början av programmet. Då har man också en möjlighet att skriva in vad N är för någonting.

Här demonstreras en av Basics svagheter. Det som är en fördel för nybörjare blir en nackdel i mera avancerade sammanhang. Man kan i Basic arbeta med icke i förväg deklarerade variabler. De tilldelas då värdet noll resp tom sträng. Då kan det i värsta fall hända - som här - att man inte märker tankelapsusen genast, efersom programmet snabbt fyller på med mera synliga värden.

I språk som Pascal kan man inte skriva en programrad utan att först metodiskt och omständligt tala om vilka variabler man vill använda, vad de skall ha för form och egenskaper osv. Tröttsamt och tråkigt för nybörjaren, men troligen fördelaktigt för yrkesmannen. (Några kommentarer från dem som har erfarenhet av detta?)

En annan lärdom kan också hämtas: Även mycket ordentligt genomspekulerade logiska steg KAN innehålla felaktigheter som inte genast visar sig. Av naturliga skäl kan man ju inte kontrollera ALLA resultat manuellt. Somliga fel kommer fram först efter en tids användning. Det är klokt att inte alltid utgå ifrån att det som kommer ur en dator är rätt!

När jag hade delgivit Per Erik min rättelse löste han också ett annat problem. Som jag rapporterade i artikeln vägrade min ABC80 att befatta sig med tal över 500 000. Varför just detta värde? Jag misstänkte flyttalsnotationen, men det händer ju inget särskilt med den vid 500 000.

Per Erik rapporterade att med ovanstående rättelse införd började datorn konstra vid 300 000. Det måste ha att göra med det lägst primtalet i tabellen, förut 5, nu 3.

```
500000/5 = 300000/3 = 100000
```

och si DET var begripligt, för efter 99 999 övergår ABC80 till exponentialform. Skriver man från tangentbordet 123456 kommer det ut som 1.23456E+10. Om man på annat sätt vet att talet innehåller exakt 6 siffror, kan man ju avläsa och använda dem alla, men datorprogrammet garanterar inte talets utseende efter sjätte siffran och vägrar blankt att utföra operationer som INT(A) om A där A>99 999.

Det är alltså inte primärminnet som bromsar, utan talnotationen. Nu vet vi det.

Sven Wickberg

512 KB internt RAM-minne i ABC806

ELJI Elektonik marknadsför en sats 256 KBit minneskapslar till ABC806. Innehavare till företaget är medlemmen Lennart Jansson, .620>. Jag har köpt en sådan sats minneskapslar enligt en annons i ABC-Bladet nr 3, 1984.

Man får 16 st 256 KBits minneskapslar och en skiva. Man byter ut 64 KBit minneskapslarna till grafikminnet enligt anvisningarna och därmed får man 4 gånger mer extra RAM-minne.

På skivan finns Luxors/DIAB:s original RAM:rutin som är rättad så att den fungerar med 256 KB minneskapslar. Därmed kan man använda 1888 sektorer i RAM-floppyn, om man inte reserverar minne för HR-grafikbilder. Vill man använda hela minnet för grafikbilder så kan man använda 16 grafikbilder.

Det finns klara anvisningar för hur man byter minneskapslarna. Det är dock ganska pillrigt, man får vara försiktig.

ABC-klubben använder REG800 för att köra medlemsregistret. Vi har sorteringsfilen upplagd på RAM:. Vi har inte kunnat skriva ut registret med REG800 sorterat på grund av att vi inte har tillräckligt plats för sorteringsfilen. Vi har en 8" drive och det är så trångt på disken att indexfilerna måste ligga på programskivan.

Efterut byte av minneskapslarna i 806:an har vi inte längre något problem med att sortera med REG800.

Man kan också själv köpa minneskapslarna av företag som säljer sådana, det blir billigare. Sen får man då se till så att man får tag på RAM:-rutinen till ABC806:an. Genom att köpa paketet från Lennart Jansson så får man allt på en gång.

Bo Kullmar

ABC-klubben och Kronstat

ABC-klubben distribuerar från och med nyåret 1986 programpaketet Kronstat

Kronstat är ett statistikprogrampaktet som utvecklats inom Karolinska Institutet, Institutionen för Socialmedicin och Vårdcentralen Kronan. Som kontaktperson gentemot ABC-klubben finns Anders Lindeberg. Programpaketet beskrevs kortfattat i ABC-bladet 3, 1985.

Kronstat fungerar idag på ABC800/802/-806 och Facit DTC. formaten på disketterna kan vara 5 1/4" (830 och 832) och 8" (838). Diskettdrivrarna med minst 160 kB per drive behövs. Med 160 kB disketter ryms dock ej hela Kronstat på en diskett utan man tvingas byta mellan två disketter.

Programdiskett och manual, version 2.1, daterad december 1985 kan beställas från ABC-klubben. Klubben tar ut en distributionskostnad mm om 100:-.

Enklast får får Du manual och diskett genom att sätta in 100:- på ABC-klubbens postgirokonto 62 93 00-5 och ange en entydig avsändare samt till vilken kombination av maskiner Din Kronstat skall användas.

Hej på Er i styrelsen för ABC-klubben

Äntligen har jag lyckats fatta mod och skriva till Er. Tråkigt bara att detta första försök dessvärre måste knackas ner på en "SKRIVMASKIN" och inte på min ABC80. Men vad gör man när kunskaperna saknas för detta. Jo naturligtvis tar man fram pärmen "Välkommen till ABC-klubben" och letar efter svaret, och finner att svaret är att exempelvis artiklar till ABC-bladet skall insändas i maskinläsbar form. Men hur gör man?? Förlåt en så enkel fråga för er som kan detta. Är man ett "BLÅBÄR" så är man, och man vill ju inte fråga kompisarna och visa sin okunnighet för dom är ju "PROFFS".

Har dessvärre letat förgäves efter "MATRIKELN" till medlemmarna. Till min lycka hittade jag att klubben är till för att bl a verka för ökade kunskaper till nytta, utbildning och nöje, samt att ledstjärnan för verksamheten är "ERFARENHETS-UTBYTE". Har inskaffat tidigare utgivna publikationer med kassetter 1-11 och skall skaffa resterande 12-13. Övriga har jag erhållit som medlemsförmån. Har läst igenom gamla ABC-bladet och hittat artiklar för nybörjare som jag, har ni slutat med dessa för mig mycket intressanta artiklar. Det måste väl komma in nya medlemmar hela tiden?

Varför inte skriva artiklar för oss "BLÅBÄR" om hur vi skall kunna ha mesta nytta av vår dator. Börja gärna från början och skriv inte över huvudet på oss så att vi inte förstår.

Kan detta vara en orsak till att många medlemmar har försvunnit? Jag har även problem med vissa program men detta beror väl på min okunnighet och inte på programmen.

Jag har en ABC80 med 32 k och kassettbandspelare. Modem har jag aldrig provat på. Jag använder min dator till privat hobbybruk med program som andra hjälpt mig med.

En som vill veta mer

<5475>
Svenne

AVBROTTS-HANTERING PÅ ABC80.

Den här artikeln är tänkt att ge insikt i ABC80:s avbrottshantering eller interrupt som det heter på engelska. Ganska ofta hör man ju talas om interrupt och programmering av PIO, men hur fungerar det i egentligen? I början är det något teoretiskt, men mot slutet beskriver jag exempel på praktisk tillämpning.

Först och främst bör man ha klart för sig att avbrottshantering på ABC80 och dess processor Z80 är mycket generell och erbjuder stora möjligheter till egna experiment.

Vad är då avbrottshantering?

Man kan jämföra det med hjärnans funktion: Tänk dig att en person exempelvis står och diskar. Plötsligt ringer telefonen och personen avbryter diskandet för att ta telefonen och svara. Efter att samtalet är slut fortsätter han där han slutade (får man anta i alla fall!)

På samma sätt är det i datorn: Någon (enhet) utifrån vill omedelbart ha kontakt med processorn och får den att sluta med allt den håller på med för ögonblicket. När jobbet "på sidan om" är klart, fortsätter processorn där den var förut.

Att man har den möjligheten är nämligen en stor fördel: Antag att datorn är satt till att bevaka en vägkorsning: Då en fotgängare vill gå över gatan trycker han på knappen. Processorn måste på något sätt upptäcka detta, och då vore det slöseri med värdefull datorkraft om den enbart hade till uppgift att kontinuerligt känna av (polla) tryckknappen och först vid positivt resultat gå vidare i programmet. Istället kunde den användas till att göra statistik över trafiken medan den väntar på fotgängare!

Rent praktiskt signalerar en inkanal en s.k. avbrottsbegäran till processorn och den kommer då att hoppa till en subrutin (assembler!) i minnet. När subrutinen är slut, hoppar processorn automatiskt tillbaka där den var innan avbrottbegäran kom. (Ungefär som GOSUB och RETURN i BASIC).

Z80-systemet har flera olika typer av avbrott: Processorkapseln har två ben som heter $\overline{\text{INT}}$ (ben nr 16) och $\overline{\text{NMI}}$ (ben nr 17) och dessa är alltså de två interruptingångarna som processorn har. De fungerar på lite olika sätt och därför händer det lite olika saker.

Om processorn skulle få $\overline{\text{INT}}$ och $\overline{\text{NMI}}$-signal samtidigt, har $\overline{\text{NMI}}$ högst prioritet och väljs alltså först. En annan sak är att $\overline{\text{INT}}$-ingången mjukvarumässigt kan stängas av (processorn ignorerar signalen), vilket man aldrig kan göra med $\overline{\text{NMI}}$-ingången, men det hörs ju på namnet: Non-Maskable Interrupt.

$\overline{\text{NMI}}$:

Processorn hoppar alltid till adress 102 (decimalt).

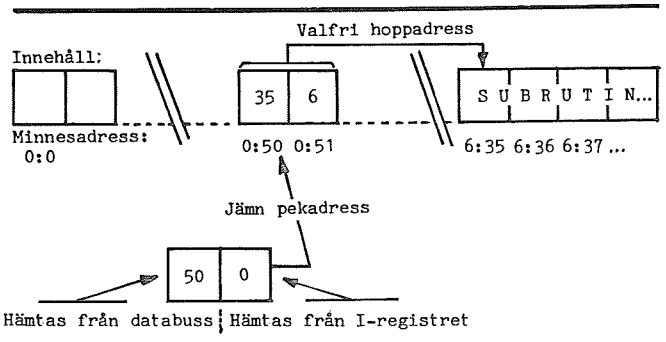
$\overline{\text{INT}}$:

Tre olika saker kan hända beroende på vilken avbrottsmode processorn är i. Denna mode kan ändras med en assemblerinstruktion.

MODE 0: Processorn hämtar en 1-bytes assemblerinstruktion på databussen, som en yttre enhet lagt dit, och denna instruktion talar om vad processorn ska göra. (Samma avbrottshantering som processorn 8080 har!)

MODE 1: Processorn hoppar till adress 56 (decimalt). Som synes är skillnaden liten jämfört med $\overline{\text{NMI}}$!

MODE 2: Processorn hoppar till en valfri adress i minnet. Denna helt valfria adress ska finnas lagrad på ett annat ställe i minnet med en annan adress! Denna adress består givetvis av två byte där den högsta byten hämtas från processorns I-register, och den lägsta byten hämtas från databussen. Denna bytes ska förstås ha blivit placerad där av en yttre enhet. Tänk på att den bildade adressen måste vara en jämn adress p.g.a. avkodningen i PIO. (Bit 0 nollställd). Denna typ av adressering brukar kallas indirekt adressering:



OBS! Alla siffror är bara exempel!

Hur ser det nu ut i ABC80:n? Används alla tre avbrottsmode:n?

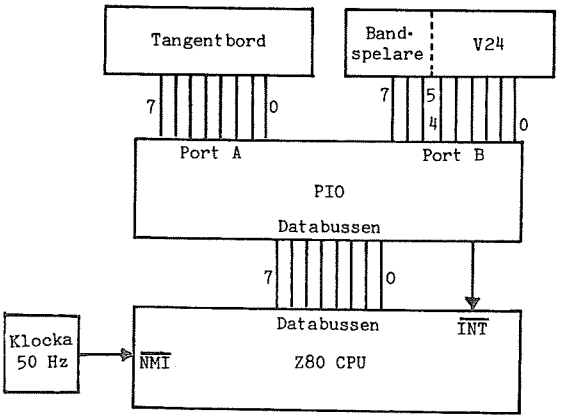
$\overline{\text{NMI}}$:Ingången är kopplad till en 50 Hz klocka. Så länge ABC80 är tillkopplad gör alltså processorn CALL(102) var 20 millisekund. En vän av ordning undrar då kanske vad för subrutin som kan tänkas finnas där? Svaret är att där finns instruktionerna som räknar ner ABC80:s realtidsklocka. (Cellerna 65008-65010).

Eftersom detta avbrott sker så pass ofta har man varit mån om att ha subrutinen så kort som möjligt för att inte datorn skulle "slöas" ner för mycket.

$\overline{\text{INT}}$: Ingången är kopplad till PIO dvs den krets som sköter in/utmatningen från tangentbord och kasettbandspelare/V24. PIO har två stycken portar A o B där tangentbordet är kopplat till port A och kasettbandspelaren är kopplat till bit 7 och bit 6 i port B. Bit 5 är motorstyrning. De andra fem bitarna är kopplade till V24-snittet. Som jag redan nämnt kan PIO sedan programmeras till önskad funktion. När ABC80 tillkopplas så väljs avbrottsmode 2, I-registret är nollställt och PIO programmeras så att $\overline{\text{INT}}$-signal ska skickas då bit 7 i port A ettställs.

Denna bit ettställs nämligen varje gång en tangent trycks ned. (Det sköter elektroniken i tangentbordet om). När man trycker på en tangent nu, skickas alltså $\overline{\text{INT}}$-signal till processorn som då hoppar till datorns tangentbordsrutin. Vidare programmeras PIO att den ska skicka ut en byte på databussen samtidigt med $\overline{\text{INT}}$. Man har valt 52 decimalt.

När kasettbandspelaren ska användas programmeras PIO tillfälligt om så att INT-signal istället ska skickas då bit 7 i port B ettställs. Dessutom ska PIO istället skicka ut 54 decimalt på databussen. Denna bit ettställes nämligen varje gång det "låter" på bandet. (Det sköter bandspelaren om).



Pion programmeras så att bit 7 i port A eller B orsakar signal

CPU programmeras så den står i avbrottsmode 2.

Nu ska vi titta på hur en avbrottssubrutin ska se ut:

I princip kan man ha vilken liten assemblersnutt som helst, det är bara ett par saker man bör tänka på:

Ett.

Om man i rutinen tänker använda några register ex. HL BC eller AF måste man först se till att spara deras innehåll på stacken med PUSH för att efter rutinens slut återställa värdena från stacken med POP.

Orsaken är ju uppenbar. Om man inte gjorde detta skulle datorn kunna spåra ut varje gång det kom en avbrottbegäran. Den måste kunna fortsätta där den var innan avbrottet utan att få förstörda registerinnehåll.

Två.

Vad händer om datorn får en ny avbrottsbegäran medan den fortfarande är kvar i avbrottssubrutinen? Om avbrottssubrutinen är lång är det inte alls omöjligt att så sker. Vi kan t.ex. trycka på två tangenter kort efter varandra. Givetvis vore det inte så lyckat, i värsta fall dyker datorn. Vi måste alltså se till att stänga av INT med en assemblerinstruktion DI (243 decimalt) i början av subrutinen för att alla eventuella fortsatta $\overline{\text{INT}}$-signaler ska ignoreras.

I slutet av rutinen måste vi förstås vara noga med att sätta på $\overline{\text{INT}}$ igen med EI (251 decimalt), annars kan vi ju inte använda avbrottshanteringen!

En god regel är att alltid avsluta sina subrutiner med EI!

Tre.

Subrutinen ska givetvis avslutas med ett RETURN, så att processorn vet att subrutinen är slut och att den kan återvända.

Den vanliga RET (201 decimalt) kan inte användas till detta, utan man använder RETN (237,69 decimalt) respektive RETI (237,77 decimalt) RETN användes som avslutning på subrutiner orsakade av $\overline{\text{NMI}}$, och RETI användes som avslutning på subrutiner orsakade av $\overline{\text{INT}}$.

Schematisk uppbyggnad: Avbrottsrutin orsakade av INT-signal.

```
DI          . : Disable interrupt. Stänger av.
PUSH HL     : Spara önskade register på
PUSH BC     : stacken.
PUCH AF
-------------------------------------------
Avbrottssubrutin HL, BC och AF användes fritt
-------------------------------------------
POP AF      : Återställer registren från
POP BC      : stacken.
POP HL      :
EI          : Enable interrupt. Sätt på !!
RETI        : Återvänd dit datorn var innan
            : avbrott.
```

Fyra.

Den enklaste användningen av avbrottshanteringen är kanske att man ska trycka någon tangent för att något ska hända. I standard ABC80 finns det bara en enda tangent för en sådan här funktion d.v.s CTRL-C.

Men det är inget som hindrar att man lägger in egna specialare, t.ex. Rensa skärmen med ett lätt tryck på CTRL-L eller printernödstopp på CTRL-S. Vi kan ju räkna med omedelbar reaktion eftersom vi utnyttjar processorns avbrottsmöjligheter.

Men alla vanliga tangenter vill vi förstås ska vara som vanligt d.v.s. vi måste vid varje avbrott undersöka om vi har tryckt på CTRL-L resp. CTRL-S. Har vi det, så ska datorn utföra rensning etc. Har vi det inte, så ska datorn hoppa till original tangentbordsrutinen. (vi kan förstås skriva av rutinen men det vore ju slöseri...) Rutinen finns på adress 3:30 (798) och där finns instruktioner angående CTRL-C och repetering av tangenterna, så den är viktig!

```
DI              :
PUSH AF         :
IN (56)         : Läs in tangentvärdet (POrt 56) i A
CP 140          : Jämför med ASCII för CTRL-L (12+128)
CALLZ RENSA     : Om rätt tangent, hoppa till RENSA
CP 193          : Jämför med ASCII för CTRL-S (65+128)
CALLZ NÖDSTOPP  : Om rätt tangent, hoppa till NÖDSTOPP
POP AF          :
JP 3:30         : Hoppa till original tangentbordsrutin
                : Där finns ju det vanliga EI och RETI,
                : så det behövs inte här!
```

Fem.

Här är ett exempel på ett specialfall: Antag att vi vill ha en funktion som gör att vi kan stryka raden i ett BASIC-program. Vi ska bara behöva trycka CTRL-D, så ska följande fråga dyka upp på skärmen: MELLAN VILKA RADER? Rad1-Rad2

Sedan ska man kunna mata in önskat radnummer, trycka RETURN och därefter ska rutinen plocka bort raderna i fråga.

Detta skulle alltså kunna vara en lång subrutin som blev anropad varje gång vi tryckte CTRL-D. Här stöter man emmelertid på ett problem. För att kunna mata in siffrorna kan ju nödvändigtvis inte datorn vara kvar i avbrottssubrutinen, eftersom vi av andra skäl har stängt av INT med instruktionen DI. Problemet löses genom att vi inte kopplar samman själva avbrottsrutinen med rutinen som tar bort raderna (DELETE). Vi gör RETI redan innan datorn går in i vår lilla rutin. Det går till så att vi lagrar en annan återhoppsadress på stacken, nämligen adressen till DELETE. Resultatet blir att datorn inte hoppar tillbaka där den var innan avbrottet, utan istället till vår lilla rutin. Därefter kan vi, om vi vill, låta datorn hoppa tillbaka på riktigt med ett enkelt RET (201 decimalt) den här gången.

```
                DI              243          :Stäng av fortsatta avbrott
                PUSH AF         245          :Spara AF
                IN(56)          219,56       :Läs in tangentvärde i A
                CP 132          254,132      :Jämför med CTRL-D
                JRZ SPECIAL     40,4         :Hoppa om rätt tangent
                POP AF          241          :Återställ AF och...
                JP 3:30         195,30,3     :...hoppa till riktiga tangent-
:               ------------------------      bordsrutinen. (EI finns där!)
                POP AF          241          :Återställ AF
                PUSH HL         229          :Spara HL
                LD HL, DELETE   33,n1,n2     :HL=adressen till rutinen (DELETE)
                EX (SP)-HL      227          :Lägg adr. på stacken återst. HL
                EI              251          :Koppla på!!
                RETI            237,77       :Hoppa till, just det, DELETE!
                ------------------------
Adress till     Här kan man nu lägga in
DELETE-rutinen  sin lilla assemblersnutt.
```

Värt att veta är att när datorn väl har kommit hit så kan nya avbrott få komma, utan att det inträffar stackfel, eftersom register AF & HL är helt återställda innan RETI. Således måste DELETE-rutinen innehålla PUSH AF och PUSH HL som vanligt om man använder dessa register. Efter alla eventuella POP så avslutas sedan rutinen DELETE med RET, så kommer datorn att återvända på "riktigt".

Nu ska vi titta på programmering av PIO som också kan vara intressant i sammanhanget.

Som jag redan nämnt förut så har man i ABC80 programmerat PIO för en viss funktion angående tangentbordet. Det är bara vid användningen av kassettbandspelaren som PIO tillfälligt programmeras om för en annan funktion. Ibland har man dock nytta av att programmera PIO på ett tredje sätt! Först en erinran om vad om i detalj händer då man trycker på en tangent: (Antag att ABC80 precis är påslagen). Tangentbordselektroniken ettställer den högsta biten (bit 7) för att markera att en tangent nedtrycks. (Detta innebär att INP(56) får värden större än 127) De andra sju bitarna innehåller tangentens ASCII-värde. Nu vet vi emellertid även att PIO då är programmerad på ett sådant sätt att när den upptäcker attt bit 7 ettställs på port A (se fig. 1), ska den skicka ut en INT-signal till processorn och samtidigt lägga ut en byte på databussen, den byte som processorn använder för att bilda pekadressen. PIO blir alltså den yttre enheten som jag talade om förut. I ABC80 är PIO programmerad för att skicka ut 52 på databussen då det gäller tangentbordet, och 54 då det gäller kasettbandspelaren. Denna lilla skillnad är egentligen den enda som gör att processorn kan avgöra om det kommer avbrottsbegäran från tangentbord eller kasetttbnandspelare, INT-signal kommer i båda fallen.

Hur går då programmeringen till rent praktiskt?

ABC-klubbens rapport nr. 1 beskriver detta, men här kommer det igen:

I ABC80 sker all PIO-programmering via utportar, i bruksanvisningen för ABC80 ser man att PIO-port A kan läsas med INP(56); i BASIC, och PIO-port B kan läsas med INP(58).

Båda dessa har motsvarande kontrollportar: 57 resp. 59, och det är via dessa portar man programmerar PIO.

Vi skriver helt enkelt OUT 57, data, 57, data etc för all data som rör PIO-port A (tangentbordet) och OUT 59,data, 59, data etc för allt som rör PIO-port B (kasettbandspelare/V24).

Nu ska vi se vilken typ av data vi ska skicka.

I princip kan vi säga att PIO har 4 instruktioner:

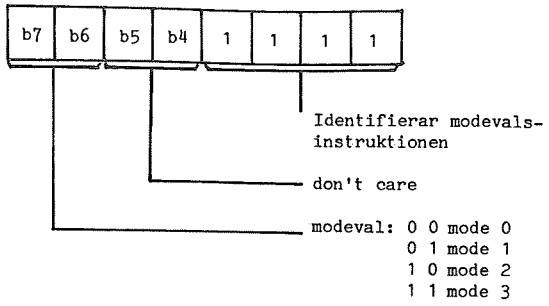
1. Modeval - bestämmer PIO-ns arbetsätt angående dataflödesriktning.

2. Interruptkontroll - bestämmer de vilkor som ska vara uppfyllda för att PIO ska generera INT-signal.

3. Interruptvektor - den byte som PIO ska lägga på databussen i samband med avbrott.

4. Interruptbortkoppling - bestämmer om avbrottshanteringen ska vara tillkopplad eller bortkopplad.

Alla instruktioner är egentligen en byte med 8 bitar. När vi programmerar PIO med OUT 57 resp OUT 59 måste vi alltså tänka i bitar och bitvikter:



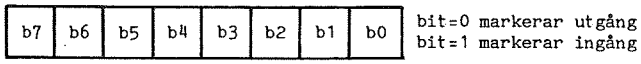
Modevalsinstruktion väljer en av PIO:s fyra arbetsmoder:

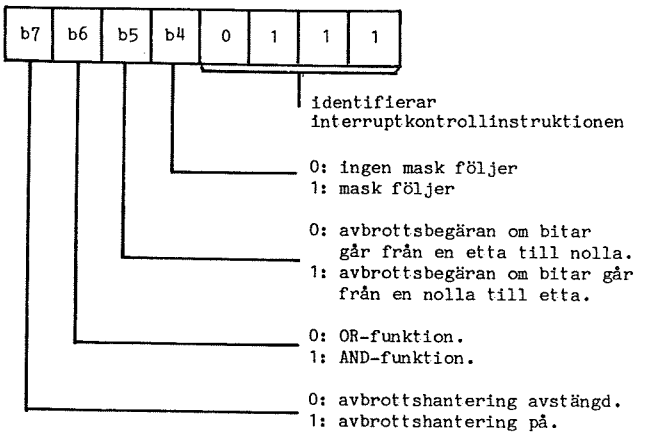
Mode 0.
Innebär att både port A och port B fungerar som utportar. Data kan alltså bara skickas från portarna.

Mode 1.
Innebär att både port A och port B fungerar som inportar. Data kan bara skickas till portarna.

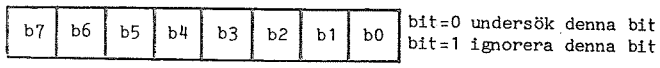
Mode 2.
Innebär att port A blir dubbelrikta d.v.s. antingen utport eller inport. Det blir dock på port B:s bekostnad som inte kan användas då.

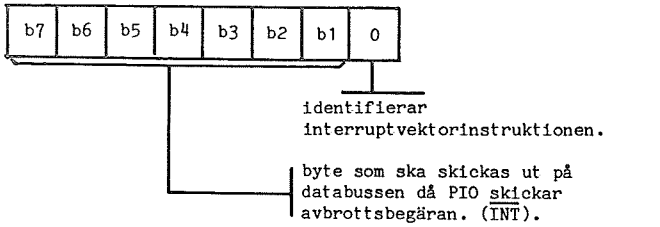
Mode 3.
Innebär att både port A och port B kan användas som in- och utgångar samtidigt. Till och med vissa bitar i en port kan vara utgångar medan de andra bitarna i samma port fungerar som ingångar. Vilka bitar som ska vara in- eller utgångar måste man alltså också ange varje gång man begär Mode 3. Efter att man lagt ut modevalsinstruktionen enligt ovan, lägger man därför även ut en ny byte som bildas så här:



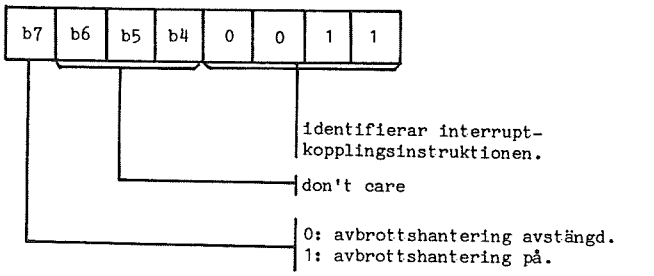


Interruptkontrollinstruktionen används bara då PIO arbetar i Mode 3. Principen är att man låter PIO bevaka en eller flera bitar i port A eller port B, och vid förändrad bitnivå skicka en avbrottsbegäran. Med bit 6 kan man bestämma om alla de valda bitarna måste ändras för att få iväg avbrottsbegäran (=AND), eller om det räcker med en eller flera av de valda bitarna (=OR). Med bit 5 kan bestämmas om PIO ska reagera för bitnivåändring till nolla respektive till etta. Med bit 4 anger man om man vill sända med en bitmask efter instruktionen. Bitmasken talar om för PIO vilka bitar den ska undersöka:





(Här kan man se orsaken till att processorns pekadress vid avbrottshantering nödvändigtvis alltid måste vara jämn!)



Nu ser vi hur PIO blir programmerad då vi tillkopplar ABC80:

```
OUT 59,207   (11001111)  :Sätter PIO i Mode 3.
OUT 59,135   (10000111)  :Definierar in/ut-bitar i port B.
OUT 57,52    (00110100)  :Interruptvektor vid avbrott
                         :från port A.
OUT 57,183   (10110111)  :Interruptkontroll från port A,
                         :INT om bitar höga, mask följer.
OUT 57,127   (01111111)  :Bitmask, undersök bit 7 i port A
OUT 57,255   (11111111)  :Sätter PIO i Mode 3.
OUT 57,255   (11111111)  :Definierar in/ut-bitar i port A.
```

Understrukna bitar är de bitar som användes för instruktionsidentifiering. Märk väl att ordningsföljden för instruktionerna spelar ingen roll.

Observera att det spelar ingen roll i vilken ordning instruktionerna till PIO kommer. Den identifierar ju instruktionerna i alla fall enligt ovan.

Märk också att t.ex. OUT 59,207 och OUT 59,255 är precis samma sak. Det beror på DON'T CARE-bitarna i instruktionerna.

Vad kan man nu ha för nytta av allt det här?

Vi ska titta på ett vanligt problem och försöka lösa det. Ofta när man gör spel och annat i BASIC behöver man känna av tangentbordet för att kunna styra spelet. (Såvida man ej använder joy-stick!) Då brukar man använda A=INP(56) för att känna av tangentnedtryck. Gäller det ett spel typ INVADER kanske spelaren har tryckt på X (fire) och då ska kanonen skjuta...

Då träffar man emellertid på problemet att vi måste ha satsen INP(56) på flera ställen i programmet för att få omedelbar reaktion. Helst vill vi ju skjuta direkt vid tangentnedtrycket. Är man mycket noga när man skriver programmet och strukturera det väl brukar detta dock lösa sig, men här ska vi utnyttja ABC80:s avbrottshantering

Principen blir att BASIC-programmet skrivs som vanligt men utan rutin för tangentavläsning. Om vi tänker på INVADER igen, så ska alltså alla rymdfarkoster röra sig, men man har ingen möjlighet att skjuta ner dem. Med diverse konstknep ska vi göra så att datorn hoppar till en speciell programrad då vi trycker ned en tangent. Raden kan t.ex. ligga allra sist i programmet och där finns sedan instruktioner för skott. Används flera tangenter i spelet måste vi ochså ha lite IF INP(56)= THEN-satser för att kunna skilja dem åt. Det fiffiga ligger i att hoppet till dessa slutrader kommer alltid att ske direkt oberoende av var i programmet datorn är någonstans. Tangentnedtrycket bryter allting! Dock måste vi se till att vi enkelt kan få tillbaka den vanliga tangentfunktionen. Vi kan ju i programmet ha INPUT-satser, och då vill vi knappast att datorn hela tiden hoppar iväg till slutraden. Precis samma måste det bli när programmet är slut eller vi trycker CTRL-C. I annat fall kommer programmet att starta och hoppa till slutraden varje gång vi trycker ned tangenter...

För att göra det hela riktigt användarvänligt ska man kunna aktivera vårt special-avbrottshantering med P%=CALL(anropsadress, radnummer) där anropsadressen är där rutinen ligger lagrad i minnet och radnummer är det radnummer i BASIC som datorn ska hoppa till vid tangentnedtryckning För att inaktivera avbrottshanteringen ska man bara behöva skriva P%=CALL(anropsadress,0). Detta ska automatiskt ske vid CTRL-C. Vi kan tydligen jämföra med användningen av ONERRORGOTO.

PROGRAM: BASICINTERRUPT

Rutinen kan placeras var som helst i minnet men i op-koden har jag räknat med A=65408 d.v.s. POKE-arean överst i minnet.

```
A     DI              243              :
      PUSH AF         245              :
      PUSH HL         229              :
      IN A,(56)       219,56           :Läs in tangentvärde och
      CP 131          254,131          :undersök om man har tryckt
      JRNZ            32,6             :CTRL-C. Om inte, hoppa
      LD HL,0:0       33,0,0           :Lägg in radnummer 0
      LD (A+47),HL    34,175,255       :i A+47
A+15  LD HL,(A+47)    42,175,255       :Undersök om A+47
      LD A,H          124              :innehåller
      OR L            181              :radnummer 0 ...
      POP HL          225              :
      JRNZ A+27       32,4             :... Om inte, hoppa
      POP AF          241              :
      JP 3:30         195,30,3         :hoppa till vanliga tangentrut.
A+27  POP AF          241              :
      PUSH HL         229              :Lägg upp
      LD HL,A+36      33,164,255       :ny återhoppsadress (=A+36)
      EX (SP)-HL      227              :på stacken för att
      EI              251              :
      RETI            237,77           :lura RETI
A+36  LD DE,(A+47)    237,91,175,255   :Hämta radnummer i A+47
      CAL 15:57       205,57,15        :Anropa rutin för att leta upp
      EX DE-HL        235              :adressen där önskad rad börjar
      JP 13:103       195,103,13       :Hoppa till RUN och exekvera fr.o.m.
                                       :den funna adressen
A+47  DEFW Radnummer                   :

A+49  LD (A+47),DE    237,83,175,255   :Lagra radnummer i A+47
      PUSH AF         245              :
      PUSH BC         197              :
      PUSH HL         229              :
      LD A,E          123              :Om det gäller
      OR D            178              :radnummer = 0
      JRZ A+74        40,14            :så hoppa
      CAL 15:57       295,57,15        :Leta upp radens adress,
      JRZ A+74        40,9             :hoppa om radens finns.
      LD HL 0:0       33,0,0           :Lägg in radnummer  = 0
      LD (A+47),HL    34,175,255       :i A+47.
      JP 7:5          195,5,7          :Avbryt och skriv 'ERR 6'
A+74  LD A,n1         62,n1            :Ändra processorns (n,se nedan)
      LD I,A          237,71           :I-register för
      POP HL          225              :
      POP BC          193              :
      POP AF          241              :att initiera rutinen
      RET             201              :
A+82  Första lediga byte..
      -------
```

A+49 är alltså anropsadress för aktivering.

Adressen till RUN (13:103) används på checksumma 10042 PÅ checksumma 11273 ska adressen istället vara 13:105.

Innan vi kör rutinen måste vi också lagra de två hoppadresserna som pekar på tangentbordsrutinen och kasettbandspelare. Normalt är dessa adresser 3:30 för tangentbord och 5:148 för kasettbandspelare, och de finns lagrade på adress 0:52-0:53 resp. 0:54-0:55.

Nu vill vi ju istället för 3:30 ha adressen till vår rutin, alltså 255:128 (med A=65408!)

Samtidigt vet vi att processorn letar efter dessa adresser med hjälp av en pekadress som bildas med I-registret & databussen.

Om vi inte, för enkelhetens skull, vill programmera om PIO så finns det bara I=registret kvar att ändra på. (På databussen kommer det alltså att finnas 52 vid varje avbrott från tangentbordet.)

Om vi sätter n1=242 ovan, och gör följande POKE-satser bör allt vara klart för test!

```
POKE 242*256+52,128,255    : REM Hoppadress tangentbord
POKE 242*256+54,148,5      : REM Hoppadress kasettbandspelare

10 REM Testprogram
20 PRINT CHR$(12)
30 P=CALL(65408+49,70) : REM Aktivera, hopp till rad 70
40 REM Oändlig loop
50 PRINT "Mary had a little lamb-tryck tangent"
60 goto 50
70 REM Hopp hit vid interrupt!
80 P=CALL(65408+49,0) : REM Inaktivering
90 END
```

I rutinen finns en liten säkerhetsdetalj inbyggd. Skulle anropet på rad 30 innehålla ett radnummer som inte existerar i programmet så stoppar datorn och skriver ERR 6 LINE 30.

Det är ju lätt hänt att den vetgirige lägger till en rad i början och sedan skriver REN(umber).

Riktigt bra är det inte att poka in de båda hoppadresserna på 242:52-242:55. Där ligger de helt oskyddade för väder och vind höll jag på att säga. Alltså bestämmer vi oss för en omprogrammering av PIO. Det stoppar vi in i vår rutin. Komplettering börjar från A+74:

```
A+74 LD A,255       62,255       :I-registret laddas
     LD I,A         237,71       :med 255!
     LD A,218       62,218       :PIO port A programmeras
     OUT 57         211,57       :med interruptvektor = A+90
     LD A,220       62,220       :PIO-port B programmeras
     OUT 59         211,59       :med interruptvektor = A+92
     POP HL         225          :
     POP BC         241          :
     POP AF         201          :
A+90 DEFW 255:128   128,255      :Hoppadress tangentbord
A+92 DEFW 5:148     148,5        :Hoppadress kasettbandspelare
 94 Första lediga byte
```

Såja! Nu ryms alltihop i pokearean. Allt ska fortfarande fungera vid ny provkörning!

Till sist: Hur vore det med avbrottshantering på PIO-port B?

I ABC-bladet nr 2, 1984 beskrev Kalle Lindström en rutin "LEAVE" på ABC800. Det handlade om att datorn själv sa till att det var sent och dags att gå och lägga sig vid ett förinställt ockslag. It's time to leave now!! Det var en kul id tyckte jag och började fundera på hur man skulle kunna överföra detta till ABC80. Det gäller bara att på något vis få datorn att regelbundet läsa av klockan och jämföra med det förinställda klockslaget. Med lite avbrott då och då vore det lätt. Detta avbrott kan antingen genereras från tangentbordet eller V24 via en yttre oscillator. Det sistnämda alternativet är förstås bäst. Om ingen hamrar på tangentbordet just vid den aktuella tiden kan ingen jämförelse ske och därmed kommer klockan att gå förbi...

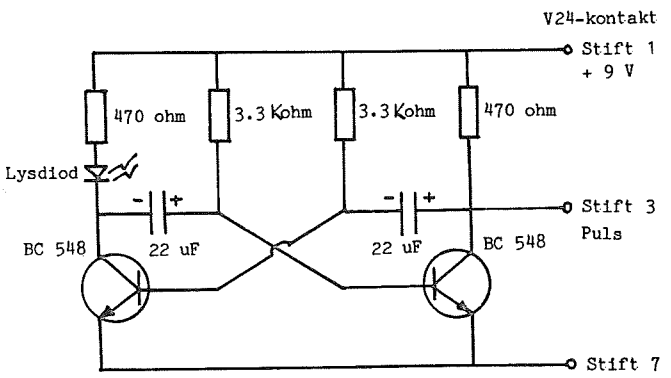
Nu tänker jag inte beskriva vidare om just detta utan istället något snarlikt:

Vi vill att datorn kontinuerligt, oberoende vad den gör, skriver ut text på över sta raden.

Det kan gälla aktuell tid, BASIC-programmets aktuella längd, antal öppna eller bara "SUNE-DATA AB".

Jag har knåpat ihop en avbrottsrutin som kan sköta det sistnämnda. Förutsättningen är förstås att en oscillator finns ansluten på V24. En vanlig blinkers på 10-20 Hz duger bra. Ju snabbare blinker desto oftare uppdateras textraden. Vid kHz blir dock datorn märkbart förslöad.

Strömförsörjningen tas från V24:



(Den som vill kan istället ansluta en tryckströmbrytare mellan stift 1 och stift 3 och sitta och trycka...)

Avbrottsrutinen fungerar så att den växlar mellan två textsträngar i minnet och skriver ut dem varannan gång i takt med avbrotten.

Om man fyller båda textsträngarna med precis samma innehåll sker inget speciellt jämfört med om det bara fanns en sträng. Fyller man den ena med mellanslag däremot kommer texten att blinka! Fritt fram för fantasin alltså. För att kunna finjustera denna bildväxlingsfrekvens utan att byta komponenter i oscillatorn är rutinen även begåvad med en räknare (frekvensdelare) som "släpper igenom" var 5:te puls t.ex.

Man kan dela frekvensen med en valfri faktor mellan 1 och 255.

Program SYSTEMLINE.

Rutinen är relokerbar men för teständamål sättes A=49152 d.v.s. BOFA på 16K-maskin. Glöm ej att höja BOFA.

```
A    DI              243          :
     PUSH AF         245          :
     PUSH BC         197          :
     PUSH DE         213          :
     PUSH HL         229          :
     LD HL,(253:243) 42,243,253   :Spara markörposition
     PUSH HL         229          :på stacken
     LD A,(255:254)  58,254,255   :Frekvensdelare:
     CP "faktor"     254 "faktor" :
     JRZ A+22        40,6         :En räknare (255:254)
     INC A           60           :inkrementeras och
     LD (255:254),A  50,254,255   :kollas med "faktorn"
     JR A+53         24,31        :Stämmer ej - avsluta!!
A+22 XOR A           175          :Nollställ frekvens-
     LD (255:254),A  50,254,255   :delarräknaren.
     LD A,(255:255)  58,255,255   :Kolla om sträng 0
     CP 0            254,0        :eller 1 ska skrivas.
     JRZ A+39        40,6         :
     LD HL,255:128   33,128,255   :Adress till sträng 1.
     XOR A           175          :Skifta sträng
     JR A+44         24,5         :
A+39 LD HL,255:172   33,172,255   :Adress till sträng 0.
     LD A,1          62,1         :Skifta sträng.
A+44 LD (255,255),A  50,255,255   :
     LD BC,0:44      1,44,0       :
     CAL 0:11        205,11,0     :Skriv ut sträng.
A+53 POP HL          225          :Återställ markör-
     LD (253:243),HL 34,243,253   :position från stacken.
     POP HL          225          :
     POP DE          209          :
     POP BC          193          :
     POP AF          241          :
     EI              251          :
     RETI            237,77       :
A+64 PUSH AF         245          :Initiera
     LD A,192        62,192       :processorns
     LD I,A          237,71       :I-register. = 192
     LD A,255        62,255       :
     OUT 59          211,59       :PIO i bitmode
     LD A,135        62,135       :
     OUT 59          211,59       :Def. in/ut-bitar
     LD A,98         62,98        :
     OUT 59          211,59       :Hoppvektor PIO-
     LD A,151        62,151       :port B=A+98
     OUT 59          211,59       :Interruptkontroll,
     LD A,254        62,254       :mask följer
     OUT 59          211,59       :Mask
     LD A,96         62,96
     OUT 57          211,57       :Hoppvektor PIO-
     POP AF          211,57       :port A=A+96
     RET             201
     NOP             0            :Förhindrar udda adress
A+96 DEFW 3:30       30,3         :Hoppadress tangentbord.
A+98 DEFW 192:0      0,192        :Hoppadress V24
A?100 Första lediga byte.
```

Innan test bör nämnas att textsträngarna finns i pokearean liksom två variabler 255:254-255:255. Textsträngarnas längd är vald med avsikt på att en rad + 4 tecken för cursorstyrning ska skrivas. Om de fyra första ASCII-teknen i strängarna är 27,61,32,32 så kommer texten att styras till rad 0 på skärmen. Saknas dessa kommer skärmen att rulla.

```
 10 REM Testprogram
 20 DIM A$(2)=44, A$=88
 30 A$(1)=CUR(0,0)+"ABC80"+SPACE$(35)
 40 A$(2)=CUR(0,0)+SPACE$(40)
 50 A$=A$(1)+A$(2)
 60 FOR I=1 to 88
 70 POKE 65408+I-1, ASC(MID$(A$,I,1))
 80 NEXT I
 90 POKE 49152+13,10 : REM "Faktor = 10"
 95 POKE 65534,0 : REM Nollställer räknaren
100 P=CALL(49152+64) : REM Initierar avbrottsrutinen
110 END
```

Nu kommer ABC80 blinka överst på skärmen hela tiden. Prova skriv LIST och RUN för att konstatera att så är fallet.

Observera att så länge PIO är programmerad på detta sätt fungerar inte kasettbandspelaren.

Johan Struwe <3338>
Smedjebacken
0240-76784

Höjoplöselig Grafik Med ABC-80

Höjoplöselig grafik til ABC-80 har länge väret et önske for mange ABC-80 ejere. Dette har dog ikke väret mulig, medmindre man anvender andre processorer.

Jeg har igennem längere tid konstateret, at der findes mange grafikprocessorer, som anvendes i en hel del udstyr, og min interesse for at udnytte dette udstyr til ABC-80 satte en hel del igang.

En af mine venner, Finn Spile, havde igennem längere tid gjort brug af höjoplöselig grafik til sin ABC-800, så hvorfor ikke lave dette til ABC-80.

I Tyskland findes en hel del print til selvbygger computer systemer, og i mange tilfälde til en meget rimelig pris, derfor var det naturligt at kikke på nogle färdige lösninger.

Hos:
ELEKTRONIKLADEN MIKROCOMPUTER
GIESLER & DANNE GMBH & CO. KG
W. MELLIES STRASSE 88
4930 DETMOLD 18
WESTTYSKLAND

kunne man köbe et grafikkort GDP64K, som kunne anvendes til en selvbygger computer, dette printkort bestod af en grafikprocessor EF9366 samt styrelogik til opdeling af 4 billedsider på hver 16Kbyte, ialt 64K billedram. Printet er dobbelsidigt 100 x 105 mm. Prisen for hele herligheden er ca. 350 DM, hvilket må siges at väre meget rimeligt, idet grafikprocessoren alene koster fra 700-900 dkr.

Printets störrelse er netop således, at det passer til et Europakort 100 x 160 mm. På Europakortet kan man montere grafikprintet og så er der plads til at montere interface til ABC-80 og montere et 64-pol stik til ABC-bussen. Hele herligheden kan herefter monteres i en ekspansionsboks og dertil montere et BNC-stik til udvendig monitor.

ABC-80 tv-skärmen anvendes til afläsning af beregninger, indtastninger eller andre rutiner, mens man kan kikke på sine höjoplöselige billeder på en egnet monitor.

Oplösningen er 512 x 256 punkter på hver af de 4-billedsider, og systemet er indrettet så man selv kan välge hvilken side man vil läse eller skrive på uafhängigt af hinanden.

Listningen af inputrutinen er lavet som et stykke värktöj, hvor samtlige registre og data kan afläses, således at den enkelte kommandos registre kan vises på ABC-80 skärmen og handlingen af instruktionerne på grafikskärmen. Dette giver en forståelse af hvordan grafikprocessoren arbejder. Dermed skal ikke forstås, at den er vanskelig at arbejde med - tvärtimod. Der gälder dog nogle regler, som skal overholdes, derfor kan det tilrådes at studere instruktionssättet over EF9366 fra Thomson Semiconducters.

Interface mellem ABC-80 og GDP 64K grafikkortet, skulle ikke volde vanskeligheder. OUT 1,X välger kortnr, hvor X er kortnr. som binärt kan strappes efter eget önske, i mit tilfälde er kortnr. 7 valgt. Det vil sige at ben 14,12 og 10 er HIGH og ben 6,4 og 2 er lagt til stel.

OUT 2,Y sender adressen til grafikprocessorens adresselatch (Y=adressen)

OUT 0,Z sender data til grafikprocessoren (Z=data).

For at få adgang til grafikkortets registre, skal 2 porte sättes, dette er adresse 96, hvor sidenr. välges, og adresse 112, hvor adresseregistrene i til EF9366 välges.

Side nr. 0 = OUT2,96:OUT0,0 (eller OUT2,96,0,0) Side nr. välges i det binäre tal som sendes ud på OUT0 Bit nr. 4+5 giver de 4 läsesider, og bit 6+7 giver de 4 skrivesider. Kombinationen af dette möster giver muligheder for både at läse og skrive på samme side eller forskellige sider, en meget interessant mulighed, idet opbygningen af et billede kan ske, mens man kikker på et andet billed.

Adresseregistrene fra 0 til 15 välges med OUT 2,112+0 til OUT2,112+15

De mange muligeder for tegnopbygning og tegninger kan bedst afpröves med forsög. I EF9366 processoren er ilagt tegnsät, og dette tegnsät kan forstörres, formindskes eller tiltes efter önske.

Jeg har igennem et stykke tid arbejdet med en plotter til min ABC-80, og mine tegningers koordinater i lagt i datasätninger, disse datasätninger kan også behandles af grafikprocessoren, således at diagrammer og tegninger kan iagttages på skärmen, för de plottes ud.

Testprogram 1 er en kontrol på, at systemet virker, hvis det er samlet rigtig. Programmet printer ABC-80 midt på skärmen.

Testprogram 2 viser ABC-80 i stor skrift.

For dem der vil vide mere om dette

grafikprints muligheder, kan jeg anbefale at købe bogen: "MIKROCOMPUTER SELBSTGEBAUT UND PROGRAMMIERT" af Klein fra forlag Franzis.

Idet jeg håber, at andre vil opleve ABC-80 muligheder som terminal for en höjoplöselig enhed, kan jeg kun sige, at vejen ligger åben for endnu højere oplösning f.eks. 1024 x 1024 punkter samt farver på tilsluttet monitor, men mere om dette en anden gang.

God fornöjelse!

PS

Det er mit håb, at flere vil konstatere, at ABC-80 skärmen som Terminal til en höjoplöselig monitor er en glimrende lösning, idet beregninger og indtastninger samt aflæsning af grafikprocessorens registre kan :e på ABC-80 skärmen og tegningerne .an vises på monitoren.

Grafikprocessoren er meget nem at arbejde med, så kom blot igang, og skulle der opstå problemmer eller spörgsmål så skriv til mig.

Flemming Baagöe
Söndergade 16
DK-4130 Viby Sj DANMARK

```
 0 REM TEST1 - VERS. 1.0
 20 OUT 1,7 : REM KORTVALG
 30 REM SIDE 0,SLET SKÄRM
 40 OUT 2,96,0,0 : GOSUB 140 : OUT 2,112
    ,0,6
 50 REM PEN NED
 60 GOSUB 140 : OUT 2,113,0,3 : GOSUB 14
    0
 70 REM ORIGO X,Y ADR
 80 OUT 2,120,0,0 : GOSUB 140 : OUT 2,12
    1,0,240 : GOSUB 140
 90 OUT 2,122,0,0 : GOSUB 140 : OUT 2,12
    3,0,127 : GOSUB 140
100 REM SKRIV ABC-80
110 OUT 2,112,0,65,0,66,0,67,0,45,0,56,0,
    48
120 END
130 REM READY FOR NY KOMMANDO
140 OUT 2,112
150 IF NOT INP(0) AND 4 THEN 150
160 RETURN
```

```
 10 REM TEST2 - VERS. 1.0
 20 OUT 1,7 : REM KORTVALG
 30 REM SIDE 0,SLET SKÄRM
 40 OUT 2,96,0,0 : GOSUB 180 : OUT 2,112
    ,0,6
 50 REM PEN NED
 60 OUT 2,113,0,3 : GOSUB 180
 70 REM CSIZE
 80 OUT 2,115,0,255 : GOSUB 180
 90 REM ORIGO X,Y ADR
100 OUT 2,120,0,0 : GOSUB 180 : OUT 2,12
    1,0,0 : GOSUB 180
110 OUT 2,122,0,0 : GOSUB 180 : OUT 2,12
    3,0,50 : GOSUB 180
120 REM SKRIV ABC80
130 OUT 2,112,0,65 : GOSUB 180 : OUT 0,6
    6 : GOSUB 180
140 OUT 0,67 : GOSUB 180 : OUT 0,45
150 GOSUB 180 : OUT 0,56 : GOSUB 180 : O
    UT 0,48
160 END
170 REM READY FOR NY KOMMANDO
180 OUT 2,112
190 IF NOT INP(0) AND 4 THEN 190
200 RETURN
```

```
 10 REM GDPINPUT - VERS. 1.0
 20 REM FLEMMING BAAGÖE - 17/8-85.
 30 REM 80 TEGN ANVENDES
 40 ; INP(4%);CHR$(12%)
 50 ; CUR(0%,5%)'SIDE 0';CUR(0%,35%)'GDP,
    -REGISTRE'
 60 OUT 1%,7% : REM KORTVALG
 70 OUT 2,96 : REM SIDE VALG
 80 OUT 0,0 : REM SIDE 0
 90 GOSUB 230
100 REM UDSKRIFT
110 F$=STRING$(80%,45%) : ; CUR(1%,0%)F$
    ;CUR(18%,0%)F$
120 ; CUR(20%,18%)'ADRESSE: '
130 ; CUR(20%,35%)'DATA: '
140 ; CUR(20%,27%); : ; SPACE$(3%) : ; C
    UR(20%,27%); : INPUT B$
150 IF B$='' THEN 140
160 ; CUR(20%,41%); : ; SPACE$(3%) : ; C
    UR(20%,41%); : INPUT C$
170 IF C$='' THEN 160
180 A=VAL(C$) : GOSUB 380 : ; CUR(20%,55
    %)'BIN: 'D$
190 OUT 2%,VAL(B$)+112% : REM ADRESSE SÄ
    TTES
200 OUT 0%,VAL(C$) : REM DATA SENDES
210 OUT 1%,7%
220 GOSUB 230 : GOTO 140
230 RESTORE 480
240 REM REGISTRE LÄSES
250 FOR I=0 TO 15
260 OUT 2,I+112 : A=INP(0)
270 GOSUB 380 : GOSUB 470 : A1$=NUM$(A)
280 IF I=4 OR I=6 OR I=14 THEN 290 ELSE
    300
290 D$='-' : A1$=' -'
300 ; CUR(I+2,5)'ADR: 'I;TAB(18%)'INHOLD
    : 'A1$;TAB(35%)'BIN: 'D$;TAB(55%)E$
310 GOTO 320
320 NEXT I
330 REM BIN SOM IKKE BENYTTES
340 ; CUR(3%,40%)'X';CUR(4%,40%)'XXXX';C
    UR(10%,40%)'XXXX'
350 ; CUR(12%,40%)'XXXX';CUR(14%,40%)'X'
360 RETURN
370 REM DEC TIL BIN
380 C=128
390 D$=''
400 FOR T=7 TO 0 STEP -1
410 IF A AND C THEN D1$='1' ELSE D1$='0'
420 IF T=4 THEN D1$=D1$+' '
430 D$=D$+D1$
440 C=C/2 : NEXT T
450 RETURN
460 REM BETEGNELSER
470 ONERRORGOTO 510 : READ E$
480 DATA 'STATUS/CMD','CTRL 1','CTRL 2',
    CSIZE,RESERVED,DELTAX
490 DATA RESERVED,DELTAY,'X MSB','X LSB'
500 DATA 'Y MSB','Y LSB',XLP,YLP,RESERVE
    D,STATUS,,
510 RETURN
```


TABLE 1 – REGISTER ADDRESS

ADDRESS REGISTER					REGISTER FUNCTIONS		Number of bits
Binary A3	A2	A1	A0	Hexa	Read R/W = 1	Write R/W = 0	
0	0	0	0	0	STATUS	CMD	8
0	0	0	1	1	CTRL 1 (Write control and interrupt control)		7
0	0	1	0	2	CTRL 2 (Vector and symbol type control)		4
0	0	1	1	3	CSIZE (Character size)		8
0	1	0	0	4	Reserved		—
0	1	0	1	5	DELTAX		8
0	1	1	0	6	Reserved		—
0	1	1	1	7	DELTAY		8
1	0	0	0	8	X MSB		4
1	0	0	1	9	X LSB		8
1	0	1	0	A	Y MSB		4
1	0	1	1	B	Y LSB		8
1	1	0	0	C	XLP (Light-pen)	Reserved	7
1	1	0	1	D	YLP (Light-pen)	Reserved	8
1	1	1	0	E	Reserved		—
1	1	1	1	F	STATUS	Reserved	8

Reserved : These addresses are reserved for future versions of the circuit. In read mode, output buffers D0-D7 force a high state on the data bus.

TABLE 2 – COMMAND REGISTER

b7 b6 b5 b4 / b3 b2 b1 b0	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0 0	Set bit 1 of CTRL1 Pen selection															
0 0 0 1 1	Clear bit 1 of CTRL1 Eraser selection															
0 0 1 0 2	Set bit 0 of CTRL1 Pen/Eraser down* selection															
0 0 1 1 3	Clear bit 0 of CTRL1 Pen/Eraser up selection*															
0 1 0 0 4	Clear screen															
0 1 0 1 5	X and Y registers reset to 0															
0 1 1 0 6	X and Y resets to 0 and clear screen															
0 1 1 1 7	Clear screen, set CSIZE to code mentioned in the next byte. All other registers reset to 0 except XLP, YLP.															
1 0 0 0 8	Light pen white flag* (WHITE for logic one)															
1 0 0 1 9	Light pen white flag*															
1 0 1 0 A	3 × 8 block drawing *size according to CSIZE															
1 0 1 1 B	4 × 4 block drawing *size according to CSIZE															
1 1 0 0 C	Screen blanking* Pen or Eraser up according to CTRL1															
1 1 0 1 D	X register reset to 0															
1 1 1 0 E	Y register reset to 0															
1 1 1 1 F	Direct image memory data request* (one read or write cycle)															

Vector generation (for b2, b1, b0 see small vector definition)

Small direction vectors (for b2, b1, b0 see small vector definition)

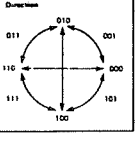
SMALL VECTOR DEFINITION :

b7	b6	b5	b4	b3	b2	b1	b0
1	ΔX1			ΔY1		D-direction	

D-direction

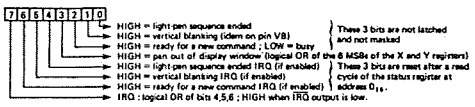
ΔX or ΔY	Vector length
0 0	0 step
0 1	1 step
1 0	2 steps
1 1	3 steps

Direction

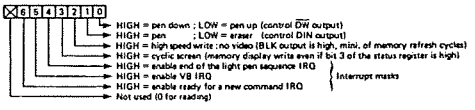


OTHER REGISTERS

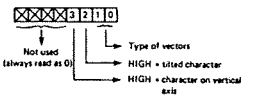
STATUS REGISTER (Read only)



CONTROL REGISTER 1 (Read/Write)

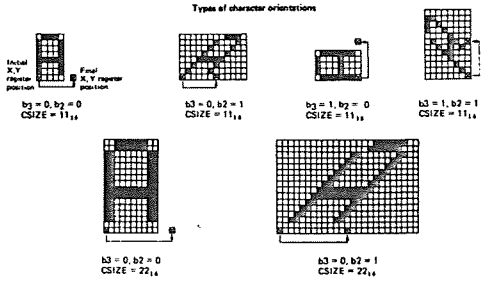


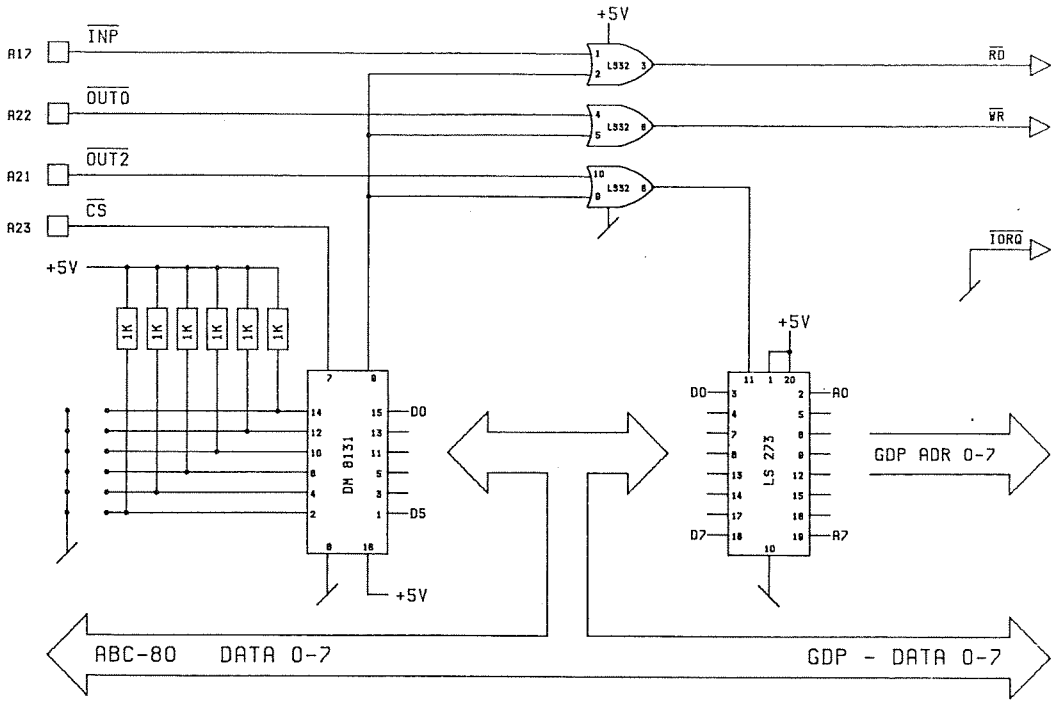
CONTROL REGISTER 2 (Read/Write)



b1	b0	Type of vectors
0	0	——— continuous
0	1	------ dotted 2 dots on, 2 dots off
1	0	– – – – dashed 4 dots on, 4 dots off
1	1	—·—·— dotted-dashed 10 dots on, 2 dots off, 2 dots on, 2 dots off.

Types of character orientations





ABC-80 INTERFACE TIL GDP GRAFIKKORT
17.08.85 - FB.

Flerradiga funktioner i BASIC II

Vad är flerradiga funktioner? Normalt så finns det i BASIC enradiga funktioner av typen DEF FNA(X,Y) = X + X * Y. Jag förutsätter att läsaren är bekant med dessa. Den typ av funktioner är alltså enradiga, dvs allt skrivs på en sats/rad.

Noteras kan att enradiga funktioner i BASIC I för ABC80 inte fungerar för strängfunktioner, men denna begränsning finns inte i BASIC II. Man kan alltså skriva DEF FNText$="Hejsan".

Flerradiga funktioner skriver man på flera rader eller flera satser. Egentligen är det bara det faktum att en flerradig funktion skrivs på flera rader eller satser som skiljer den från enradiga funktioner.

Man använder flerradiga funktioner som subrutiner i stället för att anropa subrutiner med GOSUB. Flerradiga funktioner är inte beroende av ett visst radnummer i anropet vilket är en stor fördel jämfört med GOSUB. Parametrar kan enkelt skickas med vid själva anropet och man kan få tillbaka ett värde. Vill man inte ha tillbaka ett värde så kan man skicka tillbaka ett blint värde, dvs 0 eller en tom sträng.

Variabler som används i en flerradig funktion kan göras lokala, dvs de kan ej påverkas av miljön utanför funktionen. De kan inte heller påverka variabler utför funktionen.

```
Exempel:
10 Tal=10
20 ;"Tal som global variabel" Tal
30 Dummy=FNFunk(5)
100 DEF FNFunk(in) LOCAL Tal
110    Tal = in
120    ;"Tal som lokal variabel" Tal
130    RETURN 3
140 FNEND
```

Resultatet blir:

```
Tal som global variabel 10
Tal som lokal variabel 5
```

Lokala variabler finns inte i den vanliga variabellistan (se Bit för Bit sid 100 ff) utan de finns på stacken bara under själva anropet. Stackén är en slags tabell av stacktyp som BASIC:en använder sig av när den arbetar.

Man kan inte hoppa in i en funktion på annat sätt än att anropa den och man kan inte hoppa ut ur funktionen på annat sätt än genom RETURN av ett värde. Detta gäller dock inte de första versionerna av BASIC II (till ABC800 M/C), men detta har spärrats i senare versioner av BASIC II eftersom sådana uthopp kan ge dyk i vissa fall.

Anrop görs på följade sätt:

```
10 Dummy=FNTest(Tal)
```

Värdet som returneras hamnar i variabeln Dummy. I detta fallet används namnet Dummy för att markera att det inte är något värde som man vill använda sig av. Normalt returnerar man då noll. Skall man jobba med värdet så kan man skriva något annat namn än Dummy.

Inparametern skall alltså finnas i Tal före anropet. Man kan också om man alltid skall anropa funktionen med ett visst värde skriva:

```
10 Resultat=FNTest(10)
```

Man kan också anropa flera funktioner på en gång enligt följande:

```
10 Dummy=FNFunk1+FNFunk2+FNFunk3
```

Dummy tilldelas summan av de av funktionerna returnerade värdena.

Det går också att anropa en funktion i en IF sats på fäljande sätt:

```
10 IF FNSant ;'Sant' ELSE ;'Falskt'
```

Detta förutsätter att funktionen returnar -1 för sant och 0 för falskt.

Alla inparametrar till en funktion är lokala, utan att man behöver deklarerar dessa som lokala. Strängar som inparametrar kan man inte förändra värdet på, utan man måste tilldela inparametern till en lokalt deklarerad sträng.

Exempel:

```
10   DEF FNTest(Tal) LOCAL I
20      I = Tal + 5
30      RETURN I
40   FNEND
```

Detta är samma sak som DEF FNTest(Tal) = Tal + 5, men skillnaden är att funktionen är skriven som en flerradig funktion.

```
10   DEF FNSträngfunktion$(In$) LOCAL

xx   Par$=80
20      Par$=In$
30      IF INSTR(1,Par$,"*") MID$(Par$,
xx      INSTR(1,Par$,"*"),1)="+"
40      RETURN Par$
50   FNEND
```

Denna funktion är i stället en sträng funktion, dvs den returnerar en sträng i stället för ett tal. I detta fallet så tilldelas Par$ värdet av In$. Eftersom man måste deklarera Par$ i LOCAL satsen så kan den bara vara 80 tecken lång. Om In$ är längre än 80 tecken så blir det fel. Egentligen måste man testa längden på In$ i funktionen innan man tilldelar Par$ In$:s värde. Bäst är nog att se till så att In$ inte är för lång innan själva anropet, eftersom man i programmet inte kan förändra maxlängen på Par$. Man kan alltså inte skriva:

```
... LOCAL Par$=Längd
```

Detta är också ett exempel på den MID$ funktion som inte finns i BASIC I. Denna MID$ funktion kan användas för att förändra tecken i en sträng.

Om man inte vill returnera något värde, utan bara använda en funktion som en subrutin så kan man göra så här:

```
10   DEF FNSkriv
20      ;"Denna funktion returnerar"
30      ;"inget värde"
40      RETURN 0
50   FNEND
```

Detta är också ett exempel på en funktion som inte har någon parameter vid anropet. Är det en strängfunktion så skriver man så här:

```
10   DEF FNSkriv$
20      ;"Denna funktion returnerar"
30      ;"inget värde"
40      RETURN ""
50   FNEND
```

Felhantering med ON ERROR GOTO är lokal i en funktion. Man kan anropa en funktion i flera nivåer och om det inte finns en ON ERROR GOTO i en funktion så fortplantar sig felet uppåt tills en ON ERROR GOTO hittas eller om igen sådan finns så stoppar programmet på översta nivån.

Exempel:

```
10 !Huvudprogram
20 !
30 Dummy=FNFunk
40 DEF FNFunk
50    Dummy=FNFunk2
60    RETURN 0
70 FNEND
80 DEF FNFunk2 LOCAL Errortrigger
90 ON ERROR GOTO 130
100      Errortrigger = 5
110      Errortrigger = Errortrigger / 0
120      ;'Fel nr' ERRCODE : STOP
130 FNEND
```

Kör man detta program så får man ett stopp i rad 110 på grund av fel 210, Felaktigt tal. Att dividera med noll går ju inte. Om rad 90 inte skulle finnas så skulle programmet stanna på rad 30 med en vanlig felutskrift. I det senare fallet så kan det vara lite svårt att fatta att felet härrör från rad 110 när man får stopp i rad 30!

Rekursiva anrop kan också användas. Vad detta är för något kan du läsa om i en artikel i ABC-Bladet nr 3, 1984 sid 10.

Bo Kullmar

Vad är klockan (3)?

Jag har just läst de två artiklarna i ABC-bladet 3-84, om hur man kan skriva ut klockan på skärmen medan man väntar på inmatning av tecken från tangentbordet. De båda programsnuttarna har, såvitt jag förstår, båda den nackdelen att klockan stannar på skärmen efter avslutad inmatning. (Rätta mig om jag har fel!)

Jag konstruerade för ett par år sedan en mer sofistikerad rutin, som kontinuerligt lägger ut klockan på skärmen, oavsett vad processorn sysslar med. Jag gör så att jag lägger in en assemblerrutin "under" programmet, och det kräver att man först lyfter upp "golvet" för program-arean, liksom COMMON-pekaren om COMMON-variabler används. (Allt enligt "Bit för bit".) Jag ändrar sedan adressen till ordinarie klockrutin (m h a en annan assemblerrutin, för om man ändrar med POKE i BASIC, går det definitivt snett) till att peka på assemblerrutinen, vilken skriver ut klockan på skärmen en gång i sekunden, och/eller hoppar till den ordinarie klockrutinen.

Tyvärr är inte assemblerprogrammering min starka sida, och rutinen i FNClock% nedan är helt hand-assemblerad. Vill du ha källkoden, kan du slå mig en signal (eller disassemblera den själv). Du som kan sånt här bätre än jag, kanske kan snygga upp rutinen, och ev göra den ännu snabbare.

Knappa in följande program (eller låt instruktionerna ingå i andra program):

```
10 ! SAVE CLOCK
20 !
30 ! Pekare till ordinarie klockrutin
   ligger i adress 65494 & 65495
40 ! Pekar denna redan nu utanför BASIC-
   ROM ?
50 IF PEEK2(65494%)<0% THEN 120
60 ! Lyft golv och COMMON-pekare 100
   bytes
70 Z%=PEEK2(65292%)+100%
80 POKE 65292%,Z%,SWAP%(Z%)
90 POKE 65328%,Z%,SWAP%(Z%)
100 !
110 CHAIN "CLOCK2"
120 END

10 ! SAVE CLOCK2
20 !
30 ! Lägg in egen rutin här för att
   ställa klockan, om du
40 ! inte har batteriklocka som i ABC806
   och DTC II
50 !
60 Rad%=0% : Kolumn%=72% ! Övre högra
   hörnet
70 IF PEEK2(65494%)>0% THEN Z%=FNClock%
   (Rad%,Kolumn%)
80 !
90 END
100!
27649 ! Blockmove-rutin från Toolbox
      (OK Örjan?)
27650 DEF FNBlockmove%(Addr1%,Addr2%,
      Count%) LOCAL Assm$=11
27700   Assm$=CHR$(1%)+CVT%$(Count%)+
        CHR$(33%)+CVT%$(Addr1%)
        +CVT%$(-20243%)+CHR$(201%)
27710   RETURN CALL(VARPTR(Assm$),Addr
        2%)
27715 FNEND
59999 !
60000 DEF FNClock%(R%,K%) LOCAL Asmadr%,
      Lowsec%,Clockptr%,Curpos%,T$=8,
      Asm$=92,Link$=10
60010   T$=RIGHT$(TIME$,12%)
60020   Asmadr%=PEEK2(65292%)-91% :
        Lowsec%=Asmadr%-1%
60030   Clockptr%=PEEK2(65494%) :
        Curpos%=30720%+80%*R%+K%+7%
60040   Asm$=CHR$(245%,197%,213%,229%,17
        %,245%,255%,26%,254%,1%,32%,43%)
60050   Asm$=Asm$+CHR$(33%,Lowsec%,SWAP%
        (Lowsec%),27%,6%,3%
60060   Asm$=Asm$+CHR$(14%,0%,26%)
60070   Asm$=Asm$+CHR$(167%,254%,10%,56%
        ,5%,214%,10%,12%,24%,246%)
60080   Asm$=Asm$+CHR$(198%,48%,119%,121
        %,198%,48%,43%,119%,43%,43%,27%,
        16%,230%)
60090   Asm$=Asm$+CHR$(1%,8%,0%,33%,Low
        sec%,SWAP%(Lowsec%),17%,Curpos%,
        SWAP%(Curpos%),237%,184%,225%,20
        9%)
60100   Asm$=Asm$+CHR$(193%,241%,195%,Cl
        ockptr%,SWAP%(Clockptr%))
60110   Z%=FNBlockmove%(VARPTR(T$),Asmad
        r%-8%,8%)
60120   Z%=FNBlockmove%(VARPTR(Asm$),Asm
        adr%,LEN(Asm$))
60130   Link$=CHR$(243%,33%,Asmadr%,SWAP
        %(Asmadr%),34%,214%,255%,251%,20
        1%)
60140   Z%=CALL(VARPTR(Link$))
60150   RETURN 0%
60160 FNEND
```

För att ta bort klockan från skärmen igen gör du en RESET eller kör följande rutin:

```
10 ! SAVE UNCLOCK
20 !
30 IF PEEK2(65494%)>0% THEN 90 ! Ordinar
   ie klockrutin
40 Z%=FNUnclock%
50 Z%=PEEK2(65292%)-100% ! Sänk golvet
   igen
60 POKE 65292%,Z%,SWAP%(Z%)
70 POKE 65328%,Z%,SWAP%(Z%)
80 !
90 PRINT CHR$(12%) CUR(20%,0%) TIME$
100 END
60199 !
60200 DEF FNUnclock% LOCAL Clockptr%,Lin
      k$=10
60210   Clockptr%=PEEK2(PEEK2(65292%)-31
        %)
60220   Link$=CHR$(243%,33%,Clockptr%,SW
        AP%(Clockptr%),34%,214%,255%,251
        %,201%)
60230   Z%=CALL(VARPTR(Link$))
60240   RETURN 0%
60250 FNEND
```

Lennart Persson <5983>

Svårigheterna vid stora program i BASIC II

Här följer en kort specifikation över vad som händer vid stora program i BASIC II, och hur man beräknar ledigt utrymme.

Program kan lagras i .BAS-format (vanlig ASCII-fil utan krav) med kommandot LIST: Program kan också lagras i .BAC-format med hjälp av kommandot SAVE. Endast SAVE berörs nedan.

Som exempel har jag tagit en version av PFFAKT1 (ett ordinärt fakturaprogram), som är cirka 27000 bytes stort i källkod. SYS(4) visar att ledigt utrymme är 2174 bytes. Vid SAVE fixas programmet först upp, dvs syntaxkontroller utförs på koden och utrymme reserveras för variabler och strängar. SYS(4) ger nu END OF MEMORY. Det lediga utrymmet är alltså mindre än en buffer, och kan inte längre mätas direkt.

Det mäts istället genom att variera programmets storlek och se vad som händer. Gränserna nedan är givitvis programberoende, men kan ge god ledning.

1. Programstorleken ökas så att det lediga utrymmet minskas från 2174 bytes till 2140 bytes. Programmet fixas alltjämt upp.

2. När det lediga utrymmet ligger i intervallet 2120-2139 bytes inträffar följande märkliga fenomen. Programmet förmår alltjämt fixa upp, men kan ej längre läsas in från disk (END OF MEMORY)! Det går emellertid bra att squeza filen. Men återigen - källkoden är förlorad. Något att tänka på för programmerare, som ej tar backup med LIST.

3. När det lediga utrymmet ligger i intervallet 2115-2119 inträffar ett annat fenomen. Vid SAVE lyckas programmet "nästan" fixas up. Men både det som lagras, och det som behålls i primärminnet, är nu totalkvaddat. En listning är en nedslående syn - variabler och funktioner består nu mest av punkter och andra krumelurer. Återigen något för programmerare som ej tar backup.

4. Slutligen när det lediga utrymmet ligger i intervallet 2114-0, så läggs filen ut i enkelt BAS-format, men alltjämt med .BAC-extension. Den är dock inte förlorad, utan kan lätt läsas in igen. Den uppmärksamme märker genast att ingen tid åtgått för fixning. Samma resultat erhålls vid syntaxfel i koden. Det går ej att squeza dessa program.

Om vi vill kunna läsa in programmet med LOAD, så är alltså det lediga utrymmet för programändringar i just detta program endast 2174 - 2140 = 34 bytes.

Anm: När man försöker editera ett stort program, som fixats upp, så erhålls ofta END OF MEMORY. Detta åtgärdas då med att först editera rad 10 (vilket alltid går), varvid programmet fixas ned och vidare editering är trivial.

<840>
Ulf Lingärde