Här följer nu programrutinerna som utför denna utökning av BASIC.

För program som utför själva inlänkningen av de nya instruktionerna/funktionerna - se programmet EXTBAS i bilaga 2.

```
EXTBAS    ZPROG    Utvidgad BASIC
          ;*******************************************************
          ;*                                                     *
          ;*  EXTBAS                                             *
          ;*                                                     *
          ;*  820914                                             *
          ;*  NANCO Elektronik                                   *
          ;*  UPDATED ...                                        *
          ;*  SAVED AS EXTBAS.ASM                                *
          ;*******************************************************
          ;*  Utvidgning av BASIC med PROCEDURE, WAIT, GETITEM   *
          ;*                                                     *
          ;*  INPUT:    ---                                      *
          ;*                                                     *
          ;*  OUTPUT:   ---                                      *
          ;*                                                     *
          ;*  GLOBAL:                                            *
          ;*    INPUT:  SEC      - Sekunder för systemklocka     *
          ;*            TICK     - 100-delssekunder för system-  *
          ;*                       klocka                        *
          ;*            SCANEXPR - Fast rutin för Uttrycksscan-*
          ;*                       ning                          *
          ;*    OUTPUT: ---                                      *
          ;*                                                     *
          ;*  LOCAL:    TP_INT   - Typ = heltal för SCANEXPR   *
          ;*            TP_STR   - Typ = sträng för SCANEXPR   *
          ;*            TP_END   - Slut på syntaxkod           *
          ;*            IOFFSET  - Startkod för nya instruktioner*
          ;*            FOFFSET  - Startkod för nya funktioner  *
          ;*******************************************************
          ;
          *PAGE 32
;
;
TP_INT    EQU   01H             ;Integer
TP_STR    EQU   02H             ;Sträng
TP_END    EQU   020H            ;Slut på syntaxkod
;
SEC       EQU   0FFF4H          ;Adress till sekund i system-
                                 klockan
TICK      EQU   0FFF5H          ;Adress till 100-delar i system-
                                 klockan
SCANEXPR  EQU   001DH           ;Fast adr för scanning av
                                 uttryck
IOFFSET   EQU   0D0H            ;Startkod för X-instruktioner
FOFFSET   EQU   0D0H            ;Startkod för X-funktioner
;
;
          ORG   08000H
;
;
XI_TAB    EQU   *
          DEFW  0               ;Länk till nästa instruktions-
                                 lista
          DEFB  IOFFSET         ;Internkodsoffset
```

```
          DEFB  EXQEND-EXQLST/2  ;Antalet nya instruktioner
          DEFW  EXQLST           ;Start för JP-tabell
          DEFW  TXTLST           ;Start för textlistan
          DEFW  SYNTLST          ;Start för syntaxhopptabell
;
;
          DEFW  0                ;Länk till nästa funktionslista
          DEFB  FOFFSET          ;Internkodsoffset
          DEFB  FEX_END-FEX_LST/2        ;Antalet nya funk-
                                            tioner
          DEFW  FEX_LST          ;Start för exekveringshopp-
                                    tabell
          DEFW  FTXT_LST         ;Start för textlistan
          DEFW  FSYN_LST         ;Start för syntaxkontroll-
                                    vektor
;
;
FEX_LST   EQU   *
          DEFW  E_GETIT          ;Adress till GETITEM-exekve-
                                    ringsrutin
;
FEX_END   EQU   *
;
;
FTXT_LST  EQU   *
          DEFB  080H
          DEFM  'GETITEM'
;
          DEFB  OFFH               ;Tabellslut
;
;
FSYN_LST  EQU   *
          DEFB  080H               ;Kod för GETITEM
          DEFB  TP_STR           ;Uttyp = Sträng
          DEFB  TP_STR           ;Inparameter 1 = Sträng
          DEFB  TP_INT+TP_END    ;Inparameter 2 = Heltal
;
          DEFB  OFFH               ;Slut på syntaxlista
;
;
;
;
EXQLST    EQU   *
          DEFW  E_PROC           ;Exekvera PROCEDURE
          DEFW  E_WAIT           ;Exekvera WAIT x
;
EXQEND    EQU   *
;
;
TXTLST    EQU   *
          DEFB  080H
          DEFM  'PROCEDURE'
;
          DEFB  081H
          DEFM  'WAIT'
;
          DEFB  OFFH               ;Slut på lista över reservera-
                                    de ord
;
;
```

```
SYNTLST   EQU   *
          DEFW  S_PROC
          DEFW  S_WAIT
;
;
;
S_WAIT    EQU   *
S_PROC    EQU   *
          LD    B,1               ;Scanna heltalsuttryck
          JP    SCANEXPR          ;Scanningsrutin
;
;
E_PROC    EQU   *
          RST   020H              ;Exekvera PROCEDURE
          RET
;
;
E_WAIT    EQU   *
          RST   020H              ;Exekvera WAIT x
          XOR   A
          LD    (TICK),A
          LD    A,(SEC)
          LD    C,A
;
WAIT      EQU   *
          LD    A,H               ;Kontroll mot tidsdelay
          OR    L
          RET   Z
;
          LD    A,(SEC)
          CP    C
          JR    Z,WAIT
          LD    C,A
          DEC   HL
          JR    WAIT
;
;
E_GETIT   EQU   *
          POP   HL                ;Hämta inparameter 2
          LD    IX,0
          ADD   IX,SP             ;Sätt IX till strängens
                                   parameterblock
          PUSH  DE                ;Spara undan instruktionspeka-
                                   ren
          EX    DE,HL             ;Ladda DE med inparameter 2
          CALL  GETSLEN           ;Läs längd
          LD    HL,3
          XOR   A
          SBC   HL,BC             ;Kontroll om sträng är för
                                   kort (<3 tkn)
          JR    NC,BLNK           ;Returnera tom sträng i så
                                   fall
;
          CALL  GETSADR           ;Läs adress till sträng
          LD    A,(HL)            ;Hämta först 2 byte (Antal
                                   items)
          INC   HL
          LD    H,(HL)
          LD    L,A
          XOR   A
```

```
          SBC   HL,DE             ;Kontroll om itemnr är för
                                    stort
          JR    C,BLNK            ;Returnera tom sträng i så
                                    fall
          ;
          CALL  GETSADR           ;Ladda HL med adress till
                                    sträng
          INC   HL
          INC   HL
;
;
GETIT_L   EQU   *
          DEC   DE
          LD    A,E
          OR    D
          LD    C,(HL)
          LD    B,0
          INC   HL                ;Sätt HL adr efter längd
          JR    Z,FND_ITEM        ;Item funnen, avbryt
          ADD   HL,BC             ;Hämta nästa item
          JR    GETIT_L
;
;
FND_ITEM  EQU   *
          CALL  PUTSADR           ;Spara nya adressen
          CALL  PUTSLEN           ;Spara nya längden
          JR    STRRET
          ;
          ;
          ;
BLNK      EQU   *
          LD    BC,0
          CALL  PUTSLEN           ;Sätt stränglängd till 0 (´´)
;
STRRET    EQU   *
          POP   DE                ;Hämta instruktionspekaren
          RST   028H              ;Fortsätt BASIC-exekvering
;
;
;
GETSLEN   EQU   *
          LD    C,(IX+4)          ;Ladda BC med aktuell längd
          LD    B,(IX+5)
          RET
;
GETSADR   EQU   *
          LD    L,(IX+2)          ;Ladda HL med adr till data-
                                    area
          LD    H,(IX+3)
          RET
;
PUTSLEN   EQU   *
          LD    (IX+4),C          ;Ladda nya stränglängden
          LD    (IX+5),B
          RET
;
PUTSADR   EQU   *
          LD    (IX+2),L          ;Ladda nya adr till dataarean
          LD    (IX+3),H
          RET
```

```
;
;
;
;
;
;
            END    XI_TAB
```

5.3.4 Enhetslista

Varje yttre enhet (diskettstation, skrivare, plotter etc) är ansluten till BASIC-interpretatorn som en logisk enhet. Enheterna är samlade i en enhetslista. Om man själv ansluter kort, tex I/O-kort, kan kortet länkas in i enhetslistan.

Enhetslistans utseende är:

```
Adress     Vidarepekare (l)  (Pekare till nästa enhet i listan)
Adress+1   Vidarepekare (h)  (Pekare till nästa enhet i listan)
Adress+2   Namn   Första bokstaven
Adress+3   Namn   Andra bokstaven
Adress+4   Namn   Tredje bokstaven
Adress+5   Adress till enhetens drivrutin (l)
Adress+6   Adress till enhetens drivrutin (h)
```

Ett sådant block skall finnas för varje enhet i enhetslistan.

För att hitta den första enheten i enhetslistan gör man:

```
10 Enhet1=PEEK2(65403)
```

Enhet1 pekar nu till första byten i vidarepekaren för första enheten i listan.

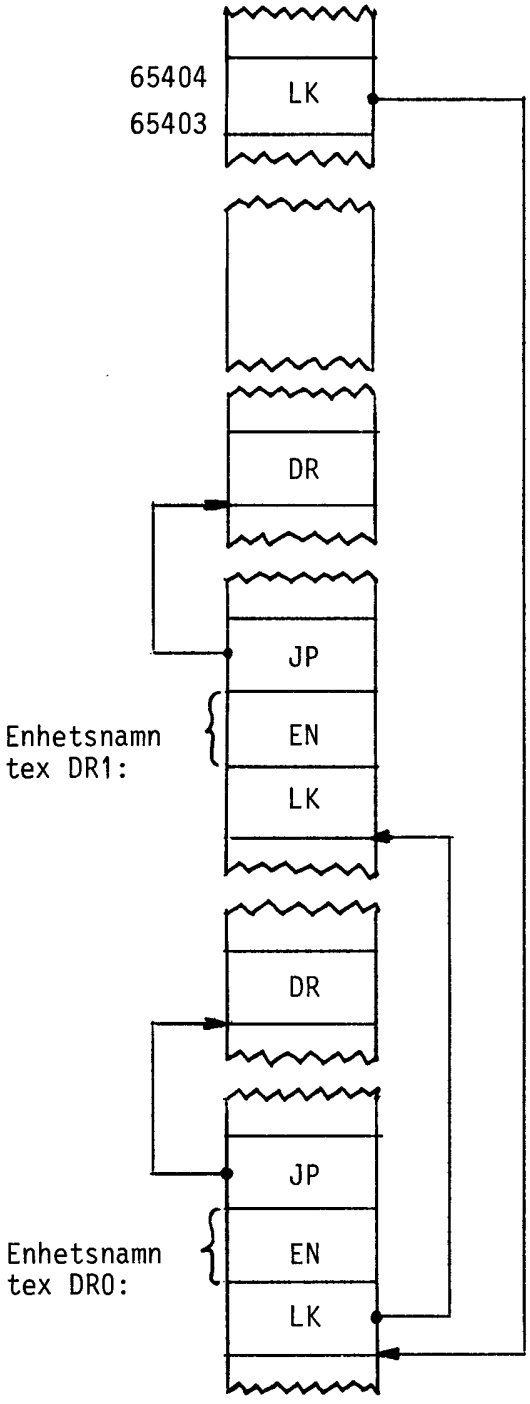
Genom att sätta vidarepekaren till noll (0) så har man indikerat att aktuell enhet är den sista i listan.

Namnen i enhetslistan måste vara exakt tre (3) tecken. Är enhetsnamnet ej så långt så skall namnet fyllas på med blanka tills längden blir tre.

Inparameter till drivrutinen är register A, som innehåller numret på den rutin som skall väljas.

Lista över anrop till drivrutin

Reg. A	Rutinnamn	Förklaring
0	OPEN	Öppna en fil på enheten
1	PREPARE	Skapa en fil på enheten
2	CLOSE	Stäng en fil på enheten
3	INPUT	Läs in en rad från enheten (INPUT)
4	PRINT	Skriv ut en rad på enheten (PRINT)
5	GET	Läs in tecken från enheten (GET)
6	PUT	Skriv ut tecken på enheten (PUT)
7	BL.IN	Läs in ett record från enheten
8	BL.UT	Skriv ut ett record på enheten
9	DELETE	Radera en fil på enheten
10	RENAME	Byt namn på en fil på enheten



DR = Drivrutin för enhet
EN = Enhetens namn
JP = Pekare till drivrutin
LK = Länk till nästa enhet

Enhetslistans utseende

5.3.5 Fillista

Liksom det för variablerna finns en variabellista så finns det för de öppnade filerna en fillista, som innehåller information om varje fil som används. Början på denna lista kan fås genom:

```
10 Filstart=PEEK2(65344)
```

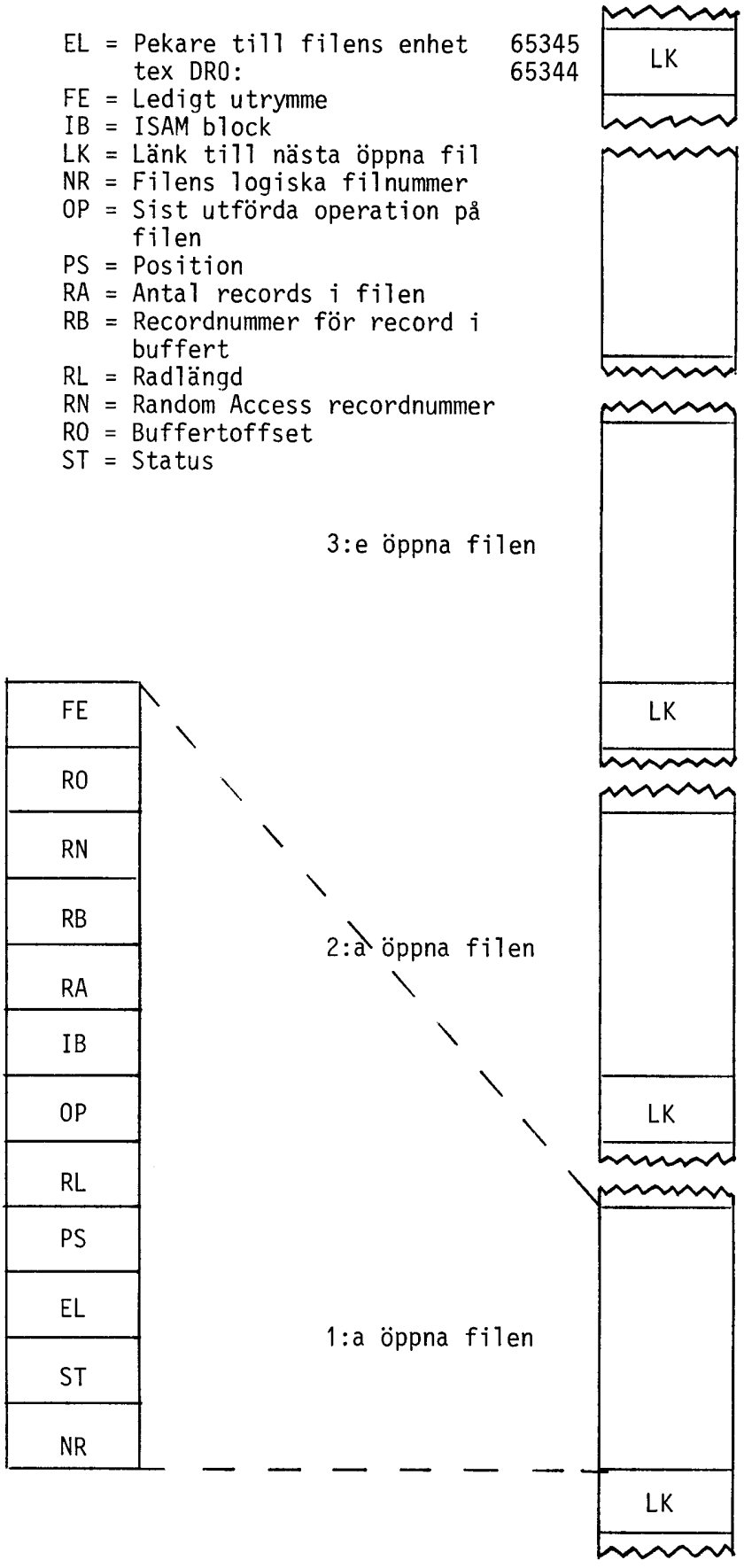
Listan är uppbyggd som en länkad lista där varje fil har ett kontinuerligt område (Fil-map). Vid assemblerprogrammering där filer används skall vid anrop av en DOS-rutin register IX peka på början av fil-mapen.

FIL-MAP

IX+0	W	Pekare till nästa fil
1		
2	B	Logiskt filnummer
3	B	Status *1
4	W	Pekare till enhetens plats i enhetslistan
5		
6	W	Position
7		
8	W	Radlängd
9		
10	B	Sista operation
11	W	ISAM-block
12		
13	W	Recordantal i fil
14		
15	W	Recordnummer i buffert
16		
17	W	Random Access recordnummer
18		
19	W	Random Access buffertoffset
20		
21		Ledigt (Används lokalt av vissa enheter)
22		Ledigt -"-
23		Ledigt -"-

*1 LU STATUS BYTE

Bit	Namn	Förklaring
7		
6		
5	LUS.FAST	Används för kassetthantering
4	LUS.ERR	Error har inträffat under CLOSE
3	LUS.WRDL	Bufferten måste skrivas ut.
2	LUS.IACT	Interaktiv.
1	LUS.PERM	Permanent
0	LUS.OPN	Filen öppen



Fillistans utseende

5.3.6 Variabellista

Detta avsnitt handlar om utseendet på variabellistan. Denna används av BASIC-interpretatorn tex vid kontroll av att aktuell längd för en sträng ej överskrider dimensionerad längd. I listan lagras även variablernas värde, namn och typ.

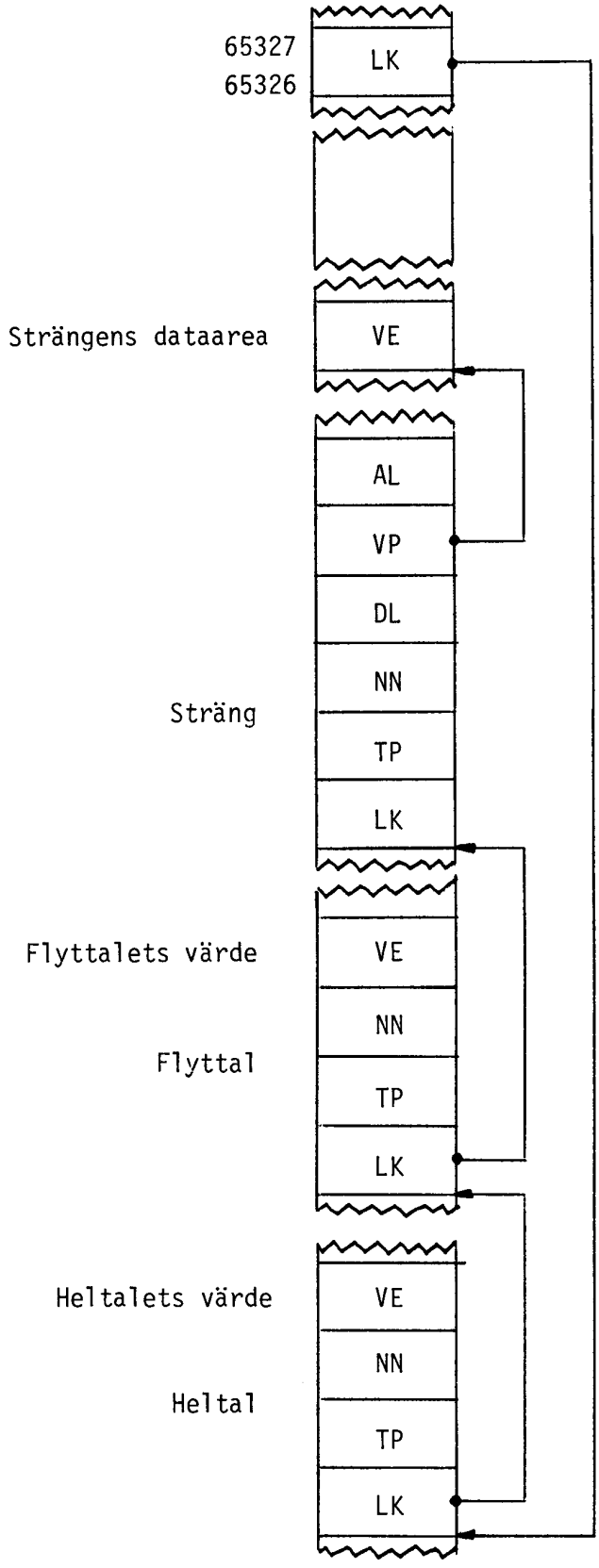
Listan är uppbyggd som en länkad lista där varje variabel har ett kontinuerligt område. För att hitta början på listan gör man

```
10 Varstart=PEEK2(SYS(12))
```

Här nedan följer utseendet på detta område för varje variabeltyp.

INTEGER

Skalär	Vektor / Matris
Länk(l)	Länk(l)
Länk(h)	Länk(h)
Typ	Typ
Namn	Namn
Värde(l)	Storlek(l)
Värde(h)	Storlek(h)
	Värdepekare(l)
	Värdepekare(h)
	Antal index
	Elementstorlek
	Min.index(l)
	Min.index(h)
	Max.index(l)
	Max.index(h)
	Avstånd(l)
	Avstånd(h)
	De sex sista byten upprepas för varje index.



Variabellistans utseende

AL = Aktuell längd för sträng
DL = Dimensionerad längd för sträng
LK = Länk till nästa variabel i listan
NN = Namn på variabeln,mm
TP = Typ på variabeln, mm
VE = Variabelns värde
VP = Pekare till strängens värde

FLOAT

Skalär	Vektor / Matris
Länk(l)	Länk(l)
Länk(h)	Länk(h)
Typ	Typ
Namn	Namn
Fvärde(1)	Storlek(l)
Fvärde(2)	Storlek(h)
Fvärde(3)	Värdepekare(l)
Fvärde(4)	Värdepekare(h)
Fvärde(5)	Antal index
Fvärde(6)	Elementstorlek
Fvärde(7)	Min.index(l)
Fvärde(8)	Min.index(h)
	Max.index(l)
De fyra sista byten	Max.index(h)
används bara vid	Avstånd(l)
DOUBLE.	Avstånd(h)
	De sex sista byten upprepas för varje index.

STRÄNG

Skalär	Vektor / Matris
Länk(l)	Länk(l)
Länk(h)	Länk(h)
Typ	Typ
Namn	Namn
Dim.längd(l)	Storlek(l)
Dim.längd(h)	Storlek(h)
Textpekare(l)	Värdepekare(l)
Textpekare(h)	Värdepekare(h)
Akt.längd(l)	Antal index
Akt.längd(h)	Elementstorlek
	Min.index(l)
	Min.index(h)
	Max.index(l)
	Max.index(h)
	Avstånd(l)
	Avstånd(h)
	De sex sista byten upprepas för varje index.

TECKENFÖRKLARING

Länk	Pekare till nästa variabel i listan	
Typ	Anger variabeltyp, m.m.	*2
Namn	Anger variabelnamn	*3
Värde	Variabelns värde	
Fvärde	Flyttalets värde (Se kap 3.1.1). Fvärde(5) - Fvärde(8) finns ej i SINGLE	
Storlek	Dataareans storlek	
Värdepekare	Pekar till variabelns dataarea	*1

Antal index	Antal dimensioner
Elementstorlek	Antalet byte variabeln upptar 2 Heltal 4 Flyttal (SINGLE) 8 Flyttal (DOUBLE) 6 Sträng
Min.index	Minsta index för aktuell dimension
Max.index	Högsta index för aktuell dimension
Avstånd	Avstånd mellan dimensioner. Beräknas ur: Elementstorlek*(Max.index-Min.index)
Dim.längd	Dimensionerad längd
Textpekare	Pekare till början av strängens värde
Akt.längd	Aktuell längd på strängen

*1 VARIABELNS DATAAREA

För strängar är dataareans utseende:

Dim.längd(l)
Dim.längd(h)
Textpekare(l)
Textpekare(h)
Akt.längd(l)
Akt.längd(h)

Dessa sex byte upprepas för varje index. För heltal och flyttal innehåller dataarean variablernas värden. Värdena ligger så här: (med DIM A(1,1,1))

Adress	A(0,0,0)
Adress+2	A(0,0,1)
Adress+4	A(0,1,0)
.	A(0,1,1)
.	A(1,0,0)
.	A(1,0,1)
.	A(1,1,0)
.	A(1,1,1)

*2 UTSEENDE PÅ TYPBYTEN

Vid korta variabelnamn innehåller denna byte information om variabeltyp och namnsuffix enligt bitkartan nedan.

bit:	7	6	5	4	3	2	1	0
	s	s	s	s	t	t	t	

där s s s s står för suffix.
t t t står för typ.

t t t	Betydelse	Exempel
0 0 0	Flyttal	A.
0 0 1	Heltal	A%
0 1 0	Sträng	A¤
0 1 1		
1 0 0	Indexerat flyttal	A.(...)
1 0 1	Indexerat heltal	A%(...)
1 1 0	Indexerad sträng	A¤(...)
1 1 1		

s s s s	Suffix	Exempel
0 0 0 0	0	A0
0 0 0 1	1	A1
0 0 1 0	2	A2
0 0 1 1	3	A3
0 1 0 0	4	A4
0 1 0 1	5	A5
0 1 1 0	6	A6
0 1 1 1	7	A7
1 0 0 0	8	A8
1 0 0 1	9	A9
1 0 1 0		
1 0 1 1		
1 1 0 0		
1 1 0 1		
1 1 1 0		
1 1 1 1	Suffix saknas	A

Vid långa variabelnamn innehåller typbyten information om variabeltyp samt en offset till namnet.

bit:	7	6	5	4	3	2	1	0
	f	f	f	f	f	t	t	t

där f f f f f är den låga delen i variabeloffset för namnet.
t t t är variabeltypen. Se ovan.

*3 UTSEENDE PÅ NAMNBYTEN

Denna byte innehåller ett heltal som representerar identifierarens bokstav.

Heltal	Namn
1	A
2	B
3	C
.	.
.	.
.	.
29	
32	Långt variabelnamn!
.	
.	
.	

De ovan beskrivna byten är vid långa variabelnamn en offset för variabelnamnet i förhållande till början på listan över långa variabelnamn. Denna offset fås ur:

(Namnbyten-32)*32+Suffixdelen

där suffixdelen är fem bitar.

UTSEENDE PÅ LÅNGA VARIABELLISTAN

```
Adress    Null                  Hit pekar värdet av EOFA + neg.
                                offset
Adress+1  Null
    .     Null
    .     Null
    .     N
          A
          M
          N
          1
          sep-byte
          N
          A
          M
          N
          2
          sep-byte
          .
          .
          .
          N
          A
          M
          N
          ETX-byte
          Neg. offset (l)
          Neg. offset (h)       Hit pekar värdet av EOFA
```

Null	Dessa byte (4 st) är under exekvering normalt CHR¤(0).
NAMN	Namnet på det långa variabelnamnet (här: NAMN1, NAMN2, NAMN)
sep-byte	Separerar namnen och anger att fler följer i listan. SKALL vara satt till 254 (FE hex).
ETX-byte	Anger slut på långa variabellistan. SKALL vara satt till 255 (FF hex).
Neg. offset	Är en negativ offset till listans början.

ARBETSGÅNG FÖR ATT HITTA EN VARIABEL

* Ta fram variabeloffset ur namn- och typbyten.
* Hämta värdet av EOFA (refereras som EOFA).
* Hämta den negativa offseten genom
  Negativ offset = PEEK2(EOFA-1)
* Placera en adress som pekar till början på listan men med null-byten förbilästa genom EOFA + negativ offset + 4.
* Addera variabeloffseten till ovanstående värde.
* Den byte som adressen nu pekar på är 1:a bokstaven i namnet.

Exempel:

```
10000 Var¤="TEST"
10010 Namnbyte=PEEK(VAROOT(Var¤)-1)
10020 Typbyte=PEEK(VAROOT(Var¤)-2)
10030 Typoffset=(Typbyte AND 248)/8
10040 Varoffset=(Namnbyte-32)*32+Typoffset
10050 Eofa=PEEK2(65288)
10060 Negoffset=PEEK2(Eofa-1)
10070 Liststart=Eofa+Negoffset+4
10080 Varpos=Liststart+Varoffset
10090 PRINT CHR¤(PEEK(Varpos))
10100 PRINT CHR¤(PEEK(Varpos+1))
10110 PRINT CHR¤(PEEK(Varpos+2))
```

Körning av programmet ger utskriften:

```
V
a
r
ABC800
```

5.4 TRACE och debugger

5.4.1 Felsökning

Felsökning används vid uttestning av program för att hitta logiska fel. Dessa möjligheter har förbättrats på ABC800 gentemot ABC80.

Tillgängligt är

* TRACE/NO TRACE
* Single-step
* Användardefinierade felsökningsrutiner (se avsnitt 5.4.2)

TRACE/NO TRACE

TRACE fungerar som på ABC80 genom att skriva ut radnummer som exekveras under körningen. Observara felet på den tidiga versionen av BASIC II där radnummer större än 32767 skrivs ut negativa (heltalsutskrift).

Skillnaden mot ABC80 BASIC är den att man kan styra utskriften till vilken logisk enhet som helst och inte bara till bildskärmen (CON:). Vanligtvis styrs utskriften till printern. Nyttan av detta är att vid TRACE av lite större program behöver inga radnummer gå förlorade eftersom dessa inte "kan försvinna ur bildskärmen".

NO TRACE stänger av TRACE.

Single-step

Om man trycker CTRL-C en (1) gång, avbryts programmet inte utan man befinner sig i "single-step" mode. Om man nu trycker CTRL-S, utförs nästa instruktion och därefter väntar programmet på ett nytt CTRL-S eller en tangentnedtryckning. En tangentnedtryckning i stället för CTRL-S innebär att programmet körs som vanligt.

Vill man avbryta programmet skall man trycka CTRL-C två (2) gånger. Men även här går det att fortsätta körningen av programmet. Detta kan göras på två sätt:

CON Fortsätter programmet där det stoppades.

GOTO nr Fortsätter programmet på rad "nr".

5.4.2 Användardefinierade felsökningsrutiner

I stället för att alltid vara hänvisad till att använda BASICens TRACE kan man nu skriva egna TRACE-rutiner i assembler. Att länka dessa till BASIC är inte svårt. Det går till så att man lägger adressen till sin assemblerrutin på adresserna 65431-65432.

OBS!
På adressen 65430 skall assemblerinstruktionen JP (195) ligga. Är denna byte ej 195 kommer rutinen aldrig att utföras.

Här följer nu en funktion med tillhörande assemblerrutin som på rad 23 skriver ut aktuellt radnummer följt av adressen på vilken stacken finns. Därefter följer värdet av HEAP. Detta är intressant eftersom skillnaden mellan stacken och HEAP anger hur mycket ledigt minne det finns.

```
10000 DEF FNPlats
10010   !
10020   ! Spara undan gamla pekaren till TRACE-rutin
10030   Oldtrace=PEEK2(65431)
10040   !
10050   ! Här följer assemblerrutinen i maskinkod
10060   POKE 65041,229,213,197,245,253,110,82,253,102,83
10070   POKE 65051,34,192,254,1,6,0,33,203,254,205
10080   POKE 65061,134,254,14,6,33,215,254,205,134,254
10090   POKE 65071,14,6,33,226,254,205,134,254,253,110
10100   POKE 65081,69,253,102,70,17,215,254,205,146,254
10110   POKE 65091,253,110,10,253,102,11,17,226,254,205
10120   POKE 65101,146,254,253,110,58,253,102,59,126,254
10130   POKE 65111,136,40,18,43,86,43,94,43,43,62
10140   POKE 65121,135,190,32,19,33,203,254,235,205,146
10150   POKE 65131,254,33,194,254,1,38,0,205,11,0
10160   POKE 65141,205,2,0,42,192,254,253,117,82,253
10170   POKE 65151,116,83,241,193,209,225,201,213,229,54
10180   POKE 65161,32,35,235,225,11,237,176,209,201,175
10190   POKE 65171,1,240,216,205,173,254,1,24,252,205
10200   POKE 65181,173,254,1,156,255,205,173,254,14,246
10210   POKE 65191,205,173,254,125,24,12,213,30,255,28
10220   POKE 65201,9,56,252,237,66,179,209,200,246,48
10230   POKE 65211,18,19,62,48,201
10240   POKE 65218,27,61,55,32,76,73,78,69,32,32
10250   POKE 65228,32,32,32,32,32,83,84,65,67,75
10260   POKE 65238,32,32,32,32,32,32,32,72,69,65
10270   POKE 65248,80,32,32,32,32,32,32,32
10280   !
10290   ! Inlänkning av TRACE-rutin
10300   POKE 65431,65041,SWAP%(65041)
10310   RETURN 0
10320 FNEND
```

För att återställa det förra värdet för pekaren till TRACE-rutiner gör man i slutet av programmet:

```
POKE 65431,Oldtrace,SWAP%(Oldtrace)
```

Assemblerrutinen i källkod.

```
DEBUG1    ZPROG    Skriver ut HEAP, SP för varje BASIC rad
          ;*************************************************************
          ;*                                                           *
          ;*  RUTINNAMN "DEBUG1"                                       *
          ;*                                                           *
          ;*  820827                                                   *
          ;*  NANCO Elektronik                                         *
          ;*  SAVED AS DEBUG1.ASM                                      *
          ;*************************************************************
          ;*  För varje BASIC-rad visas värdet av stackpekaren         *
          ;*  och pekaren till första lediga byte i minnet (HEAP)      *
          ;*                                                           *
          ;*  INPUT:    ---                                            *
          ;*                                                           *
          ;*  OUTPUT:   ---                                            *
          ;*                                                           *
          ;*  GLOBAL:                                                  *
          ;*    INPUT:  (IY+52H)  - LSH Akt cursorposition             *
          ;*            (IY+53H)  - MSH Akt cursorposition             *
          ;*            (IY+0AH)  - LSH Pntr till 1:a lediga byte      *
          ;*            (IY+0BH)  - MSH Pntr till 1:a lediga byte      *
          ;*            (IY+3AH)  - LSH Instruktionspekare             *
          ;*            (IY+3BH)  - MSH Instruktionspekare             *
          ;*            (IY+45H)  - LSH Undanlagrad stackpekare        *
          ;*            (IY+46H)  - MSH Undanlagrad stackpekare        *
          ;*    OUTPUT: ---                                            *
          ;*                                                           *
          ;*  LOCAL:    (CUR)     - Gamla värdet på cursor             *
          ;*            PRINT     - Utskriftsbuffert                   *
          ;*            INCHAR    - Läser ett tecken på CON:           *
          ;*            OUTSTR    - Skriver ut en sträng på CON:       *
          ;*************************************************************
          ;
          *PAGE 55
          ;
          ORG   0FE11H            ;Rutinen läggs i POKE-arean
          ;
OUTSTR    EQU   000BH             ;Skriver ut en sträng
INCHAR    EQU   0002H             ;Läser ett tecken från CON:
          ;
DEBUG1    EQU   *
          PUSH  HL                ;Lagra undan registerinnehållen
          PUSH  DE                ;OBS!!! Detta måste göras annars
          PUSH  BC                ;kommer BASIC att gå fel
          PUSH  AF
          LD    L,(IY+52H)        ;Aktuell kolumn och
          LD    H,(IY+53H)        ;aktuell rad
          LD    (CUR),HL          ;lagras undan
          LD    BC,0006H          ;Längd på radnummerfältet
          LD    HL,LINE           ;och dess adress
          CALL  SPACES            ;Fyller en area med blanka
          LD    C,06H             ;Längd på stackfältet
          LD    HL,STACKPR        ;och dess adress
          CALL  SPACES
          LD    C,06H             ;längd på HEAP-fältet
          LD    HL,HEAPPR         ;och dess adress
          CALL  SPACES
```

```
            LD    L,(IY+45H)      ;Hämta värdet på SP använd av BASIC
            LD    H,(IY+46H)
            LD    DE,STACKPR      ;Pekare var värdet skall läggas
            CALL  BINASC          ;Konverterar binärtal till sträng
            LD    L,(IY+0AH)      ;Hämta värdet på HEAP
            LD    H,(IY+0BH)
            LD    DE,HEAPPR       ;Pekare var värdet skall läggas
            CALL  BINASC          ;Ännu en konvertering
            LD    L,(IY+3AH)      ;Hämta instuktionspekare från BASIC
            LD    H,(IY+3BH)
            LD    A,(HL)          ;Hämta byten den pekar på
            CP    88H             ;88H <=> internkod för ':'
            JR    Z,PRINTOUT      ;I så fall behövs inget nytt radnr
            DEC   HL
            LD    D,(HL)          ;Hämta höga delen av radnr
            DEC   HL
            LD    E,(HL)          ;Hämta låga delen av radnr
            DEC   HL
            DEC   HL              ;Läs förbi satslängd
            LD    A,87H           ;87H <=> internkod för ny sats
            CP    (HL)            ;Jämför med aktuell byte
            JR    NZ,NOPRINT      ;Om ej ny sats behövs ej nytt radnr
            LD    HL,LINE         ;annars hämta pekare till radnrarea
            EX    DE,HL
            CALL  BINASC          ;Konvertera vårt radnr till sträng
;
PRINTOUT    EQU   *
            LD    HL,PRINT        ;Adress till utskriftsfältet
            LD    BC,SLUT-PRINT   ;Längden på detta
            CALL  OUTSTR          ;Skriv ut vår rad på rad 23
            CALL  INCHAR          ;Anropa läs tecken rutinen (GET)
;
NOPRINT     EQU   *
            LD    HL,(CUR)        ;Hämta cursor vid inträde i rutinen
            LD    (IY+52H),L      ;och lagra den som aktuell cursor
            LD    (IY+53H),H
            POP   AF              ;Återlagra registerinnehållen
            POP   BC
            POP   DE
            POP   HL
            RET
;
SPACES      EQU   *
            PUSH  DE              ;Spara undan DE
            PUSH  HL              ;Adress att lägga blank i
            LD    (HL),' '        ;Ladda den med en blank
            INC   HL              ;Öka adressen
            EX    DE,HL
            POP   HL              ;Adress att läsa från vid LDIR
            DEC   BC              ;En blank är redan lagrad
            LDIR                  ;Fyll area med blanka
            POP   DE              ;Hämta tillbaka DE
            RET
;
BINASC      EQU   *
            XOR   A               ;Initiera för ev inledande nollor
            LD    BC,0D8F0H       ;Komplement till 10 000
            CALL  CONV            ;Ger ASCII för 10 000-talssiffra
```

```
          LD    BC,0FC18H       ;Komplement till 1 000
          CALL  CONV            ;Ger ASCII för 1 000-talssiffra
          LD    BC,0FF9CH       ;Komplement till 100
          CALL  CONV            ;Ger ASCII för 100-talssiffra
          LD    C,0F6H          ;Komplement för 10
          CALL  CONV            ;Ger ASCII för 10-talssiffra
          LD    A,L             ;Omvandla entalssiffran
          JR    ASCII           ;till ASCII
;
CONV      EQU   *
          PUSH  DE
          LD    E,0FFH          ;Initiera sifferräknaren
;
NO_MINUS EQU  *
          INC   E               ;Öka sifferräknaren
          ADD   HL,BC           ;Addera till komplement till värde
          JR    C,NO_MINUS      ;Om ej minus fortsätt loop
          SBC   HL,BC           ;Addering gjordes 1 ggr för mycket
          OR    E               ;Kontroll om inledande nollor
          POP   DE
          RET   Z
;
ASCII     EQU   *
          OR    30H             ;Forma ASCII-kod för siffra
          LD    (DE),A          ;som lagras i textbuffert
          INC   DE              ;Justera pntrs
          LD    A,30H           ;Inga fler inledande 0 möjliga
          RET
          ;
CUR       DEFS  02H
PRINT     DEFB  1BH             ;Kod för cursorstyrning
          DEFB  '='             ;Kod för cursorstyring
          DEFB  17H+20H         ;Radnr+32
          DEFB  00H+20H         ;Kolumnposition+32
          DEFM  'LINE '
LINE      DEFM  '    '
          DEFM  'STACK '
STACKPR   DEFM  '    '
          DEFM  'HEAP '
HEAPPR    DEFM  '    '
SLUT      EQU   *
          ;
          END   DEBUG1
```

Nedanstående exempel visar värdet av en sträng vars VAROOT skickas med som parameter till BASIC-funktionen. På rad 23 visas namnet på strängen och den del av värdet som får plats på en 80-teckensrad. Ex. A9¤=ABC800.

Observera att rutinen inte skriver ut namnet vid långa variabelnamn utan i stället skriver ?¤.

```
10000 DEF FNStringtrace(Variabel)
10010   !
10020   ! Lagra undan gamla TRACE-rutinpekaren
10030   Oldtrace=PEEK2(65431)
10040   POKE 64256,229,213,245,253,110,54,253,102,55,237
10050   POKE 64266,91,135,251,167,237,82,194,131,251,197
10060   POKE 64276,253,110,82,253,102,83,34,137,251,33
10070   POKE 64286,143,251,17,144,251,1,78,0,54,32
10080   POKE 64296,237,176,237,91,135,251,33,143,251,213
10090   POKE 64306,27,26,254,32,56,2,62,255,198,64
10100   POKE 64316,119,35,254,63,40,18,27,26,230,120
10110   POKE 64326,203,63,203,63,203,63,254,15,40,4
10120   POKE 64336,198,48,119,35,54,36,35,54,61,35
10130   POKE 64346,235,225,35,35,78,35,70,197,35,78
10140   POKE 64356,6,0,225,120,177,40,2,237,176,33
10150   POKE 64366,139,251,1,83,0,205,11,0,205,2
10160   POKE 64376,0,42,137,251,253,117,82,253,116,83
10170   POKE 64386,193,241,209,225,201,0,0,0,0,27
10180   POKE 64396,61,55,32
10190   !
10200   ! Placera den sökta variabelns VAROOT i de byte
10210   ! som är avsedda för kommunikation mellan
10220   ! BASIC-funktionen och ASSEMBLER-rutinen
10230   ! POKE 64391,Variabel,SWAP%(Variabel)
10240   !
10250   ! Länka in TRACE-rutinen
10260   POKE 65431,64256,SWAP%(64256)
10270   RETURN 0
10280 FNEND
```

För att återställa det förra värdet för pekaren till TRACE-rutinen gör man i slutet av programmet; POKE 65431,Oldtrace,SWAP%(Oldtrace).

Assemblerrutinen i källkod följer på nästa sida.

```
DEBUG2    ZPROG    Trace på en strängvariabel
          ;*********************************************************
          ;*                                                       *
          ;*  RUTINNAMN "DEBUG2"                                   *
          ;*                                                       *
          ;*  820827                                               *
          ;*  NANCO Elektronik                                     *
          ;*  SAVED AS DEBUG2.ASM                                  *
          ;*********************************************************
          ;*  "Trace" av en strängvariabel. Långa variabelnamn     *
          ;*  skrivs ut som: ?¤                                    *
          ;*                                                       *
          ;*  INPUT:    ---                                        *
          ;*                                                       *
          ;*  OUTPUT:   ---                                        *
          ;*                                                       *
          ;*  GLOBAL:                                              *
          ;*    INPUT:  (IY+36H) - LSH Senast ändrade variabel     *
          ;*            (IY+37H) - MSH Senast ändrade variabel     *
          ;*            (IY+52H) - Aktuell kolumn                  *
          ;*            (IY+53H) - Aktuell rad                     *
          ;*    OUTPUT: ---                                        *
          ;*                                                       *
          ;*  LOCAL:    VARPRINT - Utskriftsarea                   *
          ;*            INCHAR   - Rutin som läser in ett tecken   *
          ;*            OUTSTR   - Rutin som skriver ut en sträng  *
          ;*            (CUR)    - Undansparat värde på cursor     *
          ;*            (VAROOT) - Plats för tracad variabel       *
          ;*********************************************************
          ;
          *PAGE 55
          ;
          ORG   OFBOOH            ;Läggs i DOSBUF6
          ;
OUTSTR    EQU   000BH             ;Utskriftsrutin på CON:
INCHAR    EQU   0002H             ;Inläsningsrutin på CON:
          ;
DEBUG2    EQU   *
          PUSH  HL                ;Spara undan registerinnehållen
          PUSH  DE                ;OBS!!! Måste göras anners kommer
          PUSH  AF                ;BASIC att gå fel
          LD    L,(IY+36H)        ;Hämta senast ändrade variabel
          LD    H,(IY+37H)
          LD    DE,(VAROOT)       ;Variabel som skall "tracas"
          AND   A                 ;Nollställ carry
          SBC   HL,DE             ;Är värdet av TRCVAR-VAROOT <> 0
          JP    NZ,VARNEQ         ;så är det ej sökt variabel
          PUSH  BC
          LD    L,(IY+52H)        ;Akt. värde på kolumn och
          LD    H,(IY+53H)        ;akt. värde för rad
          LD    (CUR),HL          ;skall sparas undan
          LD    HL,VARPRT         ;Area som skall fyllas med blanka
          LD    DE,VARPRT+1       ;För följande LDIR !!
          LD    BC,4EH            ;Antalet blanka
          LD    (HL),' '          ;Fyll första byten med en blank
          LDIR                    ;Fyll resten av arean
          LD    DE,(VAROOT)       ;Hämta VAROOT till variabeln
          LD    HL,VARPRT         ;Pekare till utskriftsarean
          PUSH  DE                ;Spara undan VAROOT
          DEC   DE
```

```
          LD    A,(DE)          ;Hämta namnbyten till variabeln
          CP    20H             ;Test om långt variabelnamn
          JR    C,SHORTVAR
          LD    A,0FFH          ;i så fall förbered för  ?¤
;
SHORTVAR  EQU   *
          ADD   A,40H           ;Ta fram ASCII-representationen
          LD    (HL),A          ;och spara den i utskriftsarean
          INC   HL              ;Uppdatera pekaren
          CP    '?'             ;Långa variabelnamn refereras som:?¤
          JR    Z,NOSUF         ;och har ej något suffix
          DEC   DE
          LD    A,(DE)          ;Läs in typbyten för variabeln
          AND   78H             ;Maska ut suffixdelen
          SRL   A               ;Skifta ned dessa bitarna till de
          SRL   A               ;minst signifikanta bitarna
          SRL   A
          CP    0FH             ;Kontrollera om suffix finns
          JR    Z,NOSUF         ;annars hoppa
          ADD   A,30H           ;Ta fram ASCII-representationen
          LD    (HL),A          ;och lagra
          INC   HL              ;Uppdatera pekaren
;
NOSUF     EQU   *
          LD    (HL),'¤'        ;Lagra '¤'
          INC   HL
          LD    (HL),'='        ;och ett '='
          INC   HL
          EX    DE,HL
          POP   HL              ;Hämta VAROOT
          INC   HL
          INC   HL              ;Läs förbi dimensionerad längd
          LD    C,(HL)          ;Hämta låga delen av pekaren till
          INC   HL              ;värdet av strängen
          LD    B,(HL)          ;Hämta höga delen av pekaren till
          PUSH  BC              ;värdet av strängen
          INC   HL
          LD    C,(HL)          ;Hämta låga delen av stränglängen
          LD    B,00H           ;Längd > 255 => Trunkering
          POP   HL
          LD    A,B             ;Om BC = 0 så är
          OR    C               ;strängen = ''
          JR    Z,PRINTOUT      ;I så fall hoppa till utskriften
          LDIR                  ;annars lagra värdet i utskriftsarea
;
PRINTOUT  EQU   *
          LD    HL,VARPRINT     ;Pekare till utskriftsarea
          LD    BC,SLUT-VARPRINT ;Längd på texten
          CALL  OUTSTR          ;Anrop av utskriftsrutinen
          CALL  INCHAR          ;Anrop av läs tecken rutinen (GET)
          LD    HL,(CUR)        ;Hämta cursor vid rutinens start
          LD    (IY+52H),L      ;och lägg i kolumnpos. samt
          LD    (IY+53H),H      ;radpos.
          POP   BC              ;Återlagra registerinnehållen
;
VARNEQ    EQU   *
          POP   AF
          POP   DE
          POP   HL
          RET
```

```
          ;
VAROOT    DEFW  0000H             ;Undanlagringsplats för VAROOT
CUR       DEFW  0000H             ;Plats för cursor
VARPRINT  DEFB  1BH               ;Kod för cursorstyrning
          DEFB  '='               ;Kod för cursorstyrning
          DEFB  17H+20H           ;Rad+32
          DEFB  00H+20H           ;Kolumn+32
VARPRT    DEFS  4FH               ;Egentlig utskriftsarea
SLUT      EQU   *
          ;
          END   DEBUG2
```

Observera att man alltid måste spara undan registerinnehållen vid inträdet i TRACE-rutinen och återlagra dessa vid utträdet ur rutinen. I annat fall kommer BASIC att "gå bort sig".

Om man inte vill använda den egna TRACE-rutinen skall man återställa orginalvärdet med:

POKE 65431,187,44

Observera att den användardefinierade TRACE-rutinen kommer att användas vid TRACE tills man gör RESET eller ställer tillbaka adressen enligt ovan. Observera också att TRACE-rutinen kommer att vara inaktiv tills kommandot TRACE utförs.

Nedan följer en sammanställning på användbara adresser vid TRACE.

65431-32 Adress till användardefinierad TRACE-rutin.

65334-35 Pekare till sist ändrade variabel (VAROOT).

65332 Anger portnummer för senaste OUT-instruktion.

65338-39 Instruktionspekaren för BASIC.

65349-50 Värdet på "run-time" stackpekaren.

65290-91 HEAP.

Vid felsökning av assembler kan följande adresser vara intressanta:

65419-20 Vid ett NMI-interrupt (styrs från ABC800 bussen), hämtas hoppadressen härifrån. Genom att ändra dessa byte kan man få en egen NMI-hantering. Men kom ihåg att return görs med RETN (Return from Non-maskable interrupt).

65435-36 RST 08H hämtar sin hoppadress härifrån.

65438-39 RST 30H hämtar sin hoppadress härifrån.

65441-42 RST 38H hämtar sin hoppadress härifrån.

Vid dessa adresser är det återigen viktigt att man inte ändrar byten före adressen eftersom assemblerinstruktionen JP (195) ligger där.

5.4.3 TRACE vid ASSEMBLER

Vid debugging av assembler kan följande adresser vara intressanta:

65419-20 Vid ett NMI-interrupt hämtas hoppadressen härifrån. Genom att ändra dessa byte kan man få en egen NMI-hantering. Men kom ihåg att återhopp görs med RETN (Return from Nonmaskable Interrupt).

65435-36 RST 08H hämtar sin hoppadress härifrån

OBS!
RST 08H skall ej användas om man vill kunna "debugga" sitt program med programmet TRACE (som finns på Assembler 800-disketten) eftersom detta program använder RST 08H.
Vill man använda både programmet TRACE och en egen TRACE-rutin, kan man använda någon av nedanstående adresser.

65438-39 RST 30H hämtar sin hoppadress härifrån.

65441-42 RST 38H hämtar sin hoppadress härifrån.

Vid ovanstående adresser är det viktigt att man inte ändrar byten före adressen eftersom assemblerinstruktionen JP (195) ligger där.

6. ASSEMBLERPROGRAMMERING

NÄR BEHÖVER MAN ASSEMBLERPROGRAM?

* Då BASIC-tolken är för långsam!

  Exempel:
  Vid kommunikation med yttre enheter (printer etc) eller vid strånghantering, som kan hanteras mycket smidigt i assembler.

* För att få ned programstorleken!

  Oftast är BASIC mycket snål med att ta upp plats, men vid exempelvis maskinorienterade rutiner som kommunikation, flyttning av data, minneshantering etc, tar assembler mindre plats.

* När BASIC ej kan användas!

  Exempel:
  För kontroll och hantering av interrupt-logiken.

HUR SKALL MAN ANROPA ASSEMBLERPROGRAMMET?

* BASICens eget CALL — `Z=CALL(Adr,Data)`

* Definition av programmet som en logisk enhet

```
OPEN "PGM:" AS FILE 1
PRINT #1,Data
INPUT #1,Data
CLOSE 1
```

* Ny BASIC-instruktion (Ej så vanligt) — `Z=DOASM`

HUR SKALL MAN ÖVERFÖRA DATA?

För att överföra data mellan BASIC och assembler kan man låta data ha en fast plats i minnet, som är åtkomlig med PEEK och POKE. Eller så låter man data ligga i en variabel. Fördelen med det senare sättet är att programmet blir mer flexibelt då inga fasta adresser finns.

Då bara ett värde skall ges till assemblerrutinen kan man använda BASICens CALL genom att ge värdet i andra argumentet på CALL-satsen. Värdet hamnar då i DE-registret.

Om bara ett värde skall lämnas som utdata till BASIC-programmet kan man också använda BASICens CALL. Värdet, som skall returneras, placeras i HL-registret, som då görs tillgängligt i BASIC.

Exempel:

```
10 Z=CALL(24678,10)
```

Värdet 10 kommer att placeras i DE och variabeln Z kommer att få värdet som finns i HL vid uthoppet från assemblerrutinen.

Registerinnehållen måste återställas vid återhopp till BASICen (till de värden de hade då assemblerrutinen anropades).

Register I och R får EJ förändras i assemblerrutinen.

Ett sätt att komma över problemet med att bara ha ett argument är att lägga parametrarna, som man vill att assemblerrutinen skall ha tillgång till, i en sträng eller i en heltalsvektor samt anropa rutinen med variabelns VARPTR.

Exempel: Anrop med sträng

```
10000 DEF FNStrasm
10010   Dat¤=CVT%¤(Data1)+CVT%¤(Data2)+CVT%¤(Data3)
10020   RETURN CALL(Adress,VARPTR(Dat¤))
10030 FNEND
```

Exempel: Anrop med heltalsvektor

```
10000 DEF FNIntasm
10010   Dat(0)=Data1
10020   Dat(1)=Data2
10030   Dat(2)=Data3
10040   RETURN CALL(Adress,VARPTR(Dat(0)))
10050 FNEND
```

Vid inläggning av assemblerrutinen är det enkelt och bekvämt att lägga assemblerkoderna i en sträng mha CHR¤-funktionen, dvs:

```
Ass¤=CHR¤(kod,kod,...,kod)
```

Startadressen till assemblerrutinen kan då fås med VARPTR.

Exempel:

```
10000 DEF FNAsm LOCAL Asm¤=8
10010   Asm¤=CHR¤(235,1,13,0,205,11,0,201)
10020   Dat¤="Testprogram"
10030   RETURN CALL(VARPTR(Asm¤),VARPTR(Dat¤))
10040 FNEND
```

Denna funktion skriver ut texten "Testprogram" på skärmen.

GENERELL PROGRAMSTRUKTUR

Ett assemblerprogram består av:

1 "Huvud"	En kommentarsruta med uppgifter om programnamn, datum, upphovsmakare och fakta om programmet (indata, utdata, mm).
2 Parameterfält	Definitioner av parametrar som används i programmet.
3 Konstanter	Angivande av konstanter och deras värde.
4 Macrodefinitioner	Definition av macrokroppar.
5 Program	Ett program består av ett huvudprogram och en eller flera subrutiner.
6 Dataarea	Area med texter, fält, övriga data, mm.
7 END	

MACRO

Macro är en sekvens assemblerinstruktioner (macrokropp), som inte assembleras då de påträffas i texten utan assembleras vid varje tillfälle då macron anropas. Instruktionerna kommer då att kopieras till det läge där anropet gjordes. Detta går till på följande sätt i assemblatorn:

Vid varje macroanrop letas motsvarande macrokropp upp, varefter assembleringen fortsätter i macrokroppen tills dess pseudoinstruktionen ENDM (avslutar macrokropp) hittas. Assembleringen fortsätter då med instruktionen efter macroanropet.

Macrokroppen innehåller instruktionerna som skall utföras när macron anropas. Macrokroppens utseende är följande:

```
MACRO
Macronamn  Ev. parametrar
instruktion
     .
     .
     .
ENDM
```

Macrokroppen anropas med ett anrop som skiljer sig från det som används vid subrutiner, nämligen:

Macronamn Ev. parametrar

Man ser här att macro är ett sätt att definiera egna opcodes som tillägg till de redan existerande.

Macro används när man vill kombinera mindre programmeringsarbete (som vid subrutiner där man skriver sitt programavsnitt en gång och gör anrop till subrutinen) med snabbhet. (Vid subrutiner förloras tid eftersom CALL och RET måste göras.)

Macro kan med fördel även användas vid programavsnitt som liknar varandra, men där vissa delar eller värden på variabler skiljer sig mellan anropen. Man brukar vid dessa tillfällen även använda villkorlig assemblering, dvs pseudoinstruktionerna

COND - Start villkorlig assemblering.
ENDC - Slut villkorlig assemblering.

COND fungerar enligt följande:

```
COND villkor
instruktioner
ENDC
```

Ovanstående motsvaras i BASIC-notation av

```
IF villkor THEN instruktioner
```

Nackdelen med macro är att det är minneskrävande i motsats till subrutiner. Detta pga att assemblatorn kopierar macrokroppen varje gång ett macroanrop sker.

Exempel:

```
MUL       ZPROG Heltalsmultiplikation (MACRO)
          ;*************************************************************
          ;*                                                           *
          ;*  RUTINNAMN "MUL"                                          *
          ;*                                                           *
          ;*  820908                                                   *
          ;*  NANCO Elektronik                                         *
          ;*  SAVED AS MUL.ASM                                         *
          ;*************************************************************
          ;*  Utför en heltalsmutiplikation på positiva tal(0-65535)*
          ;*  Algoritm ur Knuth, Art of Computer Program               *
          ;*                                                           *
          ;*  INPUT:    ---                                            *
          ;*                                                           *
          ;*  OUTPUT:   HL         - Produkt                           *
          ;*            Carry=0    - Produkt O.K. (0 <= HL <= 65535)*
          ;*            Carry=1    - För stort tal (HL > 65535)        *
          ;*                                                           *
          ;*  GLOBAL:                                                  *
          ;*    INPUT:  Fak1       - Faktor                            *
          ;*            Fak2       - Faktor                            *
          ;*    OUTPUT: ---                                            *
          ;*                                                           *
          ;*  LOCAL:    ---                                            *
          ;*************************************************************
          ;
          *PAGE 55
          ;
          EXTERNAL FAK1,FAK2
          ;
          MACRO
          INIT  F1,F2,INTER      ;Initiering
          COND  INTER<>00H       ;INTER<>0  => Indirekt adressering
          LD    BC,(F1)
          LD    DE,(F2)
          ENDC
```

```
            COND  INTER=00H         ;INTER=0  => Direkt adressering
            LD    BC,F1
            LD    DE,F2
            ENDC
            ENDM
            ;
            MACRO
            ZT    ZTYP              ;Test om registerpar = 0
            COND  ZTYP=00H          ;0  => BC
            LD    A,B
            OR    C
            ENDC
            COND  ZTYP=01H          ;1  => DE
            LD    A,D
            OR    E
            ENDC
            COND  ZTYP=02H          ;2  => HL
            LD    A,H
            OR    L
            ENDC
            ENDM
            ;
            MACRO
            MUL2  MBTYP             ;Registerpar=Registerpar*2
            COND  MBTYP=00H         ;0  => BC
            SLA   C
            RL    B
            ENDC
            COND  MBTYP=01H         ;1  => DE
            SLA   E
            RL    D
            ENDC
            COND  MBTYP=02H         ;2  => HL
            SLA   L
            RL    H
            ENDC
            ENDM
            ;
            MACRO
            DIV2  DBTYP             ;Registerpar=Registerpar/2
            COND  DBTYP=00H         ;0  => BC
            SRA   C
            RR    B
            ENDC
            COND  DBTYP=01H         ;1  => DE
            SRA   E
            RR    D
            ENDC
            COND  DBTYP=02H         ;2  => HL
            SRA   L
            RR    H
            ENDC
            ENDM
;
MUL         EQU   *
            INIT  FAK1,FAK2,01H     ;Initiera BC och DE
            LD    HL,0000H          ;Beräknad produkt i HL
;
```

```
REPEAT    EQU   *
          ZT    00H             ;Test om register = 0 (BC)
          RET   Z               ;Om så, är multiplikation färdig
;
          BIT   0,C             ;Om BC är ett jämt tal (b0=0)
          JR    Z,MULEVEN       ;så öka ej produkten
;
          ADD   HL,DE           ;annars addrera till DE till produkt
          RET   C               ;Om carry sätts, tal>65535, error
;
MULEVEN   EQU   *
          MUL2  01H             ;Multiplicera register med 2 (DE)
          DIV2  00H             ;Dividera register med 2 (BC)
          JR    REPEAT          ;Loopa
          END   MUL
```

Objektkoden efter assembleringen följer nedan. (I mnemonics-form för bättre förståelse.)

```
MUL       LD    BC,(FAK1)
          LD    DE,(FAK2)
          LD    HL,0000H
REPEAT    LD    A,B
          OR    C
          RET   Z
          BIT   0,C
          JR    Z,MULEVEN
          ADD   HL,DE
          RET   C
MULEVEN   SLA   E
          RL    D
          SRA   C
          RR    B
          JR    REPEAT
```

MACRO - SUBRUTIN

Om man kan välja mellan att programmera ett avsnitt som en macro eller som en subrutin, följer nedan en tabell som anger vid villka tillfällen man sparar plats med macro respektive subrutin.

Rutinlängd (antal bytes)	Subrutin (antal anrop)	Macro (antal anrop)
1	Aldrig	Alltid
2	Aldrig	Alltid
3	Aldrig	Alltid
4	> 5	< 6
5	> 3	< 4
6	> 2	< 3
7	> 2	< 3
8	> 1	< 2
.	.	.
.	.	.
.	.	.

Man kan beräkna det totala minnesbehovet mha följande formler:
Macro: Total längd = Rutinlängd * Antal anrop
Subrutin: Total längd = Rutinlängd + 3 * Antal anrop + 1

BILAGA 1 ERRATA

	BASIC II 1.1	BASIC II 1.12
**	Exponentiering av noll med en decimal exponent ger felmeddelande 130 Ex: ;0**1.2 ger ERROR 130 (Pga att beräkningen görs enl Exp(log(0)*1.2).)	
ADD¤	Om man sätter antalet gällande siffror till 2 (-2) och skriver ;ADD¤(´0´,´.00001´,-2) fås utskriften 0. Dvs då man adderar noll (0) med ett tal med X gällande siffror faller talet bort om det är mindre än 1.E-Q där Q=X+2.	
	Om man skriver ut (med eller utan USING) ett tal med exponent större än 99 eller mindre än -99.	Felet rättat.
	Om man till ASCII-aritmetiken matar in exponenter större än 126 så blir det fel i lagringen (lagras negativt). Samma gäller för exponenter mindre än 127 (lagras positivt).	Felet rättat.
BOOLEAN	PRINT -0=0 blir falskt dvs <>-1.	Felet rättat.
CLOSE	CLOSE utan filnummer nollställer COMMON-variablerna.	Felet rättat.
COMMON	Variabeltilldelning före COMMON-deklarationer ger "dyk".	Ger ej "dyk" men är ej tillåtet.
COMP%	Ger felaktigt resultat på vissa jämförelser. Ex: ;COMP%(´500´,´.01´) ger -1 !!.	Felet rättat.

	BASIC II 1.1	BASIC II 1.12
CVT	CVT¤%( ) är ointelligent. Vid längd skilt från två fås ett felaktigt värde. Ex : CVT¤%("A") ger varken 65 eller 256*65 då CVT¤% alltid räknar in två tecken. Det andra tecknet är odefinierat.	
ERASE	ERASE 0-"icke numeriskt" raderar allt som finns i minnet. Ex. ERASE 0-L	Felet rättat.
(var)	Om ett program innehåller långa variabelnamn och man gör ERASE ln1-ln2 och sparar programmet, finns de långa variabelnamnen kvar som låg på de borttagna raderna. Åtgärd: Spara programmet med LIST och hämta in det igen.	
ERROR	Vissa felmeddelanden ges två gånger, t.ex. Error 39.	Felet rättat.
FOR	Single-step. Ex: 10 FOR I=0 TO 1000 20 ;I; 30 NEXT I Använder man single-step i denna FOR-loop "ramlar" BASIC-en ur.	Felet rättat.
GOTO GOSUB	Flerradiga funktioner med GOSUB, GOTO, som refererar till rader utanför funktionen, kan ge "dyk" i vissa fall.	Felet rättat. Ej tillåtet att referera till radnr utanför funktionen.

	BASIC II 1.1	BASIC II 1.12
INPUT	Vid INPUT matas talet in med "rätt" noggrannhet och avrundas. Vid LET däremot matas talet alltid in med DOUBLE och trunkeras till önskad längd. Ex: SINGLE 10 A.=.1 20 INPUT B. 30 IF A.=B. THEN PRINT "A=B" ELSE PRINT "A<>B" RUN ? .1 A<>B ABC800	Felet rättat.
KILL	KILL ´filnamn´ på skrivskyddad skiva ger ej felmeddelande.	Felet rättat.
MID¤	MID¤(...¤,X,Y)=´ABC800´ ger felaktiga felmeddelanden om "..." är ett reserverat ord. Ex: MID¤(TAB¤,X,Y)=´OLLE´ ger Error 36.	
NUM¤	DOUBLE PRINT NUM¤(7.057) Ger felaktig utskrift.	Felet rättat.
ON ERROR	I samband med ERROR-hantering och flerradiga funktioner kan "dyk" ske vid felaktigt uthopp ur funktionen. Se GOTO/GOSUB.	Felet rättat.
SAVE	SAVE filnamn Ger Error 41 vid öppen lucka. Ska vara Error 42. Ger Error 41 vid skrivskyddad skiva och filen ej finns på skivan. Ska vara Error 43. Ger Error 41 om skivan är full och det finns en skiva i drive 1.	Felet rättat.

	BASIC II 1.1	BASIC II 1.12
SINGLE STEP	Ex: 10 REM 20 PRINT ´text´ 30 GOTO 20 Om programmet körs med single step och avbryts med CTRL-C när rad 20 skall exekveras, får man STOP IN LINE 10	Felet rättat.
TAB	Utskrift m.h.a. TAB( ) på CON: fungerar ej riktigt Ex: 10 OPEN ´CON:´ AS FILE 1 20 PRINT #1 TAB(10)´ABC´;TAB(20) ´DATA´ 30 CLOSE 1 Detta ger utskriften "ABC" i pos. 10 men "DATA" i pos. 30.	Felet rättat.
TRACE	TRACE ger fel utskrift vid radnummer större än 32767 (heltalsutskrift).	Felet rättat.
	I kommandomode: TRACE #1 NOTRACE Detta ger Error 32.	Felet rättat.
TRIG	Trigonometriska funktioner ger: Fel värde vid indata större än 4096*2*PI. Errormeddelande vid indata större än 8192*2*PI.	

	BASIC II 1.1	BASIC II 1.12
TAB	Tab position vänsterjusteras ett steg vid utskrift, fr.o.m andra inläsningen.	Felet rättat.

Ex:

```
10 PREPARE ´TEST.TXT´ AS FILE 1
20 ;#1,TAB(20) ´ABC´
30 ;#1,TAB(20) ´800´
40 ;#1,TAB(20) ´DATOR´
50 CLOSE 1
60 OPEN ´TEST.TXT´ AS FILE 1
70 ON ERROR GOTO 130
80 WHILE -1
90   INPUT LINE #1,A¤
100  A¤=LEFT¤(A¤,LEN(A¤)-2)
110  ;A¤
120 WEND
130 CLOSE 1
```

BILAGA 2 PROGRAMLISTNINGAR

På följande sidor är nedanstående program och exempel listade.

Kapitel	Program/funktion
2.3.3	BINSÖK
2.3.4	INMATA UPPDATERA BORT LISTA
3.2.1	FILL INSERT DELETE
4.2	CRTADJ
4.3.3	ANIMERING
4.3.4	HRGET HRPUT HRLOAD HRSAVE HRERASE
5.3.3	EXTBAS
ÖVRIGT	CHAIN OPEN PREPARE CUR¤ CURPOS ERROR GET INPUT MESSAGE SHIFT¤

```
47000 ! ---------------------------------------
47001 DEF FNBinsök(Söktext¤,Min,Max) LOCAL Mitten
47002   ! ---------------------------------------
47003   ! Söker efter en sträng med binärsökning
47004   ! Sökvektorn skall vara sorterad !!!
47005   ! Minsta index i sökvektorn skall vara ett (1)
47006   ! Rekursiv variant
47007   !
47008   ! IN:
47009   ! -   PARAMETRAR:     Söktext¤  - Text som skall sökas
47010   ! -                   Min       - Minsta index i sökvek-
        torn
47011   ! -                   Max       - Största index i sökvek-
        torn
47012   ! -   GLOBALA         Sträng¤() - Sökvektor
47013   !
47014   ! UT:
47015   !
47016   ! -   FUNKTIONSVÄRDE:     0     - Texten fanns ej i sök-
        vektorn
47017   ! -                     <> 0    - Index för funnen text i
        sökvektorn
47018   ! -   GLOBALA:           ---
47019   !
47020   ! LOKALA VARIABLER:
47021   !
47022   ! Mitten -  Index för aktuell text vid sökning
47023   !
47024   ! ANVÄNDNA FUNKTIONER:
47025   ! FNBinsök()
47026   !
47027   ! ÖVRIGT:
47028   !
47029   ! ---
47030   ! ---------------------------------------
47031   !
47032   ! Tag ut aktuellt index för sökvektorn
47033   Mitten=(Min+Max)/2
47034   !
47035   ! Är sökvektor lika med söktexten så returnera index till
        sökvektorn
47036   IF Sträng¤(Mitten)=Söktext¤ THEN RETURN Mitten
47037   !
47038   ! Är texten ej funnen så returnera 0
47039   IF Mitten=Max THEN RETURN 0
47040   !
47041   ! Är söktexten i övre halvan av sökvektorn så rekursera med
        Min= Mitten+1
47042   IF Söktext¤>Sträng¤(Mitten) THEN RETURN FNBinsök(Söktext¤,
        Mitten+1,Max)
47043   !
47044   ! Söktexten är i undre halvan av sökvektorn. Rekursera med Max=
        Mitten-1
47045   RETURN FNBinsök(Söktext¤,Min,Mitten-1)
47046 FNEND
```

```
1000  ! EXEMPEL PÅ ISAM ANVÄNDING
1010  ! INMATNING
1020  !
1030  INTEGER : EXTEND : DOUBLE : OPTION BASE 0
1040  !
1050  ! Dimensionera variabler
1060  DIM Post¤=61,Artnr¤=15,Benämn¤=20
1070  !
1080  ! Öppna ISAM fil (OBS Vi öppnar nyckelfilen)
1090  ISAM OPEN "ARTIKLAR.ISM" AS FILE 1
1100  !
1110  ! Läs poster tills ARTIKELNR=""
1120  INPUT "ARTIKELNUMMER: "Artnr¤
1130  WHILE Artnr¤<>""
1140    INPUT "BENÄMNING    : "Benämn¤
1150    INPUT "SALDO        : "Saldo
1160    INPUT "IN-PRIS      : "Ipris.
1170    INPUT "UT-PRIS      : "Upris.
1180    INPUT "FÖRSÄLJNG    : "Förs.
1190    !
1200    ! Fyll ut fält till maximal längd
1210    Artnr¤=Artnr¤+SPACE¤(15-LEN(Artnr¤))
1220    Benämn¤=Benämn¤+SPACE¤(20-LEN(Benämn¤))
1230    !
1240    ! Skapa postvariabel
1250    Post¤=Artnr¤+Benämn¤+CVT%¤(Saldo)+CVTF¤(Ipris.)+CVTF¤(Upris.)+
         CVTF¤(Förs.)
1260    !
1270    ! Skriv ut post i ISAM-fil
1280    ISAM WRITE #1,Post¤
1290    !
1300    ! Läs nytt artikelnummer
1310    INPUT "ARTIKELNUMMER: "Artnr¤
1320  WEND
1330  !
1340  ! Stäng ISAM fil
1350  CLOSE 1
1360  !
1370  END
```

```
1000  ! EXEMPEL PÅ ISAM ANVÄNDING
1010  ! UPPDATERING
1020  !
1030  INTEGER : EXTEND : DOUBLE : OPTION BASE 0
1040  !
1050  ! Dimensionera variabler
1060  DIM Post¤=61,Newpost¤=61,Artnr¤=15,Benämn¤=20
1070  !
1080  ! Öppna ISAM fil (OBS Vi öppnar nyckelfilen)
1090  ISAM OPEN "ARTIKLAR.ISM" AS FILE 1
1100  !
1110  ! Läs post tills ARTIKELNR=""
1120  INPUT "ARTIKELNUMMER: "Artnr¤
1130  WHILE Artnr¤<>""
1140    !
1150    ! Sök och läs in aktuell post
1160    ISAM READ #1,Post¤ INDEX "ARTNR" KEY Artnr¤
1170    !
1180    ! Dela upp Post¤ i respektive delar
1190    Artnr¤=MID¤(Post¤,1,15)
1200    Benämn¤=MID¤(Post¤,16,20)
1210    Saldo=CVT¤%(MID¤(Post¤,36,2))
1220    Ipris.=CVT¤F(MID¤(Post¤,38,8))
1230    Upris.=CVT¤F(MID¤(Post¤,46,8))
1240    Förs.=CVT¤F(MID¤(Post¤,54,8))
1250    !
1260    ! Skriv ut postinnehåll
1270    ; "ARTIKELNUMMER: " Artnr¤
1280    ; "BENÄMNING    : " Benämn¤
1290    ; "SALDO        : " Saldo
1300    ; "IN-PRIS      : " Ipris.
1310    ; "UT-PRIS      : " Upris.
1320    ; "FÖRSÄLJNG    : " Förs.
1330    !
1340    ! Läs lageruttag
1350    INPUT "UTTAG    : "Ut
1360    !
1370    ! Uppdatera post
1380    Saldo=Saldo-Ut
1390    Förs.=Förs.+Ut*Upris.
1400    !
1410    ! Skapa postvariabel
1420    Newpost¤=Artnr¤+Benämn¤+CVT%¤(Saldo)+CVTF¤(Ipris.)+
        CVTF¤(Upris.)+CVTF¤(Förs.)
1430    !
1440    ! Uppdater  post i ISAM-fil
1450    ISAM UPDATE #1,Post¤ TO Newpost¤
1460    !
1470    ! Läs nytt artikelnummer
1480    INPUT "ARTIKELNUMMER: "Artnr¤
1490  WEND
1500  !
1510  ! Stäng ISAM fil
1520  CLOSE 1
1530  !
1540  END
```

BILAGA 2 BORT

```
1000  ! EXEMPEL PÅ ISAM ANVÄNDING
1010  ! BORTTAGNING
1020  !
1030  INTEGER : EXTEND : DOUBLE : OPTION BASE 0
1040  !
1050  ! Dimensionera variabler
1060  DIM Post¤=61,Newpost¤=61,Artnr¤=15,Benämn¤=20
1070  !
1080  ! Öppna ISAM fil (OBS Vi öppnar nyckelfilen)
1090  ISAM OPEN "ARTIKLAR.ISM" AS FILE 1
1100  !
1110  ! Läs post tills ARTIKELNR=""
1120  INPUT "ARTIKELNUMMER: "Artnr¤
1130  WHILE Artnr¤<>""
1140    !
1150    ! Sök och läs in aktuell post
1160    ISAM READ #1,Post¤ INDEX "ARTNR" KEY Artnr¤
1170    !
1180    ! Dela upp Post¤ i respektive delar
1190    Artnr¤=MID¤(Post¤,1,15)
1200    Benämn¤=MID¤(Post¤,16,20)
1210    Saldo=CVT¤%(MID¤(Post¤,36,2))
1220    Ipris.=CVT¤F(MID¤(Post¤,38,8))
1230    Upris.=CVT¤F(MID¤(Post¤,46,8))
1240    Förs.=CVT¤F(MID¤(Post¤,54,8))
1250    !
1260    ! Skriv ut postinnehåll
1270    ; "ARTIKELNUMMER: " Artnr¤
1280    ; "BENÄMNING    : " Benämn¤
1290    ; "SALDO        : " Saldo
1300    ; "IN-PRIS      : " Ipris.
1310    ; "UT-PRIS      : " Upris.
1320    ; "FÖRSÄLJNG    : " Förs.
1330    !
1340    ! Läs om posten skall tas bort
1350    INPUT "Skall denna post raderas (J/N) ? "Svar¤
1360    !
1370    IF (ASCII(Svar¤) OR 32)=110 THEN 1420
1380    !
1390    ! Radera  post i ISAM-fil
1400    ISAM DELETE #1,Post¤
1410    !
1420    ! ENDIF
1430    ! Läs nytt artikelnummer
1440    INPUT "ARTIKELNUMMER: "Artnr¤
1450  WEND
1460  !
1470  ! Stäng ISAM fil
1480  CLOSE 1
1490  !
1500  END
```

BILAGA 2 LISTA

```
1000  ! EXEMPEL PÅ ISAM ANVÄNDING
1010  ! LAGERLISTA
1020  !
1030  INTEGER : EXTEND : DOUBLE : OPTION BASE 0
1040  !
1050  ! Dimensionera variabler
1060  DIM Post¤=61,Newpost¤=61,Artnr¤=15,Benämn¤=20
1070  !
1080  ! Öppna ISAM fil (OBS Vi öppnar nyckelfilen)
1090  ISAM OPEN "ARTIKLAR.ISM" AS FILE 1
1100  !
1110  ! Skriv överskrift
1120  ; "LAGERLISTA"
1130  ;
1140  ; "ARTNR" TAB(15) "BENÄMNING" TAB(35) "SALDO" TAB(45) "IN-PRIS"
      TAB(55) "UT-PRIS" TAB(65) "FÖRSÄLJNING"
1150  ; STRING¤(80,95)
1160  ;
1170  ! Läs lagerlista tills EOF
1180  ON ERROR GOTO 1420
1190  ISAM READ #1,Post¤ INDEX "ARTNR" FIRST
1200  WHILE -1
1210    !
1220    ! Dela upp Post¤ i respektive delar
1230    Artnr¤=MID¤(Post¤,1,15)
1240    Benämn¤=MID¤(Post¤,16,20)
1250    Saldo=CVT¤%(MID¤(Post¤,36,2))
1260    Ipris.=CVT¤F(MID¤(Post¤,38,8))
1270    Upris.=CVT¤F(MID¤(Post¤,46,8))
1280    Förs.=CVT¤F(MID¤(Post¤,54,8))
1290    !
1300    ! Skriv ut postinnehåll
1310    ; Artnr¤;
1320    ; TAB(16) Benämn¤;
1330    ; USING "&####" TAB(36) Saldo;
1340    ; USING "&######.##" TAB(45) Ipris.;
1350    ; USING "&######.##" TAB(55) Upris.;
1360    ; USING "&######.##" TAB(65) Förs.
1370    !
1380    ! Summera försäljning
1390    Sum.=Sum.+Förs.
1400    ISAM READ #1,Post¤ INDEX "ARTNR" NEXT
1410  WEND
1420  !
1430  ! Skriv ut total försäljning
1440  ; USING "&&######.##" TAB(56) "SUMMA = " TAB(65) Sum.
1450  !
1460  ! Stäng ISAM fil
1470  CLOSE 1
1480  !
1490  END
```

```
50000 ! ----------------------------------------
50001 DEF FNFill(Pos,Antal,Tkn) LOCAL Dummy,Code¤=14
50002   ! ----------------------------------------
50003   !
50004   ! Fyll en sträng med antal st tecken med ASCII-värdet tkn
50005   ! från postion Pos. Ifall Pos+Antal överskrider aktuell
50006   ! längd ökas aktuell längd med Pos+Antal
50007   !
50008   ! IN
50009   !
50010   ! Pos         -  Startposition i strängen S¤
50011   ! Antal       -  Antal tecken som skall fyllas
50012   ! Tkn         -  ASCII värdet för det tecken som S¤ skall
50013   !                fyllas med
50014   ! UT
50015   !
50016   ! 0           -  Allt gick bra
50017   ! <>0         -  Fel. Ec returneras
50018   !
50019   ! GLOBALA VARIABLER
50020   !
50021   ! S¤          -  Den globala sträng som fylls
50022   !
50023   ! ÖVRIGT
50024   !
50025   ! Code¤ innehåller följande assemblerrutin
50026   !
50027   ! LD      BC,Antal
50028   ! LD      HL,Pos
50029   ! LD      (HL),Tkn
50030   ! LD      A,B
50031   ! OR      C
50032   ! RET     Z
50033   ! LDIR
50034   ! RET
50035   !
50036   ! ----------------------------------------
50037   !
50038   ! Kontrollera att dimensioneringar och antal tkn ej
          över/under
50039   ! skrider tillåtna värden
50040   IF Antal<0 THEN Ec=135 : RETURN Ec
50041   IF Pos+Antal-1>PEEK2(VAROOT(S¤)) THEN Ec=137 : RETURN Ec
50042   !
50043   ! Tilldela Code¤ en assemblerrutin
50044   Code¤=CHR¤(1)+CVT%¤(Antal-1)+CHR¤(33)+CVT%¤(VARPTR(S¤)+
      Pos-1)+CHR¤(54,Tkn,120,177,200,237,176,201)
50045   !
50046   ! Anropa assemblerrutinen som ligger i Code¤, inparameter
50047   ! är DEreg som innehåller adressen till startpositionen+1
50048   Dummy=CALL(VARPTR(Code¤),VARPTR(S¤)+Pos)
50049   !
50050   ! Kontrollera om aktuell längd har överskridits. Isåfall
50051   ! öka aktuell längd
50052   IF Pos+Antal-1>LEN(S¤) THEN POKE VAROOT(S¤)+4,Pos+Antal
      -1,SWAP%(Pos+Antal-1)
50053   !
```

```
50054   ! Återgå med värdet 0 som indikerar att inga fel uppstod
50055   RETURN 0
50056 FNEND
```

```
50200 ! ----------------------------------------
50201 DEF FNInsert(Pos,Str¤) LOCAL Dummy,L,Code¤=9
50202   ! ----------------------------------------
50203   !
50204   ! Skjut in strängen <Str¤> i strängen <S¤> från position
      <Pos>
50205   ! Aktuell längd ökas med längden av <Str¤>.
50206   !
50207   ! IN
50208   !
50209   ! Pos     -  Startposition i strängen S¤
50210   ! Str¤    -  Sträng som skall insättas
50211   !
50212   ! UT
50213   !
50214   ! 0       -  Allt gick bra
50215   ! <>0     -  Fel. Ec returneras
50216   !
50217   ! GLOBALA VARIABLER
50218   !
50219   ! S¤      -  Arbetssträng
50220   !
50221   ! ÖVRIGT
50222   !
50223   ! Code¤ innehåller följande assemblerrutin
50224   !
50225   ! LD   BC,Antal
50226   ! LD   HL,LEN(S¤)
50227   ! LDIR
50228   ! RET
50229   !
50230   ! ----------------------------------------
50231   !
50232   ! Kontrollera att dimensionerad längd ej överskrids, iså-
      fall
50233   ! återgå med Ec satt till error
50234   IF (LEN(S¤)+LEN(Str¤)>PEEK2(VAROOT(S¤))) OR
      (Pos+LEN( Str¤)-1> PEEK2(VAROOT(S¤))) THEN Ec=137 :
      RETURN Ec
50235   !
50236   ! Ifall ´insert strängen´ har längd 0 återgå med 0, Allt
      bra
50237   IF LEN(Str¤)=0 THEN RETURN 0
50238   !
50239   ! Ifall ´insert strängen´ hamnar efter aktuell längd
50240   ! behövs ingen isärflyttning
50241   IF LEN(S¤)-Pos+1<1 THEN 50251
50242   !
50243   ! Tilldela Code¤ en assemblerrutin
50244   Code¤=CHR¤(1)+CVT%¤(LEN(S¤)-Pos+1)+CHR¤(33)+CVT%¤(VARPTR
      (S¤)+ LEN(S¤)-1)+CHR¤(237,184,201)
50245   !
50246   ! Anropa assemblerrutinen i Code¤, inparameter är DE
50247   ! som innehåller adressen till ´nya´ positionen
50248   Dummy=CALL(VARPTR(Code¤),VARPTR(S¤)+LEN(Str¤)+LEN(S¤)-1)
50249   !
50250   ! Öka aktuell längd med längden av Str¤
50251   L=LEN(S¤)
50252   IF Pos>L+1 THEN L=Pos-1
```

```
50253   POKE VAROOT(S¤)+4,L+LEN(Str¤),SWAP%(L+LEN(Str¤))
50254   !
50255   ! Stoppa in Str¤ i S¤
50256   MID¤(S¤,Pos,LEN(Str¤))=Str¤
50257   !
50258   ! Återgå med 0 , Allt gick bra
50259   RETURN 0
50260   FNEND
```

BILAGA 2 DELETE

```
50100 ! ----------------------------------------
50101 DEF FNDelete(Pos,Antal) LOCAL Dummy,L,Code¤=12
50102   ! ----------------------------------------
50103   !
50104   ! Ta bort <Antal> tecken från position <Pos>. Kvarvarande
50105   ! delar skjuts ihop och aktuell längd minskas med <Antal>
50106   !
50107   ! IN
50108   !
50109   ! Pos       -  Startposition i strängen S¤
50110   ! Antal     -  Antal tecken som skall tas bort
50111   !
50112   ! UT
50113   !
50114   ! 0         -  Allt gick bra
50115   ! <>0       -  Fel. Ec returneras
50116   !
50117   ! GLOBALA VARIABLER
50118   !
50119   ! S¤        -  Global sträng där borttagning sker
50120   !
50121   ! ÖVRIGT
50122   !
50123   ! Code¤ innehåller följande assemblerrutin
50124   !
50125   ! LD    BC,LEN(S¤)-Pos-Antal+1
50126   ! LD    HL,VARPTR(S¤)+Pos+Antal
50127   ! LD    A,B
50128   ! OR    C
50129   ! RET   Z
50130   ! LDIR
50131   ! RET
50132   !
50133   ! ----------------------------------------
50134   !
50135   ! Kontrollera att aktuell längd inte överskrids eller att
        antal> 0
50136   IF (Pos>LEN(S¤)) OR (Pos+Antal-1>LEN(S¤)) THEN Ec=134 :
        RETURN Ec
50137   IF Antal<0 THEN Ec=135 : RETURN Ec
50138   !
50139   ! Tilldela Code¤ en assemblerrutin
50140   Code¤=CHR¤(1)+CVT%¤(LEN(S¤)-Pos-Antal+1)+CHR¤(33)+CVT%¤
        (VARPTR(S¤)+Pos+Antal-1)+CHR¤(120,177,200,237,176,201)
50141   !
50142   ! Anropa assemblerrutinen i Code¤, inparameter är DE
50143   ! som innehåller adressen till första ´delete position´
50144   Dummy=CALL(VARPTR(Code¤),VARPTR(S¤)+Pos-1)
50145   !
50146   ! Minska aktuell längd med det antal tecken som tagits
        bort
50147   L=LEN(S¤)
50148   POKE VAROOT(S¤)+4,L-Antal,SWAP%(L-Antal)
50149   !
50150   ! Återgå med 0 som indikerar att allt gick bra
50151   RETURN 0
50152 FNEND
```

BILAGA 2 CRTADJ

```
10000 ! ******************************************************
10010 ! *                                                    *
10020 ! *                      CRTADJ                        *
10030 ! *                                                    *
10040 ! ******************************************************
10050 !
10060 ! Bildskärmsjustering        NANCO Elektonik  820926
10070 !
10080 Vert=4
10090 Hor=100
10100 Minvert=0
10110 Maxvert=31
10120 Minhor=80
10130 Maxhor=120
10140 !
10150 ; CUR(0,0) STRING¤(80,127)
10160 FOR I=1 TO 22
10170   ; CUR(I,0) ´ TAB(80) ´
10180 NEXT I
10190 ; STRING¤(80,127);
10200 ; CUR(3,28) ´Bildskärmsjustering´
10210 ; CUR(4,28) ´==================´
10220 ; CUR(8,38) ´Upp´
10230 ; CUR(9,38) ´PF5´
10240 ; CUR(10,25) ´Vänster                Höger´
10250 ; CUR(11,25) ´<--                     -->´
10260 ; CUR(12,38) ´PF7´
10270 ; CUR(13,38) ´Ner´
10280 ; CUR(20,1) STRING¤(78,ASCII(´=´))
10290 ; CUR(21,5) ´<--   =   Vänster        PF5   =   Upp
      RETURN  =  Avsluta´
10300 ; CUR(22 5) ´-->   =   Höger          PF7   =   Ner´
10310 !
10320 WHILE 1
10330   OUT 56,2,57,Hor,56,5,57,Vert
10340   ; CUR(22,65) ´Välj :´;
10350   GET A¤
10360   Z=INSTR(1,CHR¤(13,8,9,196,198),A¤)
10370   IF Z=0 THEN 10340
10380   IF Z=1 THEN 10440
10390   IF Z=2 IF Hor<Maxhor Hor=Hor+1 ELSE PUT CHR¤(7)
10400   IF Z=3 IF Hor>Minhor Hor=Hor-1 ELSE PUT CHR¤(7)
10410   IF Z=4 IF Vert>Minvert Vert=Vert-1 ELSE PUT CHR¤(7)
10420   IF Z=5 IF Vert<Maxvert Vert=Vert+1 ELSE PUT CHR¤(7)
10430 WEND
10440 ; CHR¤(12) ´Bildskärmsjustering avslutad´
10450 END
```

BILAGA 2 ANIMERING

```
10000 ! ANIMERING
10010 !
10020 ! Ett litet programexempel som flyttar en bild över skärmen
10030 !
10040 !
10050 DEF FNBakgrund
10060   !
10070   ! FNBakgrund skriver ut slumpvis valda punkter (stjärnor)
10080   ! på skärmen med färg 3 (visas i båda FGCTL moderna).
10090   !
10100   FGPOINT 23,56,3
10110   FGPOINT 68,88
10120   FGPOINT 210,180
10130   FGPOINT 120,211
10140   FGPOINT 230,5
10150   FGPOINT 230,50
10160   FGPOINT 120,140
10170   FGPOINT 50,123
10180   FGPOINT 3,190
10190   FGPOINT 170,210
10200   FGPOINT 150,40
10210   FGPOINT 10,34
10220   FGPOINT 45,30
10230   FGPOINT 50,230
10240   RETURN 0
10250 FNEND
10260 DEF FNShape(X,Y,Färg)
10270   !
10280   ! FNShape ritar ut "raketen" med startposition X,Y
10290   ! och färg  Färg.
10300   !
10310   FGPOINT X,Y+3,Färg
10320   FGLINE X+1,Y+4
10330   FGLINE X+2,Y+4
10340   FGLINE X+8,Y+10
10350   FGLINE X+10,Y+10
10360   FGLINE X+10,Y+8
10370   FGLINE X+4,Y+2
10380   FGLINE X+4,Y+1
10390   FGLINE X+3,Y
10400   FGLINE X,Y+3
10410   !
10420   ! Vi ritar olika lång "svans" beroende på vilken
10430   ! "raketen" ritas med.
10440   !
10450   IF Färg>500 THEN 10530
10460   FGPOINT X+1,Y+2
10470   FGLINE X-2,Y-1
10480   FGLINE X-2,Y-2
10490   FGLINE X-1,Y-2
10500   FGLINE X+2,Y+1
10510   GOTO 10580
10520   !
10530   FGPOINT X+1,Y+2
10540   FGLINE X-4,Y-3
10550   FGLINE X-4,Y-4
10560   FGLINE X-3,Y-4
```

```
10570   FGLINE X+2,Y+1
10580   RETURN 0
10590 FNEND
10600 !
10610 DEF FNHrclear¤
10620   !
10630   ! FNHrclear nollställer HR-minne och VDU-minne
10640   !
10650   FGPOINT 0,0,0
10660   FGFILL 239,239
10670   ; CHR¤(12);
10680   RETURN ""
10690 FNEND
10700 !
10710 ! >> huvudprogram
10720 !
10730 ; FNHrclear¤; ! Nollställ skärm
10740 !
10750 ! rita bakgrund med färg 3 som visas i båda FGCTL moderna
10760 A=FNBakgrund
10770 ! Rita om bilden 96 ggr med förflyttning av "raketen"
10780 FOR I=10 TO 200
10790   FGCTL 73 ! visa färg 2
10800   A=FNShape(I-1,I-1,512) ! radera gammal bild och skydda
        färg 2
10810   A=FNShape(I+1,I+1,513) ! rita ny bild med färg 1 och
        skydda färg 2
10820   I=I+1
10830   FGCTL 72 ! visa färg 1
10840   A=FNShape(I-1,I-1,256) ! radera gammal bild och skydda
        färg 1
10850   A=FNShape(I+1,I+1,258) ! rita ny bild med färg 2 och
        skydda färg 1
10860 NEXT I
10870 END
```

```
50300 ! ------------------------------------------------------
50301 DEF FNHrget(Filnamn¤) LOCAL Adr,Nr,Dummy,Code¤=10,Graf¤=256
50302   ! ------------------------------------------------------
50303   !
50304   ! FNHrget hämtar en bild från valfri fil och laddar
50305   ! HR-minnet med den. Bilden/filen bör vara skapad med
50306   ! FNHrput. Rutinen överför 256 byte i taget för att
50307   ! spara plats.
50308   !
50309   ! IN
50310   !
50311   ! Filnamn¤   -  Namn på vilken fil HR-bilden skall hämtas
                       ifrån
50312   !
50313   ! UT
50314   !
50315   ! = 0        -  Allt gick bra
50316   ! <>0        -  Fel. Ec returneras
50317   !
50318   ! GLOBALA VARIABLER
50319   !
50320   ! ---
50321   !
50322   ! ÖVRIGT
50323   !
50324   ! HR-option måste vara ansluten
50325   !
50326   ! Code¤ innehåller följande assemblerrutin
50327   !
50328   ! LD   HL,Adr*64
50329   ! LD   BC,256
50330   ! EX   DE,HL
50331   ! JP   32765
50332   !
50333   ! ------------------------------------------------------
50334   !
50335   !
50336   ! Öppna infilen. Ifall fel uppstod returnera felkod
50337   Nr=99
50338   IF FNOpen(Filnamn¤,Nr) THEN RETURN Ec
50339   !
50340   !
50341   ! Sätt startadressen för HR-minnet till 0
50342   Adr=0
50343   !
50344   ! Sätt ON ERROR GOTO ifall fel uppstår under överföringen
50345   ON ERROR GOTO 50369
50346   !
50347   ! Upprepa överföringen tills vi har nått rad 240 (239)
50348   WHILE Adr<240
50349     !
50350     ! Tilldela Code¤ den assemblerrutin som överför 256 byte
50351 Code¤=CHR¤(33)+CVT%¤(Adr*64)+CHR¤(1,0,1,235,195,253,127)
50352     !
50353     ! Hämta 256 byte av bilden från infil
50354     GET #Nr,Graf¤ COUNT 256
50355     !
```

```
50356      ! Överför 256 byte till HR-minnet
50357      Dummy=CALL(VARPTR(Code¤),VARPTR(Graf¤))
50358      !
50359      ! Öka adressräknaren med 4
50360      Adr=Adr+4
50361    WEND
50362    !
50363    ! Stäng infilen och returnera 0. Allt gick bra
50364    CLOSE Nr
50365    RETURN 0
50366    ! FELRUTIN
50367    !
50368    ! Fel uppstod under överföringen. Returnera felkod.
50369    Ec=ERRCODE
50370    RETURN Ec
50371 FNEND
```

```
50400 ! ------------------------------------------------------
50401 DEF FNHrput(Filnamn¤) LOCAL Adr,Dummy,Nr,Code¤=9,Graf¤=256
50402   ! ------------------------------------------------------
50403   !
50404   ! FNHrput sparar HR-minnet på valfri fil. Bilden/filen
50405   ! kan sedan hämtas och åter visas med hjälp av FNHrget.
50406   ! Överför endast 256 byte åt gången för att spara plats.
50407   !
50408   ! IN
50409   !
50410   ! Filnamn¤   -  Namn på vilken fil HR-bilden skall sparas
50411   !
50412   ! UT
50413   !
50414   ! = 0        -  Allt gick bra
50415   ! <>0        -  Fel. Ec returneras
50416   !
50417   ! GLOBALA VARIABLER
50418   !
50419   ! ---
50420   !
50421   ! ÖVRIGT
50422   !
50423   ! HR-option måste vara ansluten
50424   !
50425   ! Code¤ innehåller följande assemblerrutin
50426   !
50427   ! LD   HL,Adr*64
50428   ! LD   BC,256
50429   ! JP   32765
50430   !
50431   ! ------------------------------------------------------
50432   !
50433   ! Skapa utfilen. Ifall fel uppstår returnera felkod
50434   Nr=99
50435   IF FNPrepare(Filnamn¤,Nr) THEN RETURN Ec
50436   !
50437   ! Nollställ den plats där delarna av grafikminnet lagras
      temporärt
50438   Graf¤=STRING¤(256,0)
50439   !
50440   ! Starta överföringen av grafikminnet till fil i grupper
50441   ! om 256 byte. Adr är adressen i HR-minnet som startar
      från 0
50442   Adr=0
50443   !
50444   ! Ställ ON ERROR GOTO ifall något fel skulle uppstå under
      överföringen
50445   ON ERROR GOTO 50469
50446   !
50447   ! Upprepa överföringen till vi har nått rad 240 (239) i
      HRminnet
50448   WHILE Adr<240
50449     !
50450     ! Tilldela Code¤ den assemblerrutin som överför 256 byte
50451     Code¤=CHR¤(33)+CVT%¤(Adr*64)+CHR¤(1,0,1,195,253,127)
50452     !
```

```
50453     ! Anropa Code¤. Överför 256 byte
50454     Dummy=CALL(VARPTR(Code¤),VARPTR(Graf¤))
50455     !
50456     ! Spara 256 byte på fil
50457     PUT #Nr,Graf¤
50458     !
50459     ! Öka adressräknaren med 4 (Vi hämtar 4 rader i taget)
50460     Adr=Adr+4
50461   WEND
50462   !
50463   ! Stäng utfilen och returnera 0. Allt gick bra.
50464   CLOSE Nr
50465   RETURN 0
50466   ! FELRUTIN
50467   !
50468   ! Fel uppstod under överföringen. Returnera felkod
50469   Ec=ERRCODE
50470   RETURN Ec
50471 FNEND
```

```
50500 ! ------------------------------------------------------
50501 DEF FNHrload(Filnamn¤) LOCAL Dummy,Nr,Code¤=10,Graf¤=16384
50502   ! ------------------------------------------------------
50503   !
50504   ! Hämta en bild från valfri fil och lägg i HR-minnet.
50505   ! Bilden överförs i ett stycke och därför behövs en lokal
50506   ! variabel med längden 16384 byte.
50507   !
50508   ! IN
50509   !
50510   ! Filnamn¤   -  Namn på vilken fil HR-bilden skall hämtas
                       ifrån
50511   !
50512   ! UT
50513   !
50514   ! = 0        -  Allt gick bra
50515   ! <>0       -  Fel. Ec returneras
50516   !
50517   ! GLOBALA VARIABLER
50518   !
50519   ! ---
50520   !
50521   ! ÖVRIGT
50522   !
50523   ! HR-option måste vara ansluten
50524   !
50525   ! Code¤ innehåller följande assemblerrutin
50526   !
50527   ! LD   HL,0
50528   ! LD   BC,16384
50529   ! EX   DE,HL
50530   ! JP   32765
50531   !
50532   ! ------------------------------------------------------
50533   !
50534   ! Öppna infil. Om öppningen misslyckas returnera felkod
50535   Nr=99
50536   IF FNOpen(Filnamn¤,Nr) THEN RETURN Ec
50537   !
50538   ! Sätt ON ERROR GOTO ifall överföringen misslyckas
50539   ON ERROR GOTO 50556
50540   !
50541   ! Tilldela Code¤ den assemblerrutin som överför 16384 byte
50542   Code¤=CHR¤(33,0,0,1,0,64,235,195,253,127)
50543   !
50544   ! Läs 16384 byte från infil
50545   GET #Nr,Graf¤ COUNT 16384
50546   !
50547   ! Anropa Code¤ och överför 16384 byte till HR-minnet
50548   Dummy=CALL(VARPTR(Code¤),VARPTR(Graf¤))
50549   !
50550   ! Överföringen gick bra. Stäng infil och returnera 0
50551   CLOSE Nr
50552   RETURN 0
50553   ! FELRUTIN
50554   !
50555   ! Överföringen misslyckades. Returnera felkod
```

```
50556    Ec=ERRCODE
50557    RETURN Ec
50558 FNEND
```

```
50600 ! ------------------------------------------------------
50601 DEF FNHrsave(Filnamn¤) LOCAL Dummy,Nr,Code¤=9,Graf¤=16384
50602   ! ------------------------------------------------------
50603   !
50604   ! Överför HR-minnet till valfri fil. HR-minnet överförs i
50605   ! ett stycke och därför krävs en lokal variabel på 16384
        byte.
50606   !
50607   ! IN
50608   !
50609   ! Filnamn¤   -  Namn på vilken fil HR-bilden skall sparas
                        på
50610   !
50611   ! UT
50612   !
50613   ! = 0        -  Allt gick bra
50614   ! <>0       -  Fel. Ec returneras
50615   !
50616   ! GLOBALA VARIABLER
50617   !
50618   ! ---
50619   !
50620   ! ÖVRIGT
50621   !
50622   ! HR-option måste vara ansluten
50623   !
50624   ! Code¤ innehåller följande assemblerrutin
50625   !
50626   ! LD   HL,0
50627   ! LD   BC,16384
50628   ! JP   32765
50629   !
50630   ! ------------------------------------------------
50631   !
50632   ! Skapa utfil. Ifall fel uppstår returnera Ec
50633   Nr=99
50634   IF FNPrepare(Filnamn¤,Nr) THEN RETURN Ec
50635   !
50639   ! Ställ aktuell längd på Graf¤ till 16384
50640   POKE VAROOT(Graf¤)+4,0,64
50641   !
50642   ! Sätt ON ERROR GOTO ifall fel uppstår under överföringen
50643   ON ERROR GOTO 50660
50644   !
50645   ! Tilldela Code¤ den assemblerrutin som överför 16384 byte
50646   Code¤=CHR¤(33,0,0,1,0,64,195,253,127)
50647   !
50648   ! Anropa Code¤.
50649   Dummy=CALL(VARPTR(Code¤),VARPTR(Graf¤))
50650   !
50651   ! Spara Graf¤ på utfil.
50652   PUT #Nr,Graf¤
50653   !
50654   ! Överföringen gick bra. Returnera 0
50655   CLOSE Nr
50656   RETURN 0
50657   ! FELRUTIN
```

```
50658   !
50659   ! Överföringen misslyckades. Returnera felkod
50660   Ec=ERRCODE
50661   RETURN Ec
50662 FNEND
```

```
50700 ! ------------------------------------------------------
50701 DEF FNHrerase LOCAL Adr,Dummy,Code¤=10,Null¤=256
50702   ! ------------------------------------------------------
50703   !
50704   ! Nollställer HR-minnet. Skillnaden mot denna funktion och
50705   ! metoden FGPOINT 0,0,0 : FGFILL 239,239 är att FNhrerase
50706   ! nollställer HELA HR-minnet, även de delar HR-grafiken ej
50707   ! utnyttjar.
50708   !
50709   ! IN
50710   !
50711   ! - PARAMETRAR       ---
50712   !
50713   ! - GLOBALA          ---
50714   !
50715   ! UT
50716   !
50717   ! - FUNKTIONSVÄRDE     0     -  Nollställningen gick bra
50718   ! -                    <>0   -  Fel uppstod. Felkod retur-
50719   !                               neras.
50720   ! - GLOBALA            Ec    -  Om funktionsvärdet är <>0
50721   ! -                             är Ec satt till ERRCODE
50722   !
50723   ! LOKALA VARIABLER
50724   !
50725   ! Adr      -  Adressräknare i HR-minnet
50726   ! Dummy    -  Dummyvariabel
50727   ! Null¤    -  Konstantsträng som innehåller 256 st CHR¤(0)
50728   ! Code¤    -  Innehåller Assemblerrutinen för överföring
50729   ! -           av 256 byte till HR-minnet
50730   !
50731   ! ANVÄNDA FUNKTIONER
50732   !
50733   ! ---
50734   !
50735   ! ÖVRIGT
50736   !
50737   ! HR-option måste vara ansluten.
50738   !
50739   ! Code¤ innehåller följande assemblerrutin:
50740   !
50741   ! LD    HL,Adr*64
50742   ! LD    BC,256
50743   ! JP    32765
50744   !
50745   !
50746   !
50747   ! ------------------------------------------------------
50748   !
50749   !
50750   ! Sätt ON ERROR GOTO ifall fel uppstår under nollställning
50751   ON ERROR GOTO 50777
50752   !
50753   ! Nollställ Null¤ som utgör ´blanksträng´
50754   Null¤=STRING¤(256,0)
50755   !
50756   ! Sätt startadressen för HR-minnet till 0
```

```
50757   Adr=0
50758   !
50759   ! Upprepa nollställning tills vi har nått rad 256 (255)
50760   WHILE Adr<256
50761     !
50762     ! Tilldela Code¤ den assemblerrutin som överför 256 byte
50763 Code¤=CHR¤(33)+CVT%¤(Adr*64)+CHR¤(1,0,1,235,195,253,127)
50764     !
50765     ! Överför 256 byte till HR-minnet
50766     Dummy=CALL(VARPTR(Code¤),VARPTR(Null¤))
50767     !
50768     ! Öka adressräknaren med 4
50769     Adr=Adr+4
50770   WEND
50771   !
50772   RETURN 0
50773   !
50774   ! FELRUTIN
50775   !
50776   ! Fel uppstod under nollställningen. Returnera felkod.
50777   Ec=ERRCODE
50778   RETURN Ec
50779 FNEND
```

BILAGA 2 EXTBAS

```
10000 ! ********************************************
10010 ! *                                          *
10020 ! *              E X T B A S                 *
10030 ! *                                          *
10040 ! ********************************************
10050 !
10060 ! Länkningsprogram för utvidgad BASIC   NANCO Elektronik
                                              821110
10070 !
10080 ! Höjer botten på BASIC-minnet
10090 POKE 65292,0,129
10100 POKE 65328,0,129
10110 !
10120 ! De nya funktionerna/instruktionerna
10130 POKE 32768,0,0,208,2,36,128,40,128,56,128
10140 POKE 32778,0,0,208,1,20,128,22,128,31,128
10150 POKE 32788,89,128,128,71,69,84,73,84,69,77
10160 POKE 32798,255,128,2,2,33,255,65,128,67,128
10170 POKE 32808,128,80,82,79,67,69,68,85,82,69
10180 POKE 32818,129,87,65,73,84,255,60,128,60,128
10190 POKE 32828,6,1,195,29,0,231,201,231,175,50
10200 POKE 32838,245,255,58,244,255,79,124,181,200,58
10210 POKE 32848,244,255,185,40,247,79,43,24,243,225
10220 POKE 32858,221,33,0,0,221,57,213,235,205,154
10230 POKE 32868,128,33,3,0,175,237,66,48,37,205
10240 POKE 32878,161,128,126,35,102,111,175,237,82,56
10250 POKE 32888,25,205,161,128,35,35,27,123,178,78
10260 POKE 32898,6,0,35,40,3,9,24,244,205,175
10270 POKE 32908,128,205,168,128,24,6,1,0,0,205
10280 POKE 32918,168,128,209,239,221,78,4,221,70,5
10290 POKE 32928,201,221,110,2,221,102,3,201,221,113
10300 POKE 32938,4,221,112,5,201,221,117,2,221,116
10310 POKE 32948,3,201
10320 !
10330 ! Inlänkning av de nya instruktionerna
10340 POKE 32768,PEEK(65405),PEEK(65406)
10350 POKE 65405,0,128
10360 !
10370 ! Inlänkning av den nya funktionen
10380 POKE 32778,PEEK(65407),PEEK(65408)
10390 POKE 65407,10,128
10400 CHAIN "NUL:"
```

```
30300 ! ------------------------------------------------------
30301 DEF FNChain(P¤)
30302   ! ------------------------------------------------------
30303   !
30304   ! CHAIN-ar till filen P¤ om fel uppstår
30305   ! ställs fråga om diskettbyte avbryt
30306   ! returnerar värdet 0
30307   !
30308   ! IN
30309   !
30310   ! - PARAMETRAR    P¤          - Filnamn att anropa
30311   !
30312   ! UT
30313   !
30314   ! - FUNKTIONSVÄRDE            - alla uthopp betyder fel/
                                        avbrutet försök
30315   !
30316   ! ANVÄNDA FUNKTIONER
30317   !
30318   ! FNGet
30319   ! FNError
30320   !
30321   ! ------------------------------------------------------
30322   ON ERROR GOTO 30324
30323   ; CUR(23,FNMessage(´´,0)) ´LADDNING AV PROGRAMMODUL´; :
      CHAIN P¤
30324   ; CUR(23,0) ´Byt programdiskett, tryck RETURN (PF1 avbry-
      ter)´;
30325   IF FNGet=192 RETURN 0 ELSE 30322
30326 FNEND
```

BILAGA 2 OPEN

```
30000 ! ------------------------------------------------------
30001 DEF FNOpen(Fil¤,Filnr)
30002   ! ------------------------------------------------------
30003   !
30004   ! Öppnar en fil
30005   !
30006   ! IN
30007   !
30008   ! - PARAMETRAR    Fil¤       -  Namn på filen som skall
                                        öppnas
30009   ! -               Filnr      -  Filnummer som filen skall
                                        ha
30010   !
30011   ! - GLOBALA       ---
30012   !
30013   ! UT
30014   !
30015   ! - FUNKTIONSVÄRDE          0 -  Öppning gick bra
30016   ! -                       <>0 -  Fel. Ec innehåller felkod
30017   !
30018   ! - GLOBALA       Ec         -  Felkod
30019   !
30020   ! LOKALA VARIABLER
30021   !
30022   ! ---
30023   !
30024   ! ANVÄNDA FUNKTIONER
30025   !
30026   ! ---
30027   !
30028   ! ÖVRIGT
30029   !
30030   ! ---
30031   !
30032   ! ----------------------------------
30033   ON ERROR GOTO 30041
30034   OPEN Fil¤ AS FILE Filnr
30035   !
30036   ! Öppning lyckades
30037   Ec=0
30038   RETURN 0
30039   !
30040   ! Öppningen misslyckades
30041   Ec=ERRCODE
30042   RETURN Ec
30043 FNEND
```

BILAGA 2 PREPARE

```
30100 ! ------------------------------------------------------
30101 DEF FNPrepare(Fil¤,Filnr)
30102   ! ------------------------------------------------------
30103   !
30104   ! Skapar en fil
30105   !
30106   ! IN
30107   !
30108   ! - PARAMETRAR    Fil¤       -  Namn på filen som skall
                                        öppnas
30109   ! -               Filnr      -  Filnummer som filen skall
                                        ha
30110   !
30111   ! - GLOBALA       ---
30112   !
30113   ! UT
30114   !
30115   ! - FUNKTIONSVÄRDE         0 -  Öppning gick bra
30116   ! -                      <>0 -  Fel. Ec innehåller felkod
30117   !
30118   ! - GLOBALA       Ec         -  Felkod
30119   !
30120   ! LOKALA VARIABLER
30121   !
30122   ! ---
30123   !
30124   ! ANVÄNDA FUNKTIONER
30125   !
30126   ! ---
30127   !
30128   ! ÖVRIGT
30129   !
30130   ! ---
30131   !
30132   ! ----------------------------------
30133   ON ERROR GOTO 30141
30134   PREPARE Fil¤ AS FILE Filnr
30135   !
30136   ! Filen skapad
30137   Ec=0
30138   RETURN 0
30139   !
30140   ! Filen kunde ej skapas
30141   Ec=ERRCODE
30142   RETURN Ec
30143 FNEND
```

```
21500 ! ------------------------------------------------------
21501 DEF FNCur¤(Pos)=CUR(Pos,Pos/256)
21502 ! ------------------------------------------------------
21503 !
21504 ! Omvandla inparametern till en cursorposition
21505 !
21506 ! IN
21507 !
21508 ! -   PARAMETRAR         Pos   - Heltal som skall omvandlas
21509 ! -                             till en cursorposition
21510 ! -   GLOBALA            ---
21511 !
21512 ! UT
21513 !
21514 ! -   FUNKTIONSVÄRDE           - Cursorposition
21515 ! -   GLOBALA            ---
21516 !
21517 ! LOKALA VARIABLER
21518 !
21519 ! ---
21520 !
21521 ! ANVÄNDA FUNKTIONER
21522 !
21523 ! ---
21524 !
21525 ! ÖVRIGT
21526 !
21527 ! ---
21528 ! ------------------------------------------------------
```

```
21400 ! ------------------------------------------------------
21401 DEF FNCurpos=SWAP%(PEEK2(65362))
21402 ! ------------------------------------------------------
21403 !
21404 ! Spara aktuell cursor som heltal
21405 !
21406 ! IN
21407 ! -   PARAMETRAR           ---
21408 ! -   GLOBALA              ---
21409 !
21410 ! UT
21411 !
21412 ! -   FUNKTIONSVÄRDE                - Cursorvärde som hel-
                                            tal
21413 ! -   GLOBALA              ---
21414 !
21415 ! LOKALA VARIABLER
21416 !
21417 ! ---
21418 !
21419 ! ANVÄNDA FUNKTIONER
21420 !
21421 ! ---
21422 !
21423 ! ÖVRIGT
21424 !
21425 ! - Cursorvärdet hämtas ur minnesadresserna som innehåller
21426 ! - cursorn
21427 ! ------------------------------------------------------
```

```
21900 ! ------------------------------------------------------
21901 DEF FNError(T¤) LOCAL Text¤=80
21902   ! ------------------------------------------------------
21903   ! Skriver ut felmeddelande
21904   !
21905   ! IN:
21906   !
21907   ! -   PARAMETRAR           T¤   - Felmeddelande
21908   ! -   GLOBALA              ---
21909   !
21910   ! UT:
21911   !
21912   ! -   FUNKTIONSVÄRDE       0    - Alltid
21913   ! -   GLOBALA              ---
21914   !
21915   ! LOKALA VARIABLER
21916   !
21917   ! ---
21918   !
21919   ! ANVÄNDA FUNKTIONER
21920   !
21921   ! FNMessage
21922   !
21923   ! ÖVRIGT
21924   !
21925   ! ---
21926   ! ------------------------------------------------------
21927   !
21928   ! Generera en signal
21929   ; CHR¤(7);
21930   !
21931   Text¤=RED+FLSH+´<´+STDY+T¤+FLSH+´>´+STDY
21932   ! Skriv ut felmeddelande. Kvittera med <CE>
21933   RETURN FNMessage(Text¤,24)
21934 FNEND
```

```
21200 ! ------------------------------------------------------
21201 DEF FNGet LOCAL I¤=1
21202   ! ------------------------------------------------------
21203   !
21204   ! Läser in ett tecken från tangentbordet och omvandlar
21205   ! det till ASCII-värdet
21206   !
21207   ! IN:
21208   !
21209   ! -   PARAMETRAR              ---
21210   ! -   GLOBALA                 ---
21211   !
21212   ! UT:
21213   !
21214   ! -   FUNKTIONSVÄRDE                      - ASCII-värde
                                                   för läst tecken
21215   ! -   GLOBALA                 Cinchar     - ASCII-värde
                                                   för läst tecken
21216   ! -                           Ec          - Felnummer
21217   !
21218   ! LOKALA VARIABLER:
21219   !
21220   ! - I¤     - Tecken från fil
21221   !
21222   ! ANROP:
21223   !
21224   ! ---
21225   !
21226   ! ÖVRIGT
21227   !
21228   ! ---
21229   ! ------------------------------------------------------
21230   !
21231   ! Hämta tecken från filen
21232   GET I¤
21233   !
21234   ! Omvandla tecknet till ASCII-koden
21235   Cinchar=ASCII(I¤)
21236   !
21237   ! Returnera ASCII-koden för tecknet
21238   RETURN Cinchar
21239 FNEND
```

```
20600 ! ------------------------------------------------------
20601 DEF FNInput¤(Fråga¤,Default¤,Längd) LOCAL Ed¤=160,In¤=160,-
      Pos, Inasc,Inflag
20602   ! ------------------------------------------------------
20603   !
20604   ! Skriver ut en ledtext och tar svar på ledtexten
20605   !
20606   ! IN:
20607   !
20608   ! -    PARAMETRAR          Default¤     - Defaultsvar på
                                                 frågan
20609   ! -                        Fråga¤       - Ledtext
20610   ! -                        Längd        - Max.längd på
                                                 svarsfältet
20611   ! -   GLOBALA              ---
20612   !
20613   ! UT:
20614   !
20615   ! -    FUNKTIONSVÄRDE                   - Inmatad sträng
20616   ! -    GLOBALA             ---
20617   !
20618   ! LOKALA VARIBLER:
20619   !
20620   ! - Ed¤      - Innehåller behandlad defaultsträng
20621   ! - In¤      - Inmatade tecken
20622   ! - Inasc    - ASCII-värdet för aktuellt tecken
20623   ! - Pos      - Cursorpositionen efter ledtexten
20624   !
20625   ! ANROP:
20626   !
20627   ! FNCurpos
20628   ! FNCur¤
20629   ! FNGet
20630   ! FNShift¤
20631   !
20632   ! ÖVRIGT:
20633   !
20634   ! - Inflag  - Flagga, om 0 skall uthopp från rutinen ske
20635   ! -                   om -1 första tecken
20636   ! -                   om 1 ej första tecken
20637   ! - Cinchar - Senast inmatade tecken
20638   ! ------------------------------------------------------
20639   !
20640   ! Initiera variabler
20641   Ed¤=Default¤ : Inflag=-1
20642   !
20643   ! Skriv ut ledtexten
20644   ; Fråga¤;
20645   !
20646   ! Spara aktuell cursor
20647   Pos=FNCurpos
20648   !
20649   WHILE Inflag
20650     !
```

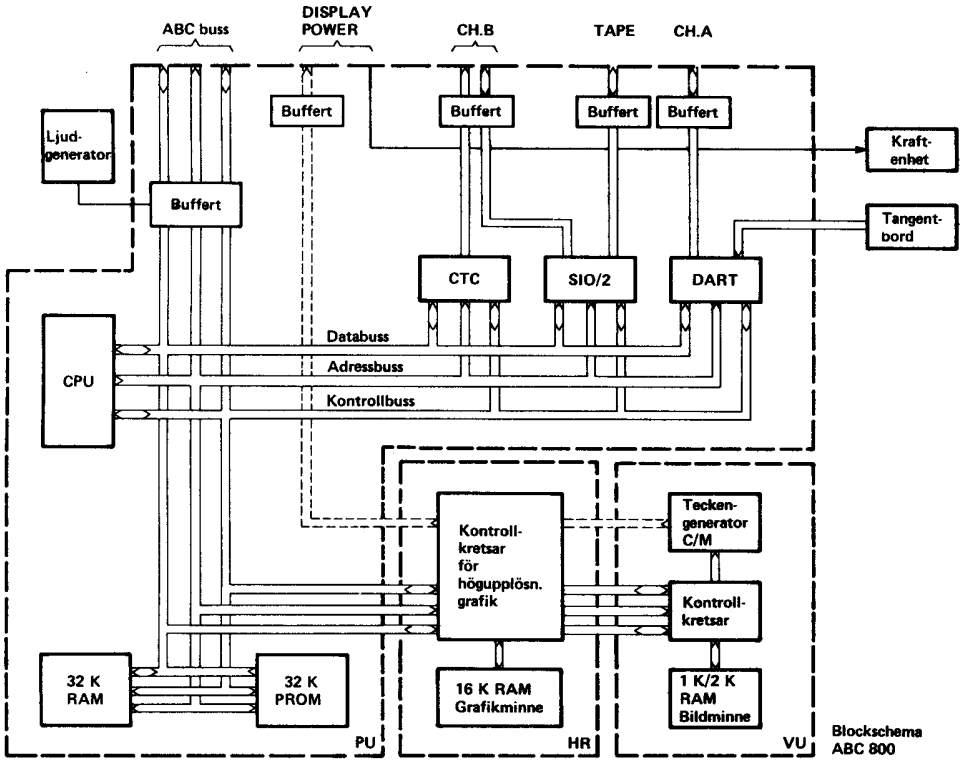
```
20651     ! Skriv ut inmatade tecken och ´_´ till fältet är
20652     ! slut och ställ cursor i rätt inmatningsposition
20653     ; FNCur¤(Pos) In¤ Ed¤ STRING¤(Längd-LEN(In¤)-
          LEN(Ed¤),ASCII(´_´));
20654     ; FNCur¤(Pos) In¤;
20655     !
20656     ! Behandla inmatade tecken
20657     Inasc=FNGet
20658     IF Inasc<32 OR Inasc>127 THEN 20670
20659     !
20660     ! Lagra tecken om det ej maxlängd överskrids
20661     IF Inflag=-1 Ed¤=´´
20662     IF LEN(In¤)<Längd THEN In¤=In¤+CHR¤(Inasc)
20663     !
20664     ! Om max.längd överskrids tas sista tecknet I Ed¤ bort
20665     IF LEN(In¤)+LEN(Ed¤)>Längd THEN Ed¤=LEFT¤(Ed¤,Längd-
          LEN(In¤))
20666     GOTO 20696
20667     !
20668     ! Behandla specialtecken
20669     ! Inasc = 0  =>  Slut på filen
20670     IF Inasc=0 THEN Inflag=0
20671     !
20672     ! Inasc = 8  => Bakåtpil. Flytta cursor ett steg åt
          vänster
20673     IF Inasc=8 THEN Ed¤=FNShift¤(4,In¤)+Ed¤ :
          In¤=FNShift¤(3,In¤)
20674     !
20675     ! Inasc  = 9  => Framåtpil. Flytta cursor ett steg åt
          höger
20676     IF Inasc=9 THEN In¤=In¤+FNShift¤(1,Ed¤) :
          Ed¤=FNShift¤(2,Ed¤)
20677     IF Inasc<>13 THEN 20685
20678     !
20679     ! Inasc = 13  =>  <RETURN>
20680     ! Returnera ALLA tecknen på raden
20681     Inflag=0
20682     In¤=In¤+Ed¤
20683     !
20684     ! Inasc = 24  =>  <CE> eller CTRL-X. Töm raden
20685     IF Inasc=24 THEN In¤=´´ : Ed¤=´´
20686     !
20687     ! Inasc = 199  =>  PF8
20688     ! Tecken finns t.h. om cursor:    Tag bort tecknet när-
          mast t.h. om cursor
20689     ! Tecken finns ej t.h. om cursor: Tag bort tecknet när-
          mast t.v. om cursor
20690     IF Inasc=199 THEN IF LEN(Ed¤) THEN Ed¤=FNShift¤(2,Ed¤)
          ELSE In¤=FNShift¤(3,In¤)
20691     !
20692     ! Inasc > 127  => PF-tangenter
20693     ! Funktionstangeneter (utom PF8) i 1:a pos.  =>  Retur-
20694     ! nering av ALLA tecken på raden
20695     IF ((Inasc>128 AND Inasc<>199) AND LEN(In¤)=0) THEN
          In¤=In¤+ Ed¤ : Inflag=0
20696     Inflag=-(Inflag<>0)
20697   WEND
20698   !
```

```
20699   ! Skriv ut färdiginmatade strängen och fyll resten av
        fältet med blanka
20700   ; FNCur¤(Pos) In¤ SPACE¤(Längd-LEN(In¤));
20701   RETURN In¤
20702 FNEND
```

```
22000 ! ------------------------------------------------------
22001 DEF FNMessage(T¤,Kvittens) LOCAL P
22002   ! ------------------------------------------------------
22003   ! Skriver ut ett meddelande
22004   !
22005   ! IN:
22006   !
22007   ! -   PARAMETRAR         Kvittens    - Vilken tangent som
22008   ! -                                     skall användas för
                                               kvittens
22009   ! -                      T¤          - Meddelande
22010   ! -   GLOBALA            Ccr         - Radbredd (40/80 tkn)
22011   !
22012   ! UT:
22013   !
22014   ! -   FUNKTIONSVÄRDE     0           - Alltid
22015   ! -   GLOBALA            ---
22016   !
22017   ! LOKALA VARIABLER:
22018   !
22019   ! - P       - Cursorpositionen vid inträde i funktionen
22020   !
22021   ! ANVÄNDA FUNKTIONER
22022   !
22023   ! FNCurpos
22024   ! FNGet
22025   ! FNCur¤
22026   !
22027   ! ÖVRIGT
22028   !
22029   ! ---
22030   ! ------------------------------------------------------
22031   !
22032   ! Spara aktuellt värde på cursorn
22033   P=FNCurpos
22034   !
22035   ! Töm meddelanderaden
22036   ; CUR(23,0) SPACE¤(Ccr-1);
22037   !
22038   ! Skriv ut meddelandet med röd text omgiven av blinkande
      röda < och >
22039   ; CUR(23,Ccr/2-LEN(T¤)/2) T¤;
22040   IF Kvittens=0 THEN RETURN 0
22041   WHILE FNGet<>Kvittens
22042     !
22043     ! Vänta till dess att meddelandet är kvitterat
22044   WEND
22045   !
22046   ! Töm meddelanderaden och återställ cursorn
22047   ; CUR(23,0) SPACE¤(Ccr-1) FNCur¤(P);
22048   !
22049   ! Återvänd med värdet 0
22050   RETURN 0
22051 FNEND
```

```
21100 ! ------------------------------------------------------
21101 DEF FNShift¤(Typ,In¤)
21102   ! ------------------------------------------------------
21103   ! Returnerar en delsträng av inströngen
21104   !
21105   ! IN:
21106   !
21107   ! -   PARAMETRAR              Typ   - Anger hur delst-
21108   ! -                                   rängen skall skapas
21109   ! -                           In¤   - Sträng att dela
21110   ! -   GLOBALA                 ---
21111   !
21112   ! UT:
21113   !
21114   ! -   FUNKTIONSVÄRDE                - Delsträngen
21115   ! -   GLOBALA                 ---
21116   !
21117   ! LOKALA VARIABLER
21118   !
21119   ! ---
21120   !
21121   ! ANROP
21122   !
21123   ! ---
21124   !
21125   ! ÖVRIGT
21126   !
21127   ! ---
21128   ! ------------------------------------------------------
21129   !
21130   ! Ingen inströng medskickad medför att delning är omöjlig
21131   IF In¤="" THEN RETURN ""
21132   !
21133   ! Om Typ=1 så returneras 1:a tecknet ur inströngen
21134   IF Typ=1 THEN RETURN LEFT¤(In¤,1)
21135   !
21136   ! Om Typ=2 så returneras alla tecken utom första ur in-
      strängen
21137   IF Typ=2 THEN RETURN RIGHT¤(In¤,2)
21138   !
21139   ! Om Typ=3 så returneras alla tecken utom sista ur in-
      strängen
21140   IF Typ=3 THEN RETURN LEFT¤(In¤,LEN(In¤)-1)
21141   !
21142   ! Om Typ=4 så returneras sista tecknet ur inströngen
21143   IF Typ=4 THEN RETURN RIGHT¤(In¤,LEN(In¤))
21144   !
21145   ! Om Typ<0 eller Typ>4 så returneras hela inströngen
21146   RETURN In¤
21147 FNEND
```

BILAGA 3 Blockschema



BILAGA 4 I/O portar

Reserverade I/O-portadresser på ABC800

IN-PORTAR XINSTB

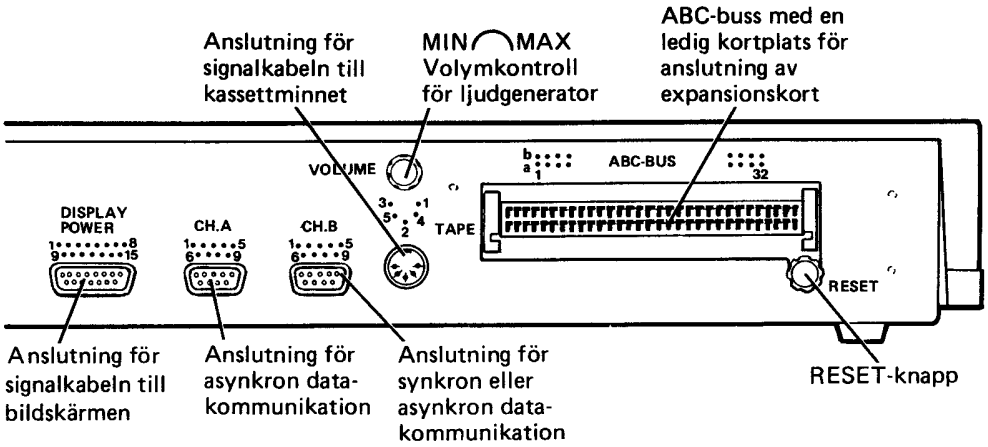
Adress Hex	Dec	Bin 7 6 5 4 3 2 1 0	Enhet (Pin-nr)	Funktion (4680-Benämn.)	Strobe (26A)
00	0	0 0 0 x x 0 0 0	BUS (17A)	(INP 0)	JA
01	1	0 0 0 x x 0 0 1	BUS (16A)	(INP 1)	JA
02	2	0 0 0 x x 0 1 0	BUS (25A)	(INP 2)	JA
03	3	0 0 0 x x 0 1 1	-	-	JA
04	4	0 0 0 x x 1 0 0	-	-	JA
05	5	0 0 0 x x 1 0 1		"PLING"	JA
06	6	0 0 0 x x 1 1 0	-	-	JA
07	7	0 0 0 x x 1 1 1	BUS (15A)	I/O-RESET (RST)	JA
08-1F	8-31	Dubbleringar av ovanstående funktioner			JA
20	32	0 0 1 0 x x 0 0	DART	Tang.bord data	NEJ
22	34	0 0 1 0 x x 1 0	DART	Tang.bord kontr.	NEJ
21	33	0 0 1 0 x x 0 1	DART (CH A)	Printer data	NEJ
23	35	0 0 1 0 x x 1 1	DART (CH A)	Printer kontr.	NEJ
24-2F	36-47	Dubbleringar av ovanstående funktioner			NEJ
30	48	0 0 1 1 0 x x 0	-	-	NEJ
31	49	0 0 1 1 0 x x 1	CRTC 80-tkn	Read Register	NEJ
32-37	50-55	Dubbleringar av ovanstående funktioner			NEJ
38-3F	56-63			Ej använda	NEJ
40	64	0 1 0 x x x 0 0	SIO2 CH B	V24 data	NEJ
41	65	0 1 0 x x x 0 1	SIO2 CH B	V24 kontroll	NEJ
42	66	0 1 0 x x x 1 0	SIO2 CAS	Kassett data	NEJ
43	67	0 1 0 x x x 1 1	SIO2 CAS	Kassett kontroll	NEJ
44-4F	68-95	Dubbleringar av ovanstående funktuioner			NEJ
50	96	0 1 1 x x x 0 0	CTC	Kanal 0	NEJ
51	97	0 1 1 x x x 0 1	CTC	Kanal 1	NEJ
52	98	0 1 1 x x x 1 0	CTC	Kanal 2	NEJ
53	99	0 1 1 x x x 1 1	CTC	Kanal 3	NEJ
54-5F	100-127	Dubbleringar av ovanstående funktioner			NEJ
80-FF	128-255	Lediga, genererar strobe			JA

Reserverade I/O-portadresser på ABC800

UT-PORTAR

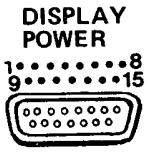
Adress Hex	Dec	Bin 7 6 5 4 3 2 1 0	Enhet (Pin-nr)	Funktion (4680-Benämn.)	XOUTSTB Strobe (27A)
00	0	0 0 0 x x 0 0 0	BUS (22A)	(OUT 0) Out data	JA
01	1	0 0 0 x x 0 0 1	BUS (23A)	(OUT 1) Card sel.	JA
02	2	0 0 0 x x 0 1 0	BUS (21A)	(OUT 2)	JA
03	3	0 0 0 x x 0 1 1	BUS (20A)	(OUT 3)	JA
04	4	0 0 0 x x 1 0 0	BUS (19A)	(OUT 4)	JA
05	5	0 0 0 x x 1 0 1	BUS (18A)	(OUT 5)	JA
06	6	0 0 0 x x 1 1 0	HR/RAM-kort	HRS/RAM-kontr	JA
07	7	0 0 0 x x 1 1 1	HR-minne	HRC=FGCTL kontr.	JA
08-1F	8-31	Dubbleringar av ovanstående funktioner			JA
20	32	0 0 1 0 x x 0 0	DART	(Tang.bord data)	NEJ
22	34	0 0 1 0 x x 1 0	DART	Tang.bord kontr.	NEJ
21	33	0 0 1 0 x x 0 1	DART (CH A)	Printer data	NEJ
23	35	0 0 1 0 x x 1 1	DART (CH A)	Printer kontr.	NEJ
24-2F	36-47	Dubbleringar av ovanstående funktioner			NEJ
30	48	0 0 1 1 0 0 0 0	(RAM-kort	RAM-kontr)	NEJ
31	49	0 0 1 1 0 0 0 1	(RAM-kort	RAM-kontr)	NEJ
32	50	0 0 1 1 0 0 1 0	(RAM-kort	RAM-kontr)	NEJ
33-37	51-55	Används ej			NEJ
38	56	0 0 1 1 1 x x 0	CRTC 80-tkn	Skriv adr reg	NEJ
39	57	0 0 1 1 1 x x 1	CRTC 80-tkn	Skriv reg	NEJ
40	64	0 1 0 x x x 0 0	SIO2 CH B	V24 data	NEJ
41	65	0 1 0 x x x 0 1	SIO2 CH B	V24 kontroll	NEJ
42	66	0 1 0 x x x 1 0	SIO2 CAS	Kassett data	NEJ
43	67	0 1 0 x x x 1 1	SIO2 CAS	Kassett kontroll	NEJ
44-4F	68-95	Dubbleringar av ovanstående funktioner			NEJ
50	96	0 1 1 x x x 0 0	CTC	Kanal 0	NEJ
51	97	0 1 1 x x x 0 1	CTC	Kanal 1	NEJ
52	98	0 1 1 x x x 1 0	CTC	Kanal 2	NEJ
53	99	0 1 1 x x x 1 1	CTC	Kanal 3	NEJ
54-5F	100-127	Dubbleringar av ovanstående funktioner			NEJ
80-FF	128-255	Lediga, genererar strobe			JA

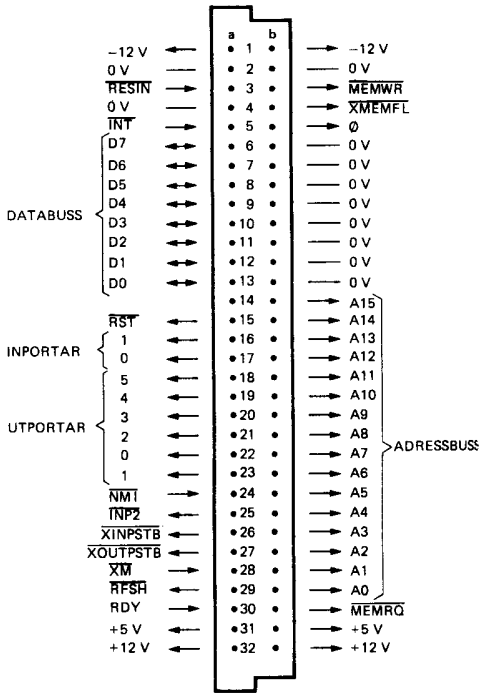
BILAGA 5 Anslutningsdon



- Signaler till/från DISPLAY POWER-anslutningen

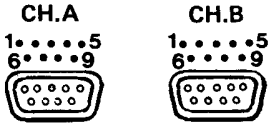
1	Matningsspänning (+17 – +24 V)
2	Kraftjord
3	
4	
5	Video
6	
7	Signaljord
8	SYNK (H+V)
9	B-signal (blå)
10	G-signal (grön)
11	R-signal (röd)
12	
13	
14	LF
15	





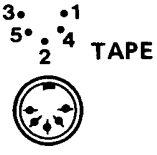
- Signaler till/från CH.A och CH.B

1	DTR	(Data Terminal Ready)
2	$\overline{\text{TxD}}$	(Transmitt Data)
3	$\overline{\text{RxD}}$	(Receive Data)
4	RTS	(Request to Send)
5	CTS	(Clear to Send)
6	DSR	(+12 V)
7	GND	(Jord)
8	DCD	(Data Carrier Detect)
9	–12 V	

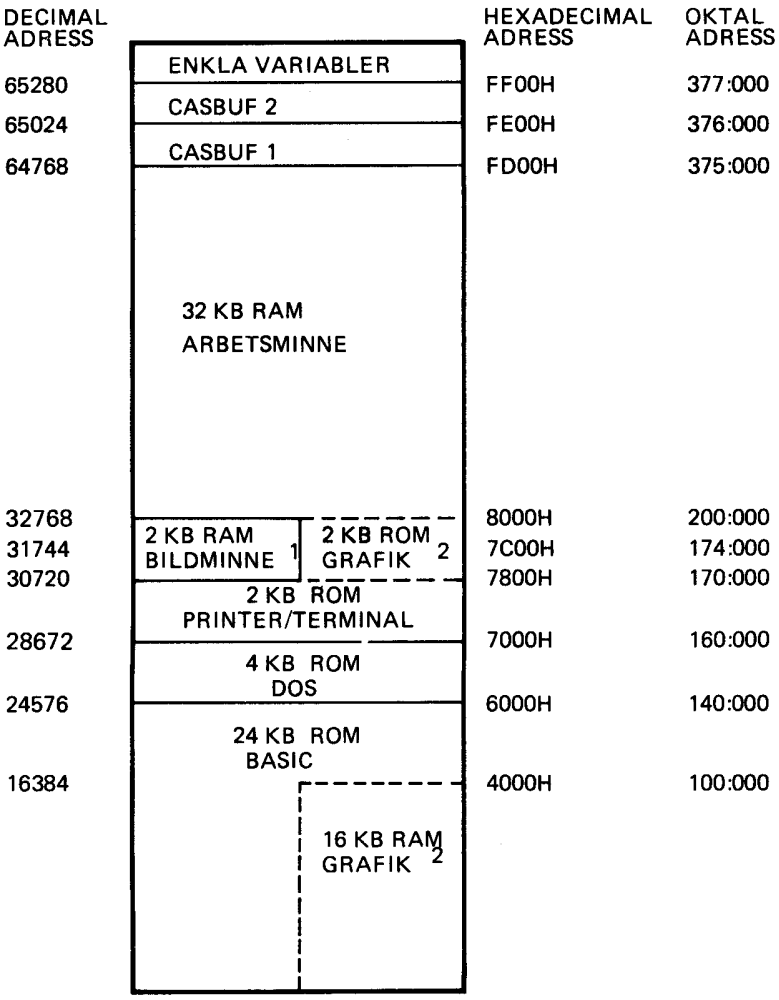


- Signaler till/från TAPE-anslutningen

1	Signal ut
2	Jord
3	Signal in
4	Motorstyrning
5	Motorstyrning



Minneskarta ABC M/C HR utan flexskiveenhet ansluten



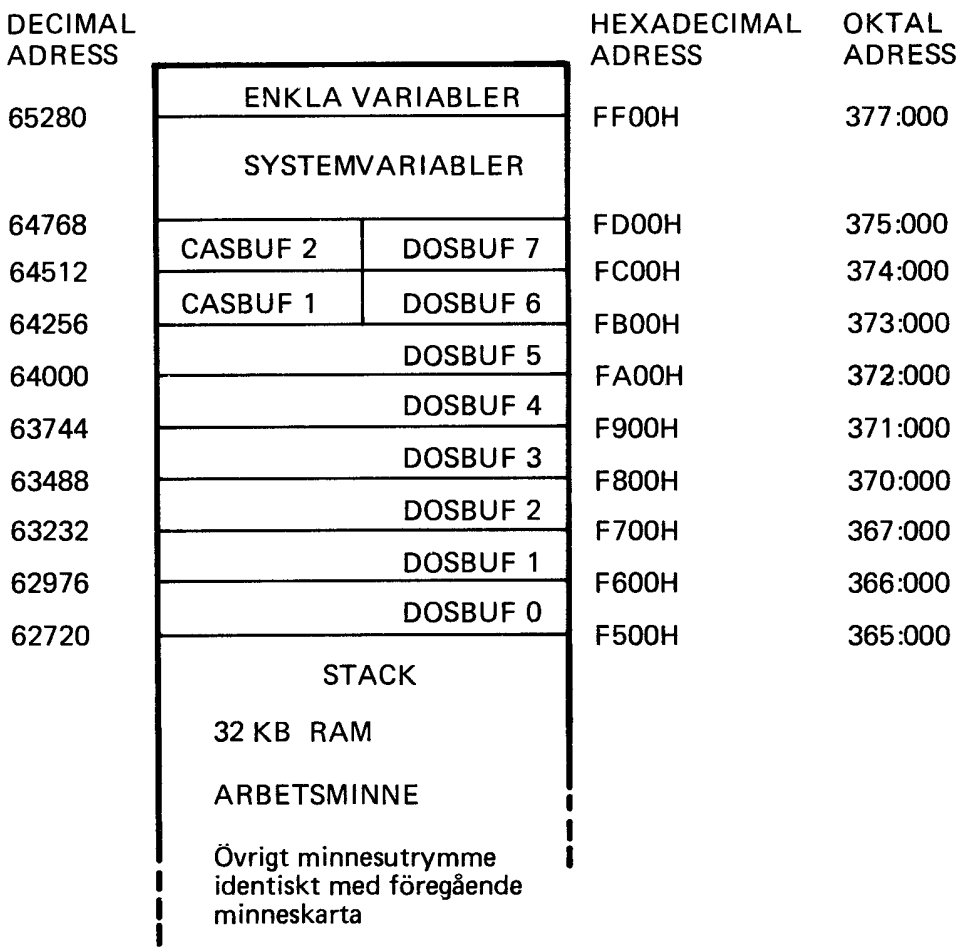
1. ABC 800 C använder endast 1 KB bildminne (31744–32768).

2. Bildminnet (2 KB) på VU-kortet ligger parallellt med systemprogrammet för grafik (2 KB) på PU-kortet. Likaså ligger bildminnet för grafik (16 KB) parallellt med systemprogrammet för BASIC. De olika minnesareorna inkräktar dock inte på varandra utan ABC 800 går över i en specialmod då grafikminnet adresseras.

   Om minnesutrymme för maskinspråksrutiner ska reserveras, ändras följande adresser:

   - Pekare till lägsta minnesadress för BASIC-program (BOTTOM): 65292
   - Pekare till högsta minnesadress för BASIC-program (TOP): 65294

Minneskarta ABC 800 med flexskiveenhet ansluten



BILAGA 7 Felmeddelanden

Fel 19–68 : I/O-fel
Fel 130–176: Fel vid programkörning
Fel 180–191: Logiska fel
Fel 200–211: Allmänna fel
Fel 220–234: Formella BASIC-fel

Fel (Error)	Meddelande	Kommentar
19	Kan ej öppna fler filer	Sju filer är öppnade
20	För lång rad (>160 tkn)	En rad får innehålla max 160 tecken
21	Hittar ej filen	Filen finns inte eller har sökts under fel namn
32	Filen ej öppnad	
34	Slut på filen	Försökt läsa efter filslut
35	Checksummafel vid läsning	Skivan eller kassettbandet är skadad
36	Checksummafel vid skrivning	Skivan är skadad
37	Felaktigt sektorformat	Fel på skiva eller kassett
38	Sektornummer utanför filen	Försök att läsa längre än filen medger
39	Filen skrivskyddad	
40	Filen raderskyddad	
41	Skivan full	Filen får ej plats på skivan
42	Enheten ej klar	Enheten ej klar, t ex ej ansluten.
43	Skivan skrivskyddad	
44	Logisk fil ej öppnad	
45	Fel logiskt filnummer	
46	Fel enhetsnummer	
47	Fel trapnummer	
48	Fel i biblioteket	
49	Felaktigt fysiskt filnummer	
51	Enheten upptagen	
52	Ej till denna enhet	
53	Funktionstangent	Funktionstangent har tryckts ned i INPUT- eller INPUT LINE-sats
54	IEC både sändare och mottagare	IEC-option
55	IEC-mottagare ej aktiv	IEC-option
56	IEC-sändare ej aktiv	IEC-option
57	Tecken från tangentbord ej i tid	
58	Ogiltigt tecken inläst	
64	Felaktigt "NAME"	Nya filnamnet existerar redan
68	Felaktig tidspecifikation	

Fel (Error)	Meddelande	Kommentar
130	För stort flyttal	
131	Index utanför tillåtet område	Försök att använda index större än motsvarande DIM
132	För stort heltal	
133	Fel i ASCII-aritmetiskt uttryck	
134	Index utanför strängen	Index för stort eller negativt
135	Negativ "SPACE ¤", "STRING ¤ " eller "TAB" <1	
136	För lång sträng	För liten dimension på den mottagande strängen
137	Ej tillåtet öka "DIM"	Ett fält får inte ökas utöver sin ursprungliga längd
138	Fel värde i "ON"-uttryck	
139	"RETURN" utan "GOSUB"	En RETURN-sats påträffad utan att en föregående GOSUB-sats har blivit utförd
140	Felaktig "RETURN"-variabel	
141	Data slut	Datalistan har blivit tömd och en READ-sats efterfrågade fler data
142	Felaktigt argument i funktion	
143	Felaktig "SYS"-funktion	
144	Ej tillåten rad	
145	"FNEND" utan föregående "RETURN"	
146	"PRINT USING" fel	Felaktigt format i PRINT USING-sats
147	Felaktiga data	
148	För lite indata	För få data inmatade vid INPUT
149	"RESTORE" ej på en "DATA"-rad	
150	För mycket indata	För många data inmatade vid INPUT
151	"RESUME" utan fel	
176	Grafisk punkt utanför bildskärmen	

Fel (Error)	Meddelande	Kommentar
180	Hittar ej detta radnummer	Referens till ett radnummer som inte finns i programmet
181	Felaktigt inhopp i funktion	
182	"NEXT" eller "WEND" saknas	
183	"FOR" eller "WHILE" saknas	
184	Fel variabel efter "NEXT"	
185	Blandade "FOR"-loopar med samma variabel	
186	"FOR"-loop med lokal variabel ej tillåtet	Gäller i flerradiga funktioner
187	Funktion ej definierad	Anrop till ej definierad funktion
188	Flera funktioner med samma namn	
189	Felaktig funktion	Ej tillåtet att blanda flera "DEF"
190	Fel antal index	Antalet index överenstämmer ej med DIM
191	Ej tilldelningsbar i funktion	Funktionens argument är ej tilldelningsbar i funktion
200	Enheten ej ansluten	
201	Minnet fullt	Datorns primärminne har ej plats för program och data
202	"LIST"-skyddat program	
203	Fel programformat	Programmet är sparat under en ickekompatibel BASIC-version
204	"MERGE" går ej på "BAC"-fil	
205	"COMMON" fel	
206	Använd kommandot "RUN"	
207	Kan ej fortsätta	Gäller GOTO radnr och CON
208	Otillåtet som kommando	Instruktionen kan ej användas som kommando
209	Fel data till kommando	Felaktigt argument till kommandot t ex LIST # #
210	Felaktigt tal	Talet innehåller tecken som inte är siffror
211	Precision får ej ändras	Ej tillåtet ändra precision efter tilldelning av variabler

Fel (Error)	Meddelande	Kommentar
220	Förstår ej	Formellt BASIC-fel
221	Otillåtet tecken efter satsen	Formellt BASIC-fel. Datorn förväntade RETURN, kolon (:) eller utropstecken (!)
222	Måste vara först på en rad	
223	Fel antal eller typ av argument	
224	Otillåten blandning av tal och strängar	
225	Ej enkel variabel	Ej tillåtet ha index på variabel t ex i FOR-loop
226	Felaktig sats efter "ON"	Formellt BASIC-fel
227	" , " saknas	Formellt BASIC-fel
228	" = " saknas	Formellt BASIC-fel
229	") " saknas	Formellt BASIC-fel
230	"AS FILE" saknas	Förekommer i OPEN- och PREPARE-satser
231	"AS" saknas	Fel i NAME . . . AS . . .
232	"TO" saknas	Förekommer i FOR-loopar
233	Radnummer saknas	
234	Felaktig variabel	

BILAGA 8 Färgvalstabell

B=blå, C=cyan, G=gul, GR=grön, M=magenta, R=röd, S=svart, V=vit

Färgvals-kommando Grafik + text	Färgnr 0	1	2	3	Färgvals-kommando Enbart grafik
0	S	S	S	S	128
1	S	V	V	V	129
2	S	R	GR	G	130
3	S	R	GR	B	131
4	S	R	GR	M	132
5	S	R	GR	C	133
6	S	R	GR	V	134
7	S	R	G	B	135
8	S	R	G	M	136
9	S	R	G	C	137
10	S	R	G	V	138
11	S	R	B	M	139
12	S	R	B	C	140
13	S	R	B	V	141
14	S	R	M	C	142
15	S	R	M	V	143
16	S	R	C	V	144
17	S	GR	G	B	145
18	S	GR	G	M	146
19	S	GR	G	C	147
20	S	GR	G	V	148
21	S	GR	B	M	149
22	S	GR	B	C	150
23	S	GR	B	V	151
24	S	GR	M	C	152
25	S	GR	M	V	153
26	S	GR	C	V	154
27	S	G	B	M	155
28	S	G	B	C	156
29	S	G	B	V	157
30	S	G	M	C	158
31	S	G	M	V	159

BILAGA 8 Forts.

Färgvals-kommando Grafik + text	Färgnr 0	1	2	3	Färgvals-kommando Enbart grafik
32	S	G	C	V	160
33	S	B	M	C	161
34	S	B	M	V	162
35	S	B	C	V	163
36	S	M	C	V	164
37	R	GR	G	B	165
38	R	GR	G	M	166
39	R	GR	G	C	167
40	R	GR	G	V	168
41	R	GR	B	M	169
42	R	GR	B	C	170
43	R	GR	B	V	171
44	R	GR	M	C	172
45	R	GR	M	V	173
46	R	GR	C	V	174
47	R	G	B	M	175
48	R	G	B	C	176
49	R	G	B	V	177
50	R	G	M	C	178
51	R	G	M	V	179
52	R	G	C	V	180
53	R	B	M	C	181
54	R	B	M	V	182
55	R	B	C	V	183
56	R	M	C	V	184
57	GR	G	B	M	185
58	GR	G	B	C	186
59	GR	G	B	V	187
60	GR	G	M	C	188
61	GR	G	M	V	189
62	GR	G	C	V	190
63	GR	B	M	C	191
64	GR	B	M	V	192
65	GR	B	C	V	193
66	GR	M	C	V	194
67	G	B	M	C	195
68	G	B	M	V	196
69	G	B	C	V	197
70	G	M	C	V	198
71	B	M	C	V	199
72	S	R	S	R	200
73	S	S	R	R	201
74	S	GR	S	GR	202
75	S	S	GR	GR	203
76	S	G	S	G	204
77	S	S	G	G	205
78	S	B	S	B	206
79	S	S	B	B	207

BILAGA 8 Forts.

Färgvals-kommando Grafik + text	Färgnr 0	1	2	3	Färgvals-kommando Enbart grafik
80	S	M	S	M	208
81	S	S	M	M	209
82	S	C	S	C	210
83	S	S	C	C	211
84	S	V	S	V	212
85	S	S	V	V	213
86	R	GR	R	GR	214
87	R	R	GR	GR	215
88	R	G	R	G	216
89	R	R	G	G	217
90	R	B	R	B	218
91	R	R	B	B	219
92	R	M	R	M	220
93	R	R	M	M	221
94	R	C	R	C	222
95	R	R	C	C	223
96	R	V	R	V	224
97	R	R	V	V	225
98	GR	G	GR	G	226
99	GR	GR	G	G	227
100	GR	B	GR	B	228
101	GR	GR	B	B	229
102	GR	M	GR	M	230
103	GR	GR	M	M	231
104	GR	C	GR	C	232
105	GR	GR	C	C	233
106	GR	V	GR	V	234
107	GR	GR	V	V	235
108	G	B	G	B	236
109	G	G	B	B	237
110	G	M	G	M	238
111	G	G	M	M	239
112	G	C	G	C	240
113	G	G	C	C	241
114	G	V	G	V	242
115	G	G	V	V	243
116	B	M	B	M	244
117	B	B	M	M	245
118	B	C	B	C	246
119	B	B	C	C	247
120	B	V	B	V	248
121	B	B	V	V	249
122	M	C	M	C	250
123	M	M	C	C	251
124	M	V	M	V	252
125	M	M	V	V	253
126	C	V	C	V	254
127	C	C	V	V	255

Tangentkoder i tecken-/grafmod (ASCII - tabell)

A	T	G	A	T	G	A	T	G	A	T	G
32	Blank		56	8		80	P	P	104	h	
33	!		57	9		81	Q	Q	105	i	
34	"		58	:		82	R	R	106	j	
35	#		59	;		83	S	S	107	k	
36	¤		60	<		84	T	T	108	l	
37	%		61	=		85	U	U	109	m	
38	&		62	>		86	V	V	110	n	
39	'		63	?		87	W	W	111	o	
40	(		64	É	É	88	X	X	112	p	
41	)		65	A	A	89	Y	Y	113	q	
42	*		66	B	B	90	Z	Z	114	r	
43	+		67	C	C	91	Ä	Ä	115	s	
44	,		68	D	D	92	Ö	Ö	116	t	
45	-		69	E	E	93	Å	Å	117	u	
46	.		70	F	F	94	Ü	Ü	118	v	
47	/		71	G	G	95	_	_	119	w	
48	0		72	H	H	96	é		120	x	
49	1		73	I	I	97	a		121	y	
50	2		74	J	J	98	b		122	z	
51	3		75	K	K	99	c		123	ä	
52	4		76	L	L	100	d		124	ö	
53	5		77	M	M	101	e		125	å	
54	6		78	N	N	102	f		126	ü	
55	7		79	O	O	103	g		127	■	

ASCII-koder (A) tolkade i teckenmod (T) och grafikmod (G).
Den grafiska moden kan endast åstadkommas med ABC 800 C.

Koder från tangentbordet

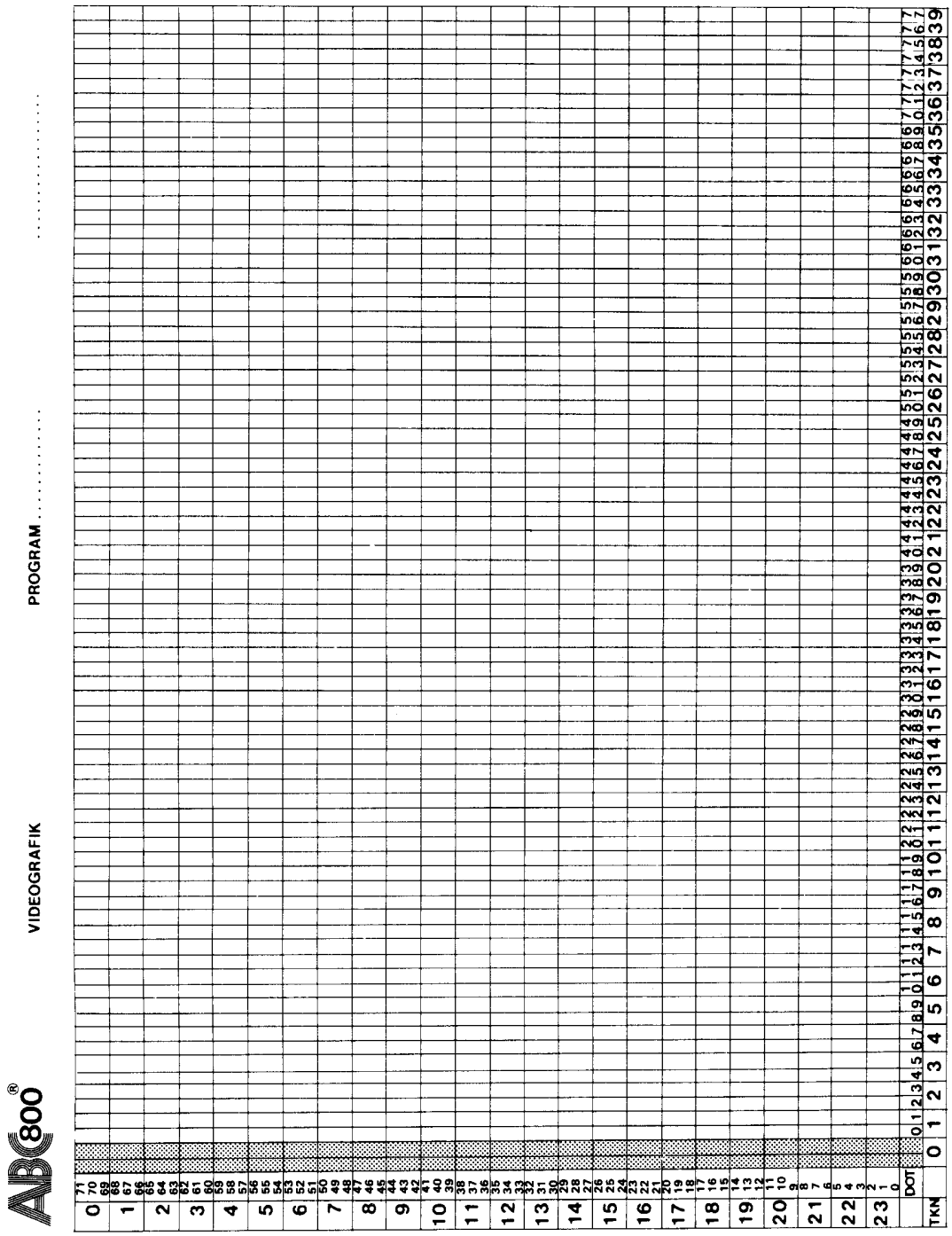
ASCII-kod	Ctrl	Shift	Tangent	ASCII-namn	Funktion
0	X		É	NUL	Tidsutfyllnadstecken
1	X		A	SOH	–
2	X		B	STX	–
3	X		C	ETX	Stoppar exekvering
4	X		D	EOT	–
5	X		E	ENQ	–
6	X		F	ACK	–
7	X		G	BEL	"Pip" i högtalaren
8	X		H	BS	*) "←" tangenten
9	X		I	HT	*) "→" tangenten
10	X		J	LF	Radframmatning
11	X		K	VT	–
12	X		L	FF	*) Raderar skärmen
13	X		M	CR	*) "RETURN" tangenten
14	X		N	SO	–
15	X		O	SI	–
16	X		P	DLE	–
17	X		Q	DC1	–
18	X		R	DC2	–
19	X		S	DC3	Stegar en programinstruktion
20	X		T	DC4	–
21	X		U	NAK	–
22	X		V	SYN	–
23	X		W	ETB	–
24	X		X	CAN	*) Tar bort skriven rad
25	X		Y	EM	–
26	X		Z	SUB	–
27	X		Ä	ESC	–
28	X		Ö	FS	–
29	X		Å	GS	–
30	X		ü	RS	–
31	X	X	O	US	–
127	X		<	DEL	–

*) Dessa tecken påverkar skärmen direkt.

Decimala koder från funktionstangenterna

		SHIFT	CTRL	SHIFT + CTRL
PF1	192	208	224	240
PF2	193	209	225	241
PF3	194	210	226	242
PF4	195	211	227	243
PF5	196	212	228	244
PF6	197	213	229	245
PF7	198	214	230	246
PF8	199	215	231	247

BILAGA 10 Videografikkarta



BILAGA 11 Referenslitteratur

1. BASIC II BOKEN
   Jan Lundgren
   Sören Thornell
   ISBN 91-86064-04-5
   Liber

2. AVANCERAD PROGRAMMERING PÅ ABC 80
   Anders Isaksson
   Örjan Kärrsgård
   ISBN 91-44-17451-9
   Studentlitteratur

3. METODHANDBOK
   Luxor Datorer AB
   Art nr: 6679589-15

4. JSP - EN PRAKTISK METOD FÖR PROGRAMKONSTRUKTION
   Leif Ingevaldsson
   ISBN 91-44-13102-X
   Studentlitteratur

5. DATASYSTEM OCH DATORSYSTEM
   Sam Nachmens
   ISBN 91-44-13152-6
   Studentlitteratur

6. MANUAL ABC 80
   Luxor datorer AB
   Art nr: 6679589-10

7. MANUAL BASIC II
   Luxor datorer AB
   Art nr: 6679210-12

ISBN 91-7260-814-5

66 79210-19

TRYCK-CENTER AB
Linköping 1983