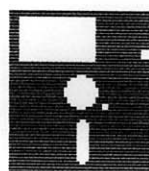
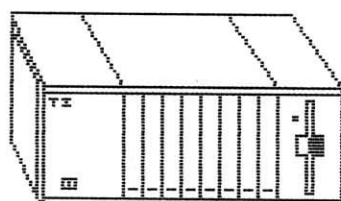
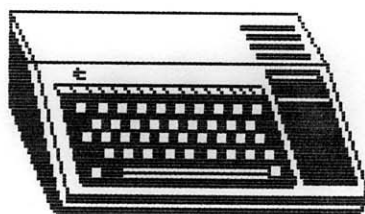


PROGRAMM

BITEN

90-9



Smartcopy för bilder	2
Tetris version 2.0	2
Brevväxling med USA	2
Myarc DM V version 1.30	2
Protokoll från årsmötet	3
Datafiler utan tårar 2	4-9
RAVE 99 Tangentbord	8
RAG-Writer 4.5 Formatera	9
McGoverns XB-skola 4	10-14
DIJIT AVPC EPROM PWRUP2A	14
Funnelweb 4.21 APR/02/90	14
Programbiten 87-88-89	15
Horizon för Geneve	15
Föreningens nya stadgar	16-18
Problem with Printers	18-21
Swedlow XB 9-11	22-24
Tigercub Tips #53	25-27
Funnelweb konfigurerings	28-34

ISSN 0281-1146

SMARTCOPY FRÅN ALEXANDER HULPKE

Programmet smArtCopy kan skriva ut bilder från TI-Artist och MyArt på en Epson-kompatibel printer. Det ger bilder ungefär som det tidigare ETIKETTER men menyn har tagits från XHI:s HARDCOPY. En gråskala används för färger både för MYA och TIA så se till att både _P och _C filerna finns med för TI-Artist. Även YAPP-bilder (testversion 0.44) skrivs ut.

Utskriften för MyArt-bilder motsvarar i stort sett HARDCOPY som finns till XHI men är nu fullständigt rensad från 9938-kod så att den även fungerar med en vanlig 99/4A.

Beställ SMARTCOPY för 15 DM från:
Alexander Hulpke, Sadowastrasse 68,
D-5600 WUPPERTAL 1, Västtyskland

TETRIS VERSION 2.0

En ny version av Tetris kan beställas från Alexander Hulpke för 15 DM. Tetris kan nu styras med joystick.

BREVVÄXLING

TI-99/4A + BOX + 2xDS/SD + Horizon
Skriv till:
Larry Tippet, P.O.Box 293,
Model City, NY 14107, USA.

MYARC DM V 1.30

MDM5-130 (för Myarc HFDC)

FIL=14 LED=94 ANV=266

filnamn sekt typ längd

LOAD	2	PROGRAM	67
MDM	7	DIS/FIX	80
MDM306	33	PROGRAM	8192
MDM307	33	PROGRAM	8192
MDM308	25	PROGRAM	5904
MDM309	25	PROGRAM	5982
MDM5	10	PROGRAM	2304
MDMCB30	26	PROGRAM	6282
MDMCR30	20	PROGRAM	4820
MDMDB30	26	PROGRAM	6180
MDMDR30	19	PROGRAM	4450
MDMFV30	15	PROGRAM	3566
MDMV	10	PROGRAM	2304
MDMXB	13	DIS/FIX	80

Redaktör: Jan Alexandersson
Utmanarredaktör: Anders Persson
Programbankir: Börje Häll

Föreningens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9 NB
S-121 46 JOHANNESHOV
Sverige

Postgiro 19 83 00-6
Medlemsavgiften för 1990 är 120:-

Datainspektionens licensnummer:
82100488

Annonser, insatta av enskild medlem (ej företag), som gäller försäljning av moduler eller andra tillbehör i enstaka exemplar är gratis.

Övriga annonser kostar 200 kr för hel sida. För lösblad (kopieras av annonsören) som skickas med tidningen gäller 200 kr per blad. Föreningen förbehåller sig rätten att avböja annonser som ej hör ihop med föreningens verksamhet eller ej på ett seriöst sätt gäller försäljning av originalexemplar av program.

För kommersiellt bruk gäller detta: Mångfaldigande av innehållet i denna skrift, helt eller delvis är enligt lag om upphovsrätt av den 30 december 1960 förbjudet utan medgivande av Föreningen Programbiten. Förbudet gäller varje form av mångfaldigande genom tryckning, duplicering, stencilering, bandinspelning, diskettinspelning etc.

Föreningens tillbehörsförsäljning: Följande tillbehör finns att köpa genom att motsvarande belopp insätts på postgiro 19 83 00-6 (porto ingår)
Användartips med Mini Memory 20:-
Nittinian T-tröja 40:-
99er mag. 12/82, 1-5, 7-9/83(st) 40:-
Nittinian årgång 1983 50:-
Programbiten årgång 84-89(styck) 50:-
TI-Forth manual 100:-
Hel diskett ur programbanken(st) 30:-
Enstaka program 5:- st + startkostnad 15 kr per skiva eller kassett (1 program=20kr, 3 program=30 kr).

MANUSSTOP: 15 aug(nr4), 1 okt(nr5),
15 nov(nr6). Telefon 08-771 05 69.
Jan Alexandersson, Springarvägen 5,
3 tr, 142 61 TRÅNGSUND

PROTOKOLL FÖR
FÖRENINGEN PROGRAMBITENS
ÅRSMÖTE 900310 I HANINGE.

1. Föreningens ordförande, Jan Alexandersson, förklarade årsmötet 1990 öppnat.
2. Sittande ordförande valdes till mötesordförande. Till sekreterare valdes föreningens ordinarie sekreterare Ake Olsson. Till justeringsmän att jämte ordföranden justera protokollet valdes Kent Edgardh och Per Rundquist.
3. Mötet konstaterades vara behörigt utlyst. 6 röstberättigade medlemmar var närvarande.
4. Föreslagen dagordning ändrades så att punkt 12 flyttades upp och blev punkt 8. Efterkommande punkter flyttades ner i motsvarande grad. Därefter godkändes dagordningen.
5. Styrelsens förslag till årsredovisning för 1989 föredrogs och godkändes.
6. Revisorssuppleanten Kent Edgardh föredrog revisorernas förslag till revisionsberättelse för 1989. Mötet godkände förslaget till revisionsberättelse för 1989.
7. Årsmötet beviljade styrelsen ansvarsfrihet för verksamhetsåret 1989.
8. Ändring av stadgarna kap. 5a, kap. 10 samt kap. 6 i enlighet med förslaget till årsmötet 1989. Ändringarna vinner laga kraft om beslut fattas med minst 2/3 majoritet. 8.1 Förslag till ändrad text i stadgarnas kap. 5a: "Styrelsen, bestående av ordförande och kassör, vilka båda utses till befattning av årsmötet samt därutöver minst tre och högst sju ledamöter i övrigt." 8.2 Förslag till ändring av hela kap. 10: "Förslag till ändring av dessa stadgar inlämnas skriftligen till FS senast sista dagen av det verksamhetsår, som skall redovisas på årsmötet, vid vilket FS utlåtande skall föreligga. För slutgiltigt bifall fordras beslut på två av varandra följande stämmor, varav minst ett årsmöte. På den stämma som sist hålles skall beslut ha fattats med

minst 2/3 majoritet." 8.3 Förslag till ändring av sista ordet i tredje stycket i kap. 6: "sammanträdet" utbytes mot "stämman". 8.4 Årsmötet beslöt enhälligt att fastställa föreslagna stadageändringar.

9. Val av styrelse. Valberedningens förslag till ny styrelse, som föredrogs av Kent Edgardh saknade ordförandekandidat. Inga avsäganden från andra styrelseledamöter fanns. Under mötet föreslogs Claes Schibler som ordförandekandidat. Årsmötet valde enhälligt följande styrelse för 1990: Ordförande: Claes Schibler Kassör: John Hanssen Övriga ledamöter: Ake Olsson, Jan Alexandersson, Börje Häll, Peter Olsson, Mikael Nordlin, Per Rundquist. Styrelsen konstituerar sig själv vad gäller befattningar i övrigt.

10. Val av revisorer. Valberedningens förslag föredrogs och efter samtal beslöt årsmötet välja följande personer: Ordinarie revisorer: Sten Gunnarsson Göteborg samt Kent Edgardh Stockholm. Suppleant: Bertil Stenfeldt Göteborg.

11. Val av valberedning. Till valberedning omvaldes Kent Edgardh samt Sten Gunnarsson. Sammankallande Kent Edgardh.

12. På förslag från styrelsen beslöt årsmötet att årsavgiften skall vara oförändrat 120:- för 1990.

13. Ordföranden avslutade årsmötet 14.50.

Vid pennan Ake Olsson

mötessekreterare

Justeras : Jan Alexandersson

mötesordförande

Kent Edgardh

justeringsman

Per Rundquist

justeringsman

DATAFILER UTAN TÅRAR - 2

av Peter Walker, TI*MES, England

Detta är den andra artikeln som syftar till att förklara mysterierna med datafiler, vilket inte täcks av användarmanualen på något bra sätt och som kan vara mycket förvirrande för nybörjaren. Även om du bara har grunddatorn med kassettspelare, kommer jag att förklara hur du kan använda datafiler.

Låt oss nu titta på konstruktion av en datafil för ett spelprogram. Vi vill kunna läsa och sedan spara tillbaka HISCORE och NAME för den som har fått högsta poäng (score). Antag att namnet inte är längre än 15 tecken. Vi önskar lagra tre sådana poster. Så varje post kommer att bestå av en sträng med längden 15 bytes plus en längdbyte, och en numerisk med 8 bytes plus en längdbyte. Vi använder INTERNAL format eftersom detta är mest användbart när det finns mer än ett element per post. Vi kan använda VARIABLE format för att spara minne, även om detta inte är någon stor fil. Vi behöver utrymme för en maximal postlängd av $15+1+8+1=25$. Således kommer vår fil att öppnas på följande sätt:

```
100 OPEN #1:"DSK1.Scores",INTERNAL,
    VARIABLE 25
110 FOR I=1 TO 3
120 INPUT #1:HISCORE(I),NAME$(I)
130 NEXT I
```

och sedan i slutet av spelet:

```
1000 FOR I=1 TO 3
1010 PRINT #1:HISCORE(I),NAME$(I)
1020 NEXT I
1030 CLOSE #1
```

Antag nu att vi vill göra exakt samma sak med CS1. Vi kan inte använda "default"-mod UPDATE och inte heller VARIABLE-filer. Programmet kommer att se ut så här:

```
100 OPEN #1:"CS1",INTERNAL,FIXED 25,
    INPUT
110 FOR I=1 TO 3
120 INPUT #1:HISCORE(I),NAME$(I)
130 NEXT I
140 CLOSE #1
```

och sedan i slutet av spelet:

```
1000 OPEN #1:"CS1",INTERNAL,
    FIXED 25,OUTPUT
1010 FOR I=1 TO 3
1020 PRINT #1:HISCORE(I),NAME$(I)
1030 NEXT I
1040 CLOSE #1
```

Detta kommer att fungera bra, men dess största nackdel är snabbheten. Som med all överföring till band, är det ett långt uppehåll innan lagringen börjar. Det som är synnerligen tröttsamt med datafiler på kassetband är att det finns en 5 sekunders bärivågston mellan varje post. Detta gör vår enkla 3-posters fil mycket långsam att läsa och skriva. Därför är det förnuftigt att packa så mycket som möjligt i varje post när man använder kassetband. Eftersom en kassetfil kan ha en maximal längd av 192 bytes, kan vi faktiskt få plats med alla tre namnen och poängen i en enda post. Detta minskar tiden betydligt. För att visa detta tittar vi på slutet av spelet igen:

```
1000 OPEN #1:"CS1",INTERNAL,
    FIXED 75,OUTPUT
1010 FOR I=1 TO 3
1020 PRINT #1:HISCORE(I),NAME$(I),
1030 NEXT I
1040 CLOSE #1
```

Lägg märke till det avslutande kommatecknet på rad 1020. Detta åstadkommer ett "pending"-print-villkor vilket säkerställer att det andra och tredje poäng/namn skrivs till samma post. Utan detta efterföljande kommatecken, hade en separat post använts varje gång PRINT-kommandot utförts vilket var just det som vi försöker undvika. "Pending"-villkoret avslutas i detta fall endast genom att filen stängs med CLOSE, men i allmänhet avslutas "pending"-print med ett PRINT-kommando utan efterföljande kommatecken. I vårt exempel skrivs alla 6 dataelementen till kassetbandet omedelbart innan filen stängs med CLOSE.

Det har ofta påståtts att kassett-filer inte kan kontrolläsas på samma sätt som när man sparar program-filer med SAVE. Även om systemet inte kan göra detta automatiskt, så kan du med Extended Basic skapa din egen kontrollrutin. ON ERROR används för att fånga in fel som upptäcks i en lagrad post och ger dig möjlighet att spara filen på nytt:

```

100 ON ERROR 5000
110 GOSUB 2000
120 (Game program)
---
999 REM STORE ROUTINE
1000 OPEN #1:"CS1",INTERNAL,
      FIXED 75,OUTPUT
1010 FOR I=1 TO 3
1020 PRINT #1:HISCORE(I),NAME$(I),
1030 NEXT I
1040 CLOSE #1
1050 PRINT
1060 PRINT "* CHECK TAPE (Y OR N)?"
1070 CALL KEY(3,K,V)::IF V-1 THEN
      1070
1080 IF K<>89 THEN STOP
1090 IP=1 :: GOSUB 2000
1100 PRINT "":"DATA OK"
1110 STOP

2000 OPEN #1:"CS1",INTERNAL,
      FIXED 75,INPUT
2010 FOR I=1 TO 3
2020 INPUT #1:HISCORE(I),NAME$(I),
2030 NEXT I
2040 CLOSE #1
2050 RETURN

5000 ON ERROR 6000::CALL ERR
      (R,N,K,J)::IF J<2000 OR J>2040
      THEN RETURN 6000
5010 PRINT "":"* ERROR DETECTED IN
      DATA":CLOSE #1::ON ERROR
      5000::IF IP=0 THEN RETURN 110
      ELSE RETURN 5020
5020 PRINT "":"PRESS R TO RECORD":
      "PRESS C TO CHECK":"PRESS E
      TO EXIT"
5030 CALL KEY(3,K,V)::IF V-1 THEN
      5030
5040 ON POS("RCE",CHR$(K),1)+1 GOTO
      5030,1000,1090,1110
6000 PRINT "":"ERROR";R;"IN LINE";J
      ::STOP

```

Så mycket för vårt spelprogram. Antag att vi önskar lagra stora mängder data på kassett? Med hänsyn till den mycket långsamma datalagringen är mitt råd:

1. Glöm de vanliga reglerna med poster uppdelade på dataelement. Försök att fylla den maximala 192 bytes-posten så bra som möjligt. En 192-bytes-post kan t.ex. rymma 21 numeriska element.

2. Undvik om möjligt långa strängar.

3. Som alternativ kan du använda Personal Record Keeping-modulen för att skapa databasprogram. Denna lagrar data mycket snabbare än BASIC genom att använda PROGRAM-format.

4. Använd Enhanced Basic (se PB 85-2 och PB 88-3 ö.a.). Detta är den BASIC som finns inbyggd i PRK- och Statistics-modulerna med några extra underprogram (CALL P,L,S,G,H,A,D). Dessa ger dig möjlighet att skapa och lagra PRK-kompatibla datafiler i PROGRAM-format. Jag kommer här att visa hur du kan lagra 50 HISCORE och NAME med användning av denna metod.

```

50 DIM HISCORE(50),NAME$(50)
100 CALL L("CS1",C)
110 IF C<>1 THEN 500
120 FOR I=1 TO 50
130 CALL G(1,I,1,M,HISCORE(I))
140 CALL G(1,I,2,M,NAME$(I))
150 NEXT I
160 REM GAME PROGRAM FOLLOWS
170 FOR I=1 TO 50
180 PRINT HISCORE(I);NAME$(I)
190 NEXT I
200 GOTO 1000
500 CALL CLEAR
510 CALL D(6,1,22,"WAS CALL P(1500)
      USED?")
520 CALL D(8,1,11,"ENTER Y/N Y")
530 CALL A(8,11,1,F,AN$)
540 IF AN$<>"Y" THEN 600
550 CALL D(10,1,13,"RETRY LOADING")
560 GOTO 100
600 CALL D(12,1,21,"USE CALL P(1500)
      THEN")
610 CALL D(13,1,14,"RELOAD PROGRAM")
620 STOP
1000 CALL H(0,1,1,"HISCORES")
1010 CALL H(0,5,2,2)
1020 CALL H(0,10,1,2)
1030 CALL H(0,10,2,1)
1040 CALL H(0,11,1,5)
1045 CALL H(0,11,2,15)
1050 FOR I=1 TO 50
1060 CALL G(0,I,1,HISCORE(I))
1070 CALL G(0,I,2,NAME$(I))
1080 NEXT I
1090 CALL S("CS1",C)
1100 REM STORING COMPLETE

```

1110 STOP

Innan detta program laddas och körs måste du skriva CALL P(1500) för att dela upp minnet för PRK-data. Om du inte gör detta kommer inte filen att laddas och du att få ett felmeddelande.

DATAFILERS LÄNGD

Ett vanligt problem vid programmering med filer är hur man ska veta hur många poster som filen innehåller. Detta är värdefullt att veta eftersom en enkel loop eller FOR-TO-NEXT-slinga vanligen används för att läsa in posterna och det är klart viktigt att veta när man ska hoppa ut ur slingan! Jag rekommenderar att ett av följande tre sätt används för att lösa av problemet:

1. Om du har ett disksystem, möjliggör användning av EOF-funktionen före varje INPUT-kommando, att du kan upptäcka när du nått slutet av filen. EOF kan inte användas med kassettspelare.

2. Du kan använda en utfyllnads-post som markering av slutet av filen. Som exempel kan du med sträng-poster skriva dataelementet "ZZZZ" för att indikera slutet eller med numeriska poster elementet 99999. Eftersom båda dessa är osannolika att uppträda normalt kan du testa om de finns när varje post läses in och vara förvissad om att de markerar slutet på filen.

3. Eftersom programmet som skapar filen vanligen vet hur många poster som har skrivits till filen, kan du skriva antalet poster till postnummer 0 och endast använda post 1 och framåt för data. På detta sätt kommer den första posten som läses att indikera hur många fler som kommer vilket således kan användas för att starta en FOR-TO-NEXT-slinga för att läsa in det rätta antalet poster.

Låt oss titta på de tre metoderna:

METOD 1

```
100 OPEN #1:"DSK1.FILE",INTERNAL,
```

FIXED 40

```
110 IF EOF(1) THEN 1000
```

```
120 INPUT #1:A$(I)
```

```
130 I=I+1
```

```
140 GOTO 110
```

```
1000 CLOSE #1
```

```
1010 REM Rest of program ...
```

METOD 2

```
100 OPEN #1:"DSK1.FILE",INTERNAL,  
FIXED 40
```

```
110 INPUT #1:A$(I)
```

```
120 IF A$(I)="ZZZZ" THEN 1000
```

```
130 I=I+1
```

```
140 GOTO 110
```

fortsätter sedan som metod 1

METOD 3

```
100 OPEN #1:"DSK1.FILE",INTERNAL,  
FIXED 40
```

```
110 INPUT #1:N
```

```
120 FOR I=1 TO N
```

```
130 INPUT #1:A$(I)
```

```
140 NEXT I
```

```
150 CLOSE #1
```

ATT UTNYTTJA LAGRINGSUTRYMMET

Vi har tidigare visat hur kassettfiler kan användas optimalt genom att fylla varje 192-bytes post av maxlängd med så mycket information som möjligt. Bra konstruktion av filer gäller även diskfiler. Varje disk-sektor lagrar 256 bytes men ingen post får delas mellan två sektorer. Således kan varje sektor endast rymma en post om postlängden är 200, så de resterande 56 byten är bortslösade. Å andra sidan kan en sektor rymma två poster med längden 128 utan att något utrymme slösas bort (postlängder på 128, 85, 64, 51, 42, 36, 32, 28 osv ger optimal packning på disk ö.a.).

Vi har också sett att filer med VARIABLE längd kan spara utrymme. Ett annat sätt att spara utrymme vilket är speciellt användbart för att lagra heltal är att använda DISPLAY istället för INTERNAL-format. Exempelvis använder talet 55 nio bytes när det lagras i INTERNAL-format men endast tre när DISPLAY-

format används (två sträng-bytes plus kommatecken som behövs för att separera dataelementen i DISPLAY-format). Detta sparande fås till priset av de ytterligare komplikationer som beskrivits ovan och som uttryckligen kräver kommatecken mellan element för att inte tala om situationstecken som krävs runt sträng-element.

EXTERNT BESTÄMDA DATASTRUKTURER

Vi har sett hur vi kan använda RELATIVE filorganisation med filer med FIXED postlängd för att få direkt åtkomst av godtyckligt post i filen. INPUT-kommandot kräver dock att varje element i posten läses i nummerordning. Detta har vissa nackdelar, men det finns en metod för filkonstruktion där elementen bestäms externt av ditt program. Om vi använder DISPLAY/FIXED-filer så kan vi inte bara komma åt posten individuellt men vi kan även definiera element som FIXED längd av varje post. Vi kan läsa varje post som en post med endast ett element (som datorn uppfattar det) och använda SEG\$-funktionen för att komma åt varje element inom posten. När en sådan post skapas hanteras alla dataelement som strängar (och således tjänar lagringsutrymme för heltal enligt ovan) och elementen adderas med &-funktionen innan den skrivs till filen. En sådan post har ingen längd-byte eller element-avskiljare vilket kan ha ett egenvärde i vissa tillämpningar.

Jag tror att ett exempel kan visa vad jag syftar på. Antag att vi vill lagra 100 namn (max längd 10 tecken) och ålder (upp till 99 år) samt ytterligare tre andra viktiga data (t.ex. 36,24,36). Det vanliga sättet skulle vara så här:

```
100 OPEN #1:"DSK1.FILE",INTERNAL,
    FIXED 47 ! 47=11+4*9
110 FOR N=1 TO 100
120 PRINT #1:NAME$(N);AGE(N);B(N);
    W(N);H(N)
130 NEXT N
140 CLOSE #1
```

Varje sektor kommer att lagra 5 poster så filen kommer att behöva totalt 21 sektorer. Filhuvudet

kräver den 21:a sektorn.

Som jämförelse betrakta följande:

```
100 DEF F$(A$)=SEG$(A$&RPT$(" ",10),
    1,L) ! PAD OUT A$ TO CHAR LENGTH
    OF L.
110 OPEN #1:"DSK1.FILE",DISPLAY,
    FIXED 18 ! 18=10+4*2
120 FOR N=1 TO 100
130 L=10
140 X$=F$(NAME$(N))
150 L=2
160 X$=X$&F$(STR$(AGE(N)))&
    F$(STR$(B(N)))&F$(STR$(W(N)))&
    F$(STR$(H(N)))
170 PRINT #1:X$
180 NEXT N
190 CLOSE #1
```

Med denna 18 bytes post får vi plats med 14 poster per sektor och behöver totalt 9 sektorer, en inbesparing på 11 sektorer. Låt oss titta på hur denna hoppackade fil kan läsas:

```
50 DIM STAT(100,4)
100 FOR N=1 TO 5
110 READ WIDTH(N)
120 DATA 10,2,2,2,2
130 NEXT N
140 OPEN #1:"DSK1.FILE",DISPLAY,
    FIXED 18
150 FOR N=1 TO 100
160 P=1
170 INPUT #1:X$
180 ITEM=1
190 NAME$(N)=SEG$(X$,P,WIDTH(ITEM))
200 P=P+WIDTH(ITEM)
210 FOR ITEM=2 TO 5 ! Numerics put
    in array for program efficiency
220 STAT(N,ITEM-2)=VAL(SEG$(X$,P,
    WIDTH(ITEM)))
230 P=P+WIDTH(ITEM)
240 NEXT ITEM
250 NEXT N
260 CLOSE #1
```

Du kanske vill veta att det utmärkta (men svårt att förstå till att börja med) Navarone Data Base Management Systems använder denna typ av filstruktur. Denna struktur har ytterligare en fördel. Om du t.ex. har en namn- och adressfil, kan du få hopsatta fält för efternamn och förnamn vilka kan användas som ett enskilt element när filen sorteras i alfabetisk ordning. Du kan således undvika svårigheten med komplicerad sortering. Detta är möjligt eftersom filen inte har några skiljetecken

TANGENTBORDSINTERFACE FRÅN RAVE 99

av Bertil Stenfeldt

Rave 99 har tagit fram dels ett tangentbord som är klart att ansluta till 99:an, dels ett interface för anslutning av ett XT-tangentbord. Detta interface finns dels färdigbyggt dels som byggsats. Tillsammans med årets första nummer av P-biten fanns en kopia av Rave 99:s broschyr om dessa produkter, varför jag inte i detalj beskriver funktionerna.

Jag införskaffade en byggsats från Rave samt några komponenter från en elektronikfirma i stan. Sedan lödde jag ihop prylarna. Det visade sig vara ganska enkelt att bygga interfacet.

När det var klart köpte jag ett tangentbord som var omkopplingsbart mellan XT och AT. Det krävs att tangentbordet ska klara av XT. En sak som man måste se upp med när det gäller tangentbordet är att det bör vara den amerikanska placeringen av tangenterna. Det blir nämligen ganska trassligt med A, Å och Ö annars. De tangenter som på ett tangentbord med svenska tangenter är A, Å och Ö blir nämligen inte dessa tecken tillsammans med interfacet. Det är också lättare att känna igen sig på ett amerikanskt tangentbord när man är van vid Texas tangentbord.

Den enda nackdelen är att stora A, Å och Ö hamnar på "lower case" dvs utan SHIFT medan man måste använda SHIFT för att skriva motsvarande små bokstäver. Hur som helst blir det

lättare än med Texas bord.

Det som avgjort blir enklare är användningen av specialtecken och funktionstangenter. Man behöver inte trycka på t ex <FCTN 1> för att ta bort ett tecken. Piltangenterna fungerar utan problem, bläddring sidvis i t ex TI Writer fungerar genom tryckning på en enda tangent för vardera riktningen osv.

Vad kostar kalaset? Kortet kostade 92 dollar + frakt och moms dvs ca 900:-. Till detta kommer komponenter för ca 75:- samt ett tangentbord som man kanske kan skaffa relativt billigt begagnat. Ett nytt kostar ca 450:- men då är det lite problem att få tag i ett utan svenska tecken. Det enklaste blir att köpa ett färdigt tangentbord och koppla in direkt i konsolen. Ett sådant kostar 225 dollar + frakt och moms dvs ca 2000:-.

Varför ska man då skaffa ett annat tangentbord? Så här i efterhand frågar jag mig detta också. Det som är fördelen är den enklare användningen genom att man mestadels slipper trycka på flera tangenter och att man kan ställa konsolen en bit från arbetsplatsen. Tangentbordet är också lägre och allmänt bättre att arbeta med. Nackdelen är priset. Om man kan tänka sig att lägga ned denna summa kanske man i stället ska överväga att skaffa en riktig PC. Det går att få begagnade sådana för en hyfsad slant! ■

eller längdbytes mellan två element.

IDIOTSÅKRING

Om ditt program tillåter att filnamnet skrives in under programkörningen, är det förutseende att ta hand om händelser som kan krascha programmet eller förlora värdefulla data:

1 Datamediat kan vara tomt, skrivskyddat eller ej anslutet.

2 Filen kan redan finnas med andra attribut.

3 Filen kan redan finnas med samma attribut.

För att gardera mot dessa svårigheter kan du använda ON ERROR för

RAG-WRITER VERSION 4.5

FORMATERAREN

av Jan Alexandersson

Programbiten 89-1 t.o.m. 90-2 har sammanställts med FW 80-kolumns-editor och sedan skrivits ut med Print File direkt från editorn. Utskriften har gjorts på en NEC P6 skrivare med 24 pinnar. Printern är inställd för fullt IBM-teckenset (256 tecken). Första sidan har dock skrivits ut med FW Formateraren så att IBM:s grafiktecken för fyrkant och ram kan användas.

Fr.o.m. Programbiten 90-3 har RAG TI-WRITER 4.5 använts för formateringen tillsammans med FW 80-kolumnseditor. Den stora förbättringen är att de fyra specialtecken som används av formateraren kan definieras om:

- & tecken 38: understrykning
- @ tecken 64: fet stil
- * tecken 42: adresslista
- ^ tecken 94: blanktecken

Det finns dock en liten miss när det gäller tecken 94, som visserligen kan definieras om, men den tidigare funktioner för tecken 94 finns kvar. Det går dock att trixa genom att skriva .TL 94:94 som translitterering så att tak-tecknet kommer ut som om inget har hänt. Det visade sig att samma sak fungerar med FW-formateraren.

Ett annat problem som dök upp var att både FW- och RAG-formateraren gör en extra Line Feed efter Form Feed. Detta medför att andra sidan kommer en rad längre ned än första sidan. Den ursprungliga TI-format-

eraren hade en Form Feed före utskriften av första sidan. Detta gör att alla sidor hamnar lika långt ned på bladet vilket nog var bättre, men det slösar ju med papper vid utskrift av enstaka sidor.

Jag använder normalt följande inställning av formateraren:

```
.CO RAG FORMATTER
.DU 128;DB 128;DM 128
.PL 72
.TL 94:94
.TL 91:142
.TL 92:153
.TL 93:143
.TL 96:130
.TL 123:132
.TL 124:148
.TL 125:134
.TL 126:254
```

Skrivaren är inställd på amerikanskt teckenset hela tiden. De svenska tecknen får jag genom .TL till det ställe där IBM har placerat de svenska bokstäverna. På detta sätt får jag riktigt dollartecken(\$), alfaslang(@) och tak(^) tillsammans med ÅÄÖåäö och fylld fyrkant(■).

Med translitterering kan jag använda alla IBM:s grafiktecken. Det gäller bara att hitta ett tecken 0-127 som ej används på vanligt sätt för detta utbyte.

Det finns även ett nytt kommando .PC (Printer Control) som kan sända godtyckliga ASCII-koder 0-255 och således kan det ofta ersätta CTRL-U. ■

punkt 1 och 2 ovan. Genom att fånga in I/O ERROR och FILE ERROR kan du hindra programmet från att stanna. Det tredje problemet fångas emellertid inte in av operativsystemet. Av de vanliga TI-modulerna är det endast Multiplan som frågar dig om du verkligen vill skriva över en existerande fil. Det finns två sätt att lösa detta i Basic. För det första kan du söka genom diskkatalogen och

se efter om filnamnet finns. Detta är emellertid långsamt. Ett andra och snabbare sätt är att öppna filen och omedelbart testa värdet av EOF-funktionen. En nyskapad fil kommer att ge EOF-värdet 1, medan en befintlig fil som innehåller data ger värdet 0. Du kan således fånga in risken för att filen skrivs över innan användningen av PRINT börjar. ■

MCGOVERNS XB-SKOLA, DEL 4

av Tony McGovern, Australien

V. XB-STILEN MED UNDERPROGRAM

Låt oss nu titta på det bästa sättet att bygga XB. Antag nu att vi håller på att utveckla ett program, men ännu inte nått den punkt där det blir viktigt att krympa programrader. Den första sak att observera är att genom att ge underprogrammen (anropas med CALL ...) bra beskrivande namn så har du redan kommit långt när det gäller att göra programmet självförklarande.

Hur stora ska individuella underprogram tillåtas bli? En orsak till att använda dem är ju att dela upp stora program i hanterbara block. Vi kommer att använda begreppet 'rad' för att hänvisa till en "multi-statement" XB-rad med eget radnummer. Min egen uppfattning är att, utom i undantagsfall, underprogram inte ska vara mer än 10 rader långa, och oftast kortare än detta. Vad kan vara ett undantagsfall? Ett uppenbart fall är titelblock, som det i SIMPLIST vilket lämnades som en nästan tom snutt. En fullständig version skulle ha grafik och hjälpskrämmar, vilka kan vara besvärande långa att skriva, men innehåller mycket litet när det gäller hopp mellan olika val eller tilldelning av variabler. Ett annat exempel är där en välkänd rutin som redan fungerar, ska användas med obetydliga ändringar som i COLIST där diskatalogrutinen från diskmanualen har tagits med som ett underprogram med endast mindre ändringar. I alla sådana fall där långa underprogram är motiverade, skall parameterlistor som ska överföras vara kort eller obefintlig.

Den andra ytterligheten är korta en- eller tvåraders som ofta anropas för små speciella uppgifter, mer eller mindre dina egna utökningar av de inbyggda uppsättningen av underprogram (anropas med CALL ...). Däremellan finns medellånga underprogram med betydande parameterlistor och logik som övertar bördan av programflödet.

Några underprogram kanske endast anropas en gång från ett annat underprogram men är av värde när det gäller att göra din kod lättare att läsa och modifiera. Dessa hänger samman med grening av programflöde med hjälp av IF...THEN...ELSE-kommandon. I både TI BASIC och XB, kan FOR-NEXT-slingor utsträckas hur mycket som helst där NEXT fungerar som en begränsare. Olyckligtvis tog inte TI med något "ENDIF"-kommando i XB som finns i TI-FORTH utan endast så att 'ENDIF' gäller när XB-raden är slut. Detta betyder att alla alternativa åtgärder som bestäms av IF-villkoret måste rymmas inom XB-raden eller ha GOTO någon annanstans om inte det vanliga enkla att fortsätta på nästa rad är tillräckligt. XB-manualen förbjuder uttryckligen användning av FOR...NEXT-loopar inom IF...THEN...ELSE-kommandon. Du har utan tvivel redan vant dig vid att komma runt detta lilla problem genom att placera loop-koden i en subrutin och att använda GOSUB. Underprogram kan användas istället, efter THEN och ELSE för att göra mera komplexa alternativ möjliga, men ändå stanna inom begränsningen en programrad med ett minimum av hoppande med GOTO.

Detta tar oss till ämnet med de berömda GOTO. En hel del hett, och inte nödvändigtvis mycket ljus, har slösats på detta ämne. När allt kommer omkring är det endast ett kommando som är tillgängligt i många språk, och som har omedelbart förutsebara konsekvenser. Den verkliga kritiken gäller att det inte lämnar något spår av varifrån programmet kom. På maskinkodsnivå, möjliggör hopp att datorn kan göra mer än att följa ett rakt spår av instruktioner. Frågan är om det är en hjälp eller ett hinder i ett högnivåspråk, och om andra sätt att kontrollera programflödet kan ersätta GOTO. TI-FORTH klarar sig utan, men det mest procedurinriktade språket, TI-LOGO, har ändå användning för GOTO. Pascal försöker klara sig utan. Det som vi har när det gäller XB klarar sig inte utan GOTO. Om något skulle anses förkastligt i högnivåspråk är

det nödvändigheten att ha PEEK och POKE (VIC 64 klarar sig inte utan detta ö.a.)

Den stora svagheten med GOTO som ett språkelement är att det kan missbrukas, eftersom odisciplinerad användning gör programkoden dålig och svår för andra att följa. Det verkliga budskapet från idéerna med 'strukturerad programmering' är inte att BASIC är dåligt p.g.a. GOTO, men de flesta BASIC-dialekter (TI grund-BASIC är typisk) tvingar programmeraren att göra inskränkningar om den hemska oredan med GOTO skall kunna undvikas.

Genom att använda XB för att dela upp programmet i mindre avsnitt, har du automatiskt tagit bort behovet av att hoppa omkring med GOTO. Det enda du då behöver komma ihåg är att använda underprogram med CALL som kommando i IF..THEN..ELSE och att vara något omsorgsfull när det gäller att skriva det logiska flödet, så kommer du att finna det mycket enklare att avlusa eller utveckla program. GOTO som hoppar bakåt mer än en eller två programrader, eller hopp framåt överhuvudtaget, bör endast förekomma i mera normala logiska konstruktioner, som i SUB BASICLINE i SIMPLIST-exemplet. Enstaka rekursiva rader som i rad 620 av SIMPLIST är mycket effektiva. Det är synd att konstruktören av XB inte lade till "MYSELF"-funktionen som finns i TI-FORTH för att förbättra sådana konstruktioner.

Ytterligare en liten sak innan vi fortsätter med andra ämnen. Många programmeringsspråk med lokala procedurer tillåter även specificering av globala variabler, åtkomliga från alla delar av programmet. XB tillåter inte separata globala variabler, och det kan vara ganska tröttsamt när en parameter som definieras i slutet av en kedja av underprogram endast används i slutet av en annan kedja och måste överföras hela vägen upp och ned genom parameterlistorna. Ett sätt att undvika detta är att använda de statiska värden hos XB-underprogram.

```
3000 SUB PAGELENGTH(A,B):: I
F A THEN C=B ELSE B=C
3010 SUBEND
```

Om skrivflaggan har satts som CALL PAGELENGTH(1,66) så lagras värdet 66 i underprogrammets lokala variabel C, medan CALL PAGELENGTH(0,PL) återvinner samma värde till variabeln PL. Detta är klumpigare än att ha globala variabler, men är också mera skyddat mot oönskade störningar. XB framtvingar inte någon hierarki av nivåer med underprogram, så PAGELENGTH kan skrivas till eller läsas från godtycklig programnivå. Exemplet gäller bara en parameter, men kan enkelt utökas.

Användning av underprogram medför vissa övergripande kostnader, både när det gäller programstorlek och när det gäller tiden det tar för XB att utföra CALL till skillnad mot ett enklare GOSUB. Om du inte är helt desperat när det gäller att spara bytes, överväger fördelarna med underprogram detta pris och de bör alltid användas frikostigt i början av programutvecklingen. Argumentet med snabbhet är ej entydigt, eftersom i ett långt underprogram kan den extra tiden för CALL mer än väl uppvägas av tidsvinsten för interpretatorn eftersom den endast har en kort lista av lokala variabelnamn att söka igenom till skillnad mot en mycket större lista av globala variabler.

VI. PRE-SCAN SWITCH-KOMMANDON

Det lilla tilläggshäftet som kommer tillsammans med version 110 av Extended Basic introducerar ett nytt par av reserverade ord, !@P+ och !@P-. Dessa har formen av "tail remark" (XB-manualen sid 38) och ignoreras fullständigt av den tidigare XB-versionen 100. Om XB-interpretatorn finner ett utropstecken ! utanför någon DATA-sträng eller sträng inom situationstecken, behandlas resten av raden som om det var ett REM-kommando. Version 110 har även möjligheten att känna igen dessa två ord som börjar med ! och som avviker från den vanliga användningen av "tail remark" när de används som ettordskommando. De kan endast användas i slutet av raden så att v.100 kan ignorera dem, så att några kompatibilitetsproblem mellan olika versioner ej skapas, något som TI alltid brydde sig om. TI kunde

inte låta dessa kommandon göra någonting alls! Så varför finns de där?

XB-manualens tillägg sid 7, berättar historien. Dessa switch-kommandon tillåter dig att styra användningen av "pre-scan" genom programmet via interpretatorn -- denna irriterande långa tid efter det du skrivit RUN innan programmet börjar löpa. Interpretatorn letar sig igenom ditt program, byte för byte, och struntar endast i DATA, REM och ! (tail remark). Utöver detta kan den inte kosta på sig att strunta i något innan den verkligen har tittat på det. Pre-scan väljer lagringsarea och sätter upp tabeller för variabler, fält-variabler (array), data, underprogram och DEF som senare skall användas av interpretatorn under själva programkörningen. När den en gång har reserverat utrymme för en variabel och länknings via tabellen, behöver den naturligtvis inte göra det igen. Pre-scan switch-kommandot tillåter att programmeraren stänger av och sätter på pre-scan av programmet så att det endast tittar på det som det verkligen behöver för att göra sitt jobb.

Vad vinner programmeraren på allt detta extra krångel? Det mest uppenbara resultatet är en minskning av tiden för pre-scan. Denna kan vara betydlig i långa program. De 6 till 7 sekunder för TEX-BOUNCE, ett 12K program, kan fortfarande synas långt men förkortas till 1/4 av detta. I ett senare kapitel skall vi se hur det även kan användas för att finjustera uppförandet under själva körningen. Vilket pris måste programmeraren betala för dessa fördelar? Den nödvändiga kostnaden är minnesutrymme som upptas av de extra kommandona. Den dolda kostnaden, som uppkommer under själva programskrivandet, är de oförklarliga buggar som kan komma in i programkoden och förlusten av några programkontroller under pre-scan som FOR-NEXT-nesting.

Låt oss arbeta oss genom XB-manualens föreskrifter. Några av dessa hjälper dig att förstå hur XB utför sitt arbete. Min erfarenhet är att vissa av restriktionerna inte behöver följas så strikt som de be-

skrivits, så länge man följer den grundläggande tanken, medan vissa är tvingande, och andra är mitt emellan. De sista är de som man skulle kunna tänka sig i en ny version av XB som gör sakerna annorlunda men fortfarande är i överensstämmelse med manualen. Detta är alltid faran med att använda ospecificerade egenskaper eller odokumenterade funktioner. Det är inte något stort problem med XB eftersom TI hoppade av 99/4A och gjorde XB till ett dött språk som latin. Detta är dock inte helt sant, trots att folk fortsätter att använda den ursprungliga XB så finns det utökade alternativ, som fortfarande är GPL-baserade med användande av olika GROM-simuleringar (Mechatronic XB II+ v.110 och Triton Super-XB v.120 ö.a.). Det har även kommit "utility"-program som automatisk utför vissa uppgifter som vi har diskuterat här och ytterligare saker. Det som har saknats stort är någon publicerad tolkning av hur just XB utför sina interna uppgifter. Jag har aldrig sett någon källkod till XB, varken TI-original eller rekonstruerad. Jag känner inte till om TI-källkoden har läckt ut, eller om den överhuvudtaget fortfarande existerar, och författarna till XB-utilities tycks inte känna någon önskan att dela med sig av de detaljer som de funnit.

(1) DATA-kommandon :-

Pre-scan letar upp det första DATA-kommandot och sätter XB:s datapekare för användning av den första READ-operationen. Om första DATA saknas inom pre-scan, måste RESTORE anropas före första READ så att datapekaren sätts på ett riktigt sätt. Om detta görs kan XB-manualens råd ignoreras.

(2) Variabler :-

Varje variabel måste sökas minst en gång, i annat fall kommer inte XB att ha med den i sin länkade lista av pekare till namn och lagringsarea. Detta kan vara orsaken till några verkligt otrevliga programbuggar, där "syntax error"-meddelandet fås av en programrad som ser helt riktig ut. Orsaken kan vara felaktig placering av pre-scan

switch-kommandot som har lämnat interpretatorn med något som borde vara en variabel, men inte kan hitta den som detta. Att vara en icke-variabel är ett mycket värre öde än att bara vara satt till noll.

OPTION BASE 1 påverkar hur lagringsutrymmet tilldelas och föregår normalt alla fält-variabler. Om det göms från pre-scan med !@P- så kommer default-värdet 0 att gälla.

Manualen säger att den första förekomsten av varje variabel eller fältvariabel (array) måste finnas inom pre-scan. Detta kan synas nödvändigt för fält-variabler, i DIM-kommandon, om du inte använder default-dimensioneringen (utan DIM). Enkla variabler kan sökas av pre-scan varsohelst bara det blir åtminstone en gång. Prova detta lilla programexempel

```
100 CALL CLEAR :: !@P-
200 I=1 :: PRINT I
300 !@P+
400 I=2
```

Kör detta program så blir det inga problem. Sudda rad 400 och se vad som händer. Nu får du "syntax error" på en rad som i sig själv är fullständigt riktig.

(3) Underprogram :-

XB-manualen rekommenderar att det första CALL av något underprogram skall tas med inom pre-scan. Det verkar som i de fall första CALL av ett användardefinierat underprogram uppträder efter dess eget SUB (från ett senare SUB-program), kommer det nödvändiga medtagandet av SUB och SUBEND att vara tillräckligt.

Inbyggda underprogram har naturligtvis inte några tillhörande SUB-kommandon, så CALL måste tas med i pre-scan om programmet körs normalt. Prova detta exempel.

```
100 FOR I=1 TO 1000 :: !@P-
200 CALL SCREEN(12)
300 !@P+
400 NEXT I
500 SUB ANYTHING :: CALL SCREEN(3) :: SUBEND
```

Detta kommer att fungera trots att SCREEN endast nås av pre-scan i underprogrammet. Sudda rad 500 och det kommer att krascha om du har XB med 32K minnesexpansion. I VDP-RAM (med endast grunddatoren) kommer det fortfarande att utföras men endast med ungefär 1/3 av hastigheten.

Vad händer om en fält-variabel (array) överförs via parameterlistan till ett underprogram, men inte dimensioneras förrän i ett senare underprogram? Om du kommer ihåg diskussionen om överföring av fält-variabler, blir du inte överraskad av att finna att XB är tillräckligt smart att spara tilldelningen av utrymme för fält-variabeln tills det kommer förbi en verklig programdel. Prova detta lilla exempel

```
100 CALL SECOND
200 SUB FIRST(A()) :: PRINT
A(20) :: SUBEND
300 SUB SECOND :: !@P-
400 DIM A(20) :: CALL FIRST(A())
500 !@P+
600 SUBEND
```

Detta program kraschar med "syntax error" på rad 400 i SECOND. Om du suddar bort pre-scan-kommandot så kommer programmet att fungera. Om du dessutom tar bort DIM A(20):: på rad 400 så kommer programmet att krascha på rad 200 med ett "subscript error".

(4) DEF, SUB och SUBEND :-

Gör som manualen säger. XB behöver dessa inom pre-scan för att sätta upp saker på rätt sätt.

Pre-scan switch-kommandona har inte stor effekt om inte programmet är av betydande längd, så det är inte värt att bekymra sig om det för mycket i början av ett programs utveckling utöver att vara beredd på möjligheten. XB-manualens supplement (sid 10) visar hur alla variabler och underprogram kan deklarerars genom att samla ihop dessa för att minimera längden av pre-scan, genom att använda GOTO för att hoppa över listan till första körbara kommandot. Man kan klara detta genom att

DIJIT AVPC EPROM PWRUP2A

av Jan Alexandersson

Jag har nu fått en ny EPROM till mitt 80-kolumnskort från DIJIT. Denna nya EPROM PWRUP2AP från feb 1990 kan fås från DIJIT för 10 dollar. Observera att denna version 2 har 4 kbytes (2732) jämfört med 2 kbytes (2716) för version 1, så bygel J2 skall sättas i läge A (till vänster). Den sista bokstaven P i namnet betyder PAL 50 Hz så det finns även en N-version med NTSC 60 Hz för USA.

De viktigaste nyheterna är förbättrad hantering av diskbufferten i VDP-RAM och interrupt-hanteringen. Detta betyder att Alexander Hulpkes XHI och Texaments The Missing Link fungerar utan problem. Program med SPEECH fungerar nu mycket bättre än tidigare.

Vid normal power-up kommer datorn in i AVPC-mode vilket markeras med text över färgbalkarna. I detta fall reserveras 2 bytes högst upp i VDP-minnet vilket betyder att SIZE i Extended Basic nu visar 11838 bytes stack space (version 1 hade 11824 bytes).

Genom att ha SHIFT-CTRL nedtryckta under reset startas datorn i 99/4A-mode vilket motsvarades av program-snutten SET99/4A för version 1. Denna 99/4A-mode reserverar inget överst i VDP-RAM vilket gör att endast de 4 grafikmoderna från 9918A/9929A (T1,MC,G1,G2) kan användas. SIZE i Extended Basic visar nu 11840 bytes stack space.

XB inte gör en fullständig syntaxkontroll förrän raden senare körs. Detta är den enda styrka som man kan tillskriva XB:s misslyckande att avvisa alla ogiltiga rader när de knappas in. Samma sak kan användas inom ett underprogram, och jag har funnit detta mycket praktiskt med samma GOTO för att reservera ett gömställe där man kan stoppa undan subrutiner anropade med GOSUB inom underprogrammet.

Denna nya EPROM har förbättrats så att P-code Editorn för UCSD Pascal kan användas. TELCO i 80-kolumner fungerar nu perfekt medan TELCO 40-kolumner fortfarande visar en kopia av översta raden under statuslinjen, men vem vill använda 40 kolumner när det finns 80 kolumner. Även PAGE-PRO fungerar nu tillfredsställande.

Ett nytt problem som dykt upp med denna nya EPROM är laddningen av Disk Manager V till Myarc HFDC. DM V kan inte laddas från Funnelweb med XB vare sig i AVPC- eller 99/4A-mode. Jag har endast lyckats ladda via Editor/Assembler i 99/4A-mode (AVPC-mode går ej).

FUNNELWEB 4.21 APR/02/90

finns nu i programbanken, 3 skivor för 99/4A respektive 4 skivor för 9938. Ändringar enligt nedan:

Vn 4.20 Oct/24/89 ... Murphy struck the distribution archives, and a number of enhancements and bug fixes have been made to give.

Vn 4.21 Dec/20/89 ... Original issue

Vn 4.21 Mar/11/90 ... Fixes to 40-col EE file to stop potential lockup with QD after return, and to eliminate junk from end of LF line after file marking. Assembler AT file revised to improve file name recognition, transfer, and default. FWDOP/EASM updated to cover this. Nonfatal bug with EoF and auto-paged read to View buffer fixed in 80-col DR. Format dot command error fixed in -READ-ME.

Vn 4.21 Apr/02/90 ... Fixes to DSRLNKs in AT,DP,LH to allow re-entry to same DSR. Original TI ASSM/EDITA1 bug as fixed in later Vn 4.13 issues for other programs in the package.

FLEXSKIVA PROGRAMBITEN

1987 - 1988 - 1989

Du kan nu beställa samlingsskivor från PB-87, PB-88 och PB-89. Pris 30 kr styck som sätts in på föreningens postgiro 19 83 00-6. Skivorna har i-bland fyllts ut med icke publicerat material för att få en fullmatad skiva. SVERIGE-programmet av Sören Bernle har tidigare ej publicerats vilket även gäller PRK-PRINT och PRK-SNABB2. MICRO- och PB-filerna är kataloger över programtester i Micropendium respektive Programbiten avsedda för TI-BASE. HUNT och CHECKPGM från 1987 har flyttats till 1988 års skiva beroende på att allt inte fick plats på den första skivan.

Du kan även beställa samlingsskivor från NN-83, PB-84, PB-85 och PB-86.

PROGBIT-87

FIL=17 LED=23 ANV=337

filnamn	sekt	typ	längd	p
---------	------	-----	-------	---

42-SPELET	33	PROGRAM	7995	*
ALFASLANG	27	PROGRAM	6486	*
AUTOFILE	8	PROGRAM	1715	*
CHAR_SET	9	DIS/FIX	80	*
CHAR_SETS	23	DIS/VAR	80	*
DIETPROG	8	DIS/VAR	80	*
HCHAR:ASS	29	DIS/VAR	80	*
HCHAR:REL	5	DIS/FIX	80	*
HCHARTEST	3	PROGRAM	476	*
LOAD/BERNL	15	PROGRAM	3330	*
MENY	3	PROGRAM	504	*
MINEFIELD	19	PROGRAM	4535	*
MITEC;C	34	DIS/VAR	80	*
PASK	9	PROGRAM	1894	*
TECKENSME	27	PROGRAM	6487	*
TIWRLOG1S	50	DIS/VAR	80	*
TOADY	33	PROGRAM	8174	*

HORIZON RAM-DISK FÖR MYARC GENEVE

Horizon kan till skillnad mot alla andra RAM-diskar även användas med Myarc Geneve, men klarar då normalt endast 260 kbytes (1140 sektorer). Med patchning av systemfilerna kan även större klaras, men det krävs en ny patch för varje ny version av Myarc MDOS (kontakta J.Schroeder i USA). Du kan även lägga till en Phoenix BOOT-drive för 384-800 kbytes RAM-diskar med Geneve.

PROGBIT-88

FIL=21 LED=26 ANV=334

filnamn	sekt	typ	längd	p
---------	------	-----	-------	---

AVRUNDNING	2	PROGRAM	83	*
BASICPRO1	3	PROGRAM	414	*
BASICPRO2	3	PROGRAM	267	*
CHECKPGM	89	DIS/VAR	80	*
DUBBELSPEL	21	PROGRAM	4954	*
HUNT	30	PROGRAM	7364	*
LISTSKYDD	4	PROGRAM	659	*
LOUDSPEAK	27	PROGRAM	6486	*
POKEVPEEKV	11	PROGRAM	2388	*
PRK-CALL	11	PROGRAM	2531	*
PRK-MERGE	5	PROGRAM	787	*
PRK-PRINT	3	PROGRAM	503	*
PRK-SNABB2	14	PROGRAM	3309	*
SATELLIT	10	PROGRAM	2211	*
STA-CALL	6	PROGRAM	1192	*
STA-MERGE	4	PROGRAM	581	*
STARTSPRIT	4	PROGRAM	532	*
STYRSPRITE	7	PROGRAM	1411	*
SVERIGE	49	INT/VAR	254	*
TEX	23	PROGRAM	5523	*
TEX-LOAD	6	PROGRAM	1116	*

PROGBIT-89

FIL=22 LED=20 ANV=340

filnamn	sekt	typ	längd	p
---------	------	-----	-------	---

ARTLES/G6	2	PROGRAM	219	*
CR*ADDER	3	PROGRAM	497	*
FILE*PRINT	8	PROGRAM	1722	*
FILSCR/G7	3	PROGRAM	322	*
LABEL	12	PROGRAM	2646	*
LF*STRIPER	3	PROGRAM	381	*
LOAD/G6	3	PROGRAM	315	*
LOAD/G7	3	PROGRAM	412	*
LOADER/JAN	8	PROGRAM	1642	*
LOTTO/SS	15	PROGRAM	3418	*
MICRO/D	79	INT/FIX	85	
MICRO/S	4	INT/FIX	255	
MOTION/G6	3	PROGRAM	358	*
PAYMENT	8	PROGRAM	1597	*
PB/D	66	INT/FIX	85	
PB/S	4	INT/FIX	255	
PRINT-MP/C	4	DIS/VAR	80	
PRINT-RD/C	4	DIS/VAR	80	
R10	37	PROGRAM	9197	*
R11	49	PROGRAM	12214	*
STYRA-DM99	9	PROGRAM	1995	*
ZOMBIE	11	PROGRAM	2396	*

FÖRENINGENS NYA STADGAR

STADGAR FÖR FÖRENINGEN PROGRAMBITEN

Arsmötets beslut den 23 feb 1985.
Rev. 21 feb 1987 samt 10 mars 1990.

KAPITEL 1. FÖRENINGENS SYFTE.

Föreningen Programbiten har till syfte att samla personer med intresse för programmerbara räknare (företrädesvis TI-59) och person- och hemdatorer (företrädesvis TI-99/4A) i sådana former att unga och äldre kan hjälpa varandra med avseende på maskiner, tillbehör, program, användningsområden mm. Föreningen är öppen för alla.

Föreningen utger, för att främja detta syfte, medlemstidningen PROGRAMBITEN, som i huvudsak avses behandla nämnda datorer och räknare.

Föreningen skall vara ideell. Genom dess verksamhet skall sådana aktiviteter främjas som stimulerar och utvecklar medlemmarnas intresse för datorteknik i allmänhet och nämnda datorer och räknare i synnerhet.

KAPITEL 2. FÖRENINGENS SAMMANSÄTTNING.

Som medlem räknas den som betalt av årsmötet fastlagd medlemsavgift.

Medlem, som bryter mot föreningens stadgar eller på annat sätt uppenbarligen skadar föreningen eller dess syften, kan av styrelsen med enhälligt beslut uteslutas ur föreningen.

KAPITEL 3. FÖRENINGENS VERKSAMHETSÅR.

Programbitens verksamhetsår och funktionärernas mandattid är kalenderåret (1 januari-31 december).

KAPITEL 4. FÖRENINGENS EXEKUTIVA ORGAN.

Programbitens exekutiva organ är:

- a) Föreningsstämman, -ordinarie (=årsmöte)
Föreningsstämman, -extra
- b) Styrelsen
- c) Revisorerna
- d) Arsmötets valberedning

KAPITEL 5. PROGRAMBITENS FUNKTIONÄRER.

Arsmötet väljer följande funktionärer:

- a) Styrelse, bestående av ordförande och kassör, vilka båda utses till befattning av årsmötet samt därutöver minst tre och högst sju ledamöter i övrigt.
- b) Två revisorer och en revisor-suppleant.
- c) Valberedning bestående av två ledamöter, varav en sammankallande.

Övriga funktionärer utses av styrelsen.

Programbitens funktionärer skall vara medlemmar i föreningen. Undantagna är revisorerna, som kan vara utomstående.

Funktionärerna är ansvariga inför det organ som utsett dem.

KAPITEL 6. STÄMMAN.

Stämman är föreningens högsta beslutande myndighet och bildas av föreningens samtliga medlemmar, som alla har en röst vardera. Stämmans ordförande har utslagsröst utom i valfrågor. Minst en stämma skall hållas årligen.

Arsmötet, dvs ordinarie stämma, skall hållas senast 15 mars.

Stämma är beslutsmässig med det antal medlemmar som infinner sig, förutsatt att kallelse med tid och plats skriftligen utsänts till medlemmarna antingen genom tidningen eller separat, senast 15 dagar innan stämman.

Årsmöte sammankallas av styrelsen.

Extra stämma sammankallas av styrel-

sen eller av revisorerna på eget initiativ eller när minst 30 medlemmar så fordrar.

Ingen får delta i beslut eller leda sammanträde när fråga om ansvarsfrihet för honom/henne själv behandlas.

Vid årsmöte förekommer ärenden, som av styrelsen hänskjutits till årsmötet eller som skriftligen inlämnats till styrelsen av medlem senast sista dagen av det verksamhetsår som skall redovisas på mötet. Beslut kan endast fattas i frågor som upptagits på föredragningslistan.

Årsmötet skall behandla följande frågor:

- a) Styrelsens berättelse över föregående verksamhetsår.
- b) Revisorernas berättelse över föregående verksamhetsår.
- c) Ansvarsfrihet för styrelsen och funktionärerna.
- d) Val av styrelse med funktionärer.
- e) Val av revisorer.
- f) Val av representanter i valberedningen.

Protokoll skall föras som beslutsprotokoll och skall justeras inom 20 dagar efter mötet av sammanträdesledaren jämte två av årsmötet utsedda justeringsmän. Protokollet skall i justerat skick utsändas till samtliga medlemmar.

KAPITEL 7. FÖRENINGSSTYRELSEN (FS).

FS består av föreningens ordförande, kassör, sekreterare och redaktör samt minst en ledamot därutöver. FS är beslutsmässig, om minst tre ledamöter är närvarande och om kallelse med tid och plats meddelats senast tre arbetsdagar innan mötet. FS ordförande har utslagsröst vid lika röstsiffror.

Protokoll skall föras. Protokollet justeras av ordförande och utskickas till styrelsemedlemmarna.

Det åligger FS:
att inför årsmötet ansvara för hela föreningens förvaltning och verksamhet.
att verkställa årsmötets beslut.
att handha och ansvara för förening-

ens medel, bereda inkomna förslag, handha föreningens korrespondens samt i övrigt sköta löpande ärenden. att vid behov förätta interimsväl. att före utgången av varje verksamhetsår tillse att gällande stadga är utskriven.

att senast 20 dagar före ordinarie stämma överlämna föreningens årsredovisning till revisorerna.

KAPITEL 8. FÖRENINGENS REVISORER.

Revisorerna skall granska föreningens förvaltning och verksamhet. skall senast fem dagar före möte om ansvarsfrihet till FS inlämna revisionsberättelse.

Revisorerna äger rätt att närhelst de så önskar taga del av samtliga räkenskaper, protokoll och andra handlingar. att begära och erhålla upplysningar rörande verksamhet och förvaltning att övervara samtliga föreningsorgans sammanträden med yttranderätt. att inkalla samtliga föreningsorgan.

KAPITEL 9. FÖRENINGENS VALBEREDNING.

Årsmötets val förbereds av valberedningen.

Valberedningens förslag skall inlämnas till FS senast tolv dagar innan årsmöte. Förslaget bör helst distribueras med kallelsen till årsmötet.

KAPITEL 10. ÄNDRING AV STADGARNA.

Förslag till ändring av dessa stadgar inlämnas skriftligen till FS senast dagen av det verksamhetsår, som skall redovisas på årsmötet, vid vilket FS utlåtande skall föreligga. För slutgiltigt bifall fordras beslut på två av varandra följande stämmor, varav minst ett årsmöte. På den stämma som sist hålles skall beslut ha fattats med minst 2/3 majoritet.

KAPITEL 11. TOLKNING AV STADGARNA.

Årsmötet äger rätt att tolka dessa

THE PROBLEM WITH PRINTERS

by Jim Peterson, Tigercub, USA

When a program is written on one TI-99/4A console, it's a pretty sure bet that it will run on any other TI-99/4A console - unless the programmer has used some of the special features of the CorComp Disk Controller, Super Extended Basic, or whatever. But when a programmer writes a program to output to his own printer, it is by no means certain that it will work with your printer. As far as printer compatibility is concerned, it's a jungle out there. Anarchy, chaos and total confusion!

To begin with, if the printer has a parallel port it must be opened with "PIO", otherwise with "RS232" followed by the baud rate - or something else again for AXIOM. And you may have to add .LF to suppress line feeds or .CR to suppress carriage returns. Next, its output and its response to control codes is partly controlled by those idiotic, microscopic, inaccessible, fragile-looking inventions of the devil called dip switches. And finally, the output is mostly controlled by the printer control codes in the program itself.

stadgar. FS kan i samråd med revisorerna avgöra i årsmötets ställe.

KAPITEL 12. FÖRENINGENS UPPLÖSNING.

Beslut om föreningens upplösning kan endast fattas på stämma. För slutgiltigt bifall fordras beslut på två på varandra följande stämmor, varav minst ett årsmöte. På det möte som sist hålles skall beslutet ha bifallits med minst 2/3 majoritet.

Kallelse till stämma som behandlar föreningens upplösning skall innehålla ett av initiativtagarna styrelsen, revisorerna eller grupp av medlemmar om minst 30- motiverat förslag till nedlägningsbeslut med ett tydligt förslag om fördelningen av föreningens tillgångar. ■

Somewhere among the thousands of publications on computers, someone must have written a comprehensive guide to writing and modifying software for printer compatibility. If anyone knows of such, please tell me! I have read literally thousands of user group newsletters over the past several years, and have seen many mentions of "fixes" to various problems, but never a detailed article. I have called printer manufacturers, and they have been most helpful in suggesting that I buy one of their expensive manuals for each of their models. I have talked to programmers with much more experience in writing printer programs than I have, and they tell me it is very difficult, even with the manual at hand, to modify a program for a particular printer without having access to that printer for testing.

I have no experience in programming for any printer other than my trusty old Gemini 10X, and my few attempts to modify programs for other printers have mostly ended in failure. However, I have borrowed several manuals and attempted to chart the differences. I had hoped to compile and publish a complete conversion chart, until I realized the complexity of the problem. Anyway, perhaps I can pass on a few tips to programmers, to help them make their programs as widely compatible as possible, and possibly I can give users a little bit of guidance to help in modifying programs to suit their printer.

In the following, in order to be brief, I have mentioned control code sequences by their ASCII numbers, such as 27 66 1. This would be programmed as CHR\$(27);CHR\$(66);CHR\$(1) or, since ASCII 66 is within the printable range, it might be CHR\$(27)&"B"&CHR\$(1). 27 77 n means that for n you substitute an ASCII, within an allowable range, according to what you want to accomplish.

There seem to have been four systems of printer control codes used with

the printers commonly found in the TI world - Epson, Micronics, Axiom, and Okidata. The Micronics people tell me that they "used the Micronics emulation until the introduction of the current NX series, when they switched to the IBM emulation". The IBM emulation appears to be the same as the Epson mode except that it has a different set of special character symbols - in fact, many current Epson-compatibles have an optional IBM mode.

The Micronics mode and the Epson mode are quite similar, although with aggravating differences. Okidata and Axiom are way out in left field. Since Micronics, Epson and Panasonic (which is basically Epson-compatible) seem to be by far the most popular in the TI community, and most software is written for them, it might be wise to avoid the Okidata. I have also seen mention of problems with Diablo and Centronics, but I have no information on those.

Any of the ASCII from 0 to 127 can be used as a printer control code. If the ASCII is above 31, it must be preceded by ASCII 27, known as the escape code, which is universally used to alert the printer that the following ASCII codes are to be interpreted as controls rather than printed as characters. If the printer recognizes an ASCII below 31, or one or more ASCII immediately following ASCII 27, as a valid control code, it acts upon them but does not print them. This is why, if you insert "control U" codes in a line of text, the text will be shifted left. However, if the codes are not recognized as valid, the ASCII below 32 or above 126 are printed as a blank space, the others are printed as the character they represent. This is why that puzzling E, G, S or whatever shows up on the first line of a printout, if a program is not compatible with your printer.

Some printer commands require a sequence of three or more ASCII codes, of which the first is 27, the second could be anything above 31 and the remainder could be anything at all. If your printer does not recognize the second ASCII as valid, but then

comes to an ASCII below 31 which it does recognize, it acts on that ASCII as if it was a single command - which is why your printer sometimes "goes crazy".

The ASCII below 27 are quite standardized, and many of them have names, such as BEL for 7 (activates buzzer) which are also commonly used in telecommunications. ASCII 10 (line feed), 12 (advance to next top of form) and 13 (carriage return) seem to be universally recognized. For some reason, Panasonic owners seem to have trouble with line feeds when running programs written for other printers.

ASCII below 27 are not preceded by the escape code 27. Some printers will optionally recognize 14 (double width for one line) and 15 (cancel 14) preceded by 27, but programmers should avoid this since other printers will treat the 27 as a blank space. The NX-10 recognizes 27 10 as a command to reverse the paper one line and 27 12 to reverse to top of page.

The escape code 27 can be input from the TI-99/4A keyboard by depressing the CTRL key and the period key together - the actual ASCII is 155 but printers, other than the Axiom, will accept it as 27 (not ok in IBM-mode ed. note). This is handy when opening the printer in immediate mode or writing a routine for your own use, but should be avoided in programs being distributed because the character prints out as a blank space which will probably confuse anyone trying to modify the program.

I have studied the manuals, and attempted to chart some of the codes, for the Gemini 10X, SG-10 and Star NX-10; Epson FX-80, FX-85/185 and its IBM mode; Panasonic KX-P1080; MX Graftrax Plus; Brother M-1009; Seikosha 550A or 550TI by Axiom; and Okidata (model unknown). Due to differences in terminology, it is not easy to relate them to each other.

The IBM mode of the Epson FX-85 seems to be entirely compatible with its Epson mode except that it lacks some features. The FX-80 seems to be

entirely compatible with the FX-85 except lacking NLQ and a few specialized codes (and I did not get into comparing graphics capabilities of any of these printers). The Brother M-1009 is also apparently highly compatible. The MX Graftrax Plus, another Epson model, is entirely compatible but lacks some features (no graphics capability?). The Panasonic KX-P1080 is very compatible but also has several unique codes of its own for setting tabs and spacing, etc. The Star NX-10 also seems to be in complete agreement with the FX-85, but with some additional codes for unique features such as reversed paper feed. I would guess that in actual practice these may not be as compatible as they seem. And of course, any maker's newer or more expensive models have additional codes for features not found on older or cheaper models.

The Seikosha 550A or 550TI, made by Axiom, recognizes a few of the common codes between ASCII 7 and 14, but from there on it has entirely its own system of codes; many of these same codes are used by Epson/Micronics for entirely different purposes, so running a program written for either on the opposite printer can be guaranteed to produce pure garbage. Okidata likewise recognizes a few of the low codes and then goes into its own system, frequently in direct conflict with the Epson standard; due to the terminology used in its manual, I am not sure what some of them do.

The Gemini 10X, made by Star Micronics, has long been superseded by newer models, but stocks are still being sold by discounters. It has been a sturdy workhorse, long popular with TI owners, and a great many programs have been written using its printer control codes. These are 90% compatible with Epson - but that other 10% causes a great deal of trouble. The differences are described below. The Star SG-10/15 was a transitional hybrid, switchable by dip switch 2-2 between the Micronics mode and the IBM mode. The Micronics mode is completely compatible with the Gemini 10X (except for download characters) and with some additional features - NLQ

and proportional printing, and a slashed zero option. The IBM mode seems to be very compatible with the Epson standard. This printer was superseded by the Star NX-10, which is again Epson compatible.

A major incompatibility between programs written for the Gemini 10X or SG-10, and Epson-compatible printers, is that Micronics recognizes 27 66 1 to select pica, 27 66 2 to select elite and 27 66 3 for compressed, and on the SG-10 also 27 66 4 to select NLQ and 27 66 5 to cancel it. On Epson/IBM printers, 27 66 is the beginning of a series of codes used to set vertical tabs. Actually, since pica is the default, there is no need to program for it except to cancel condensed print, for which purpose 18 is recognized by both Micronics and Epson. (Avoid using 27 80 to return to pica because Micronics does not understand it and might misinterpret it to change default tabs.) Similarly, 15 will select condensed print on both the Micronics and Epson. Unfortunately, there is no compatible code for elite; Epsoms use 27 77 to select elite, but 27 77 n is used by the 10X, SG-10 to set the left margin n spaces, so that misinterpreting these codes can be catastrophic! The Epsoms use 27 108 n to set the left margin, but this is not recognized by Micronics.

The other major difference is 27 51 n which sets the line feed to n/144" on the 10X and on the SG-10 in Star mode, but to n/216" on Epson compatibles and on the SG-10 in IBM mode. The "fix" here is to multiply the value of n by 1.5 when running a Micronics program on an Epson printer. The same applies to 27 74 n which sets a one-time line feed of n/144" or n/216".

Micronics uses 27 82 n to set the margin at the top of the page, but Epson recognizes this as a command to switch to one of the international character sets, which can produce some interesting results. The Epsoms use 27 114 n to set the top margin, but Micronics doesn't know this one. Micronics uses 27 55 n to select an international character set, but the Epson will read

27 55 as a command to cancel 27 54 which selected a special character set. I'm not sure what that means, but the results will surely be undesirable.

There may also be a conflict between the Micronics 27 98 n, which performs a one-time tab of n columns, and the Epson 27 98 n n 0, which "sets vertical tabs in channel", whatever that means.

Several codes, common to both Epson and Micronics, use 1 as the 3rd ASCII to turn on a feature and 0 to cancel it. For instance, 27 87 1 turns on double width (expanded) print and 27 87 0 cancels it. Also, subscript is selected by 27 83 1 and superscript by 27 83 0. Some of the Epson/IBM compatibles will accept either an ASCII or numeric 0 or 1 (i.e., "1" or CHR\$(1)) for that third code, for which reason you will often see program coding such as CHR\$(27);"W1". These should be avoided when programming for general distribution, because the older Micronics recognize only the ASCII. If I understand my notes from the manual correctly, the Panasonic KX-P1080 also accept ASCII 129 or 177 in lieu of ASCII 49 or "1" and ASCII 128 or 176 in lieu of 48 for "0" !

According to the manuals, ASCII 141 can be substituted for ASCII 13 on the Brother M-1009, and ASCII codes 128 to 255 can be substituted for 0 to 127, respectively, on the Epson - but there seems to be no good reason to confuse the other printers by using those!

Different printers also have different sets of symbols in ASCII 160-254. The Gemini 10X and SG-10 in Star mode has one set, the SG-10 in IBM mode has an entirely set which I presume is also on the Epson in IBM mode, and I think that the Epson has still a different set. This causes problems when running some banner or graphing programs which access these characters. Different printers also vary in the number of international character sets available and the sequence of their access codes.

I have never gotten involved in graphics printing, and I failed to

chart all the graphics codes when I had borrowed manuals available, so I cannot comment on compatibility here. I have not heard of any problems except that some Axiom models are apparently incapable of graphics, and there is also sometimes a problem with thin white horizontal lines through the picture - possibly because of the n/144" and n/216" difference in line spacing between Micronics and Epson? (also n/180" in some printers. ed. note)

With downloadable characters, we find another jungle which I'm not too anxious to explore. The Gemini 10X has a quite simple and efficient method, and I once published in Micropendium a DOWNCHAR program to design these characters on screen, dump them to the printer for editing, and save them to disk. I have also written a routine which will convert a sequence of any length of standard or reidentified screen characters into a D/V 80 file of download character printer codes.

But, here the SG-10 Star mode departs from compatibility with the 10X. Its system offers much greater capabilities but is also quite complex and entirely different. I tried, and failed, to convert my routine for use on the Epson FX-85; its system is somewhat similar to that of the SG-10 but again different. I am told that the Epson RX-80 does not support downloadable characters, the LX-80 only allows 6 and some Panasonics only allow 40 of them. I have seen an article describing a method of creating downloadable NLQ characters but unfortunately the name of the printer being used was not mentioned.

This article is obviously incomplete and probably inaccurate. Perhaps it will inspire someone to write something better. In the meantime, programmers could help out a great deal by putting REMs in their programs giving the name of the printer they are writing for, and REMs after every printer control command indicating its purpose. I regret that I have not been in the habit of doing that!

(REFERENCE:TI*MES 16 p.40, ed.note)■

SWEDLOW EXTENDED BASIC

X	X	BBBB	# 9 to 11
X	X	B	B
X		BBBB	By
X	X	B	B
X	X	BBBB	Jim
X	X	BBBB	Swedlow

(This article originally appeared in the User Group of Orange County, California ROM)

PRODUCT REVIEW - NEATLIST

NEATLIST will take any program in memory and list it to any legal device (printer, disk, etc.) with the same speed as the LIST command. It has, however, several advantages:

-- Each line of the listed program can be up to 255 characters wide. You can, therefore, use your printers powers (condensed print, expanded print, etc.).

-- You can send function codes to your printer enabling you to access its features.

-- NEATLIST can print out a list of variables used in the program and, if you opt, a list of line numbers where each variable is used.

-- It is an assembly language program that loads thru XB.

The list is nicely designed. Consider this program:

```
10 CALL CLEAR :: PRINT ::  
DIM A(5)  
20 FOR I=1 TO 5 :: A(I)=I  
:: NEXT I
```

NEATLIST would produce this:

```
10 CALL CLEAR ::  
PRINT ::  
DIM A(5)  
20 FOR I=1 TO 5 ::  
A(I)=I ::  
NEXT I
```

If you have memory expansion, a printer and a disk drive, you will find NEATLIST quite useful for debugging and pre scan (see below).

CALL PEEK

This is from the NEATLIST documentation. CALL PEEK(8198,A) will let you know if a CALL INIT has been performed. If A returns as 170 then it has, any other value indicates that it has not.

ASCII CHART

I was asked how I learned XB's tokens for last month's list of ASCII codes. This little program did it:

```
10 OPEN #1:"DSK1.TEST",OUTPUT,  
VARIABLE 163,DISPLAY  
20 FOR I=1 TO 254  
30 PRINT#1:CHR$(0)&CHR$(I)&  
CHR$(I)&CHR$(0)  
40 NEXT I :: PRINT #1:CHR$(255):  
CHR$(255) :: CLOSE #1
```

After entering this program, enter the following commands:

```
>RUN  
>NEW  
>MERGE DSK1.TEST  
>LIST "PIO"
```

The MERGE command can take 5 to 10 minutes. You will need to delete some lines to get a good listing as some of the lines will send your printer function codes. When you are done, compare your program list to the ASCII chart.

PRE SCAN

You load your program, enter RUN and then . . . nothing. Finally your program starts to execute. On a short program this wait is not noticable but on a long one it can seem endless. Why the delay?

Your 99/4A is going thru your program, line by line, and allotting memory space. It is noting each variable used, each subprogram CALLED, the first DATA line, DEF statements, DIM statements, etc.

Futhermore, it is making an unduplicated list. Suppose you use the

variable A 123 times in your program. The first time your 99/4A notes it and makes memory space. The other ONE HUNDRED AND TWENTY TWO TIMES it checks, notes that it already knows about this variable and moves on. Even at the speed of the 4A, this takes time.

XB has some tools to control pre scan. !@P- turns it off and !@P+ turns it on. The following items must be in the range of the pre scan:

-- At least one use of each variable.

-- At least one use of each CALL statement. For example, if you use CALL CLEAR five times, the first use must be within the pre scan.

-- All DEF, SUB, SUBEND, DIM and OPTION BASE statements.

-- The first DATA statement in the program.

!@P+ must be on a line by itself while !@P- can be at the end of a multi-statement line.

There is also a short cut -- pre scan does look at CALL statements but it doesn't check validity. Therefore you can do something like this:

```
10 DATA 2,3,4
20 OPTION BASE 1 :: DIM A(17)
   :: GOTO 30 :: CALL HCHAR ::
   :: CALL SPRITE :: CALL SAY
   :: R,S,T,U,V=W :: A$=B$
   :: !@P-
30 ! Program continues
```

Note that the code after GOTO 30 in line 20 will never be executed so it does not need to meet syntax requirements.

You should not activate pre scan until your program is fully debugged. If you forget something, you will get a SYNTAX ERROR.

With a bit of work, you will cut the pre scan time down significantly !

IF THEN AGAIN

IF THEN is the most versatile command in extended basic. This month, another approach.

Often you will use a variable as a flag. You set it to one value at the beginning of a program and then change it if something happens. Then, using IF THEN, you can tell if that 'something' happened and branch your program's operation accordingly.

When doing this, remember two things: First, during prescan, all numeric variables are set to zero. Second, the IF THEN test considers zero as false and all other values as true.

The variable that you are using as a flag, then, does not need to be given an initial value. Further, any change to a non zero value would make it look like a true statement in an IF THEN test. If A is your flag you could do this:

```
IF A THEN 220 ELSE 300
```

If the 'something' had not happened then A would still be zero, the IF test would be false and control would pass to line 300. If the 'something' had happened, A would not be zero, the IF test would be true and execution would branch to line 220.

This should help you simplify your IF THEN testing.

SPEED

You always hear how slow Basic and XB are when compared to assembly language. Alas, 'tis true. There are ways, however, to make your program run a bit faster. Consider this code:

```
10 FOR I=1 TO 200
20 B=3*I
30 IF B>999 THEN 50
40 NEXT I
50 END
```

When you RUN this, your 99/4A notes that there are two variables and

assigns memory space for each. Each time line 20 executes, the memory location for I is found and the value obtained and multiplied by three. Then the location of B is identified and this value stored there (the order of these steps may be a bit different but each step must happen).

Now look at this:

```
10 FOR I=1 TO 200
20 IF 3*I>999 THEN 40
30 NEXT I
40 END
```

Now there is only one variable and the number of steps is reduced. The result: running time drops from 3.80 seconds to 3.01 seconds and memory drops from 92 bytes to 69.

In this example time dropped 21%. If there had been a PRINT command in the loop, running time would be 10 times higher and the time drop under 5%. The reason is that it takes a lot of time to print on your screen.

When would you want to make an intermediate value? When you will use it more than once.

PRINT SEPARATORS

There are any number of ways to display text on your screen. In this and the next two items, I will cover some ideas that might help you. All of these will also work in BASIC except where noted.

Suppose that you want to display menu choices at screen lines 5, 7, 9 and 11. You could use four DISPLAY commands:

```
DISPLAY AT(5,1):"FIRST LINE"
DISPLAY AT(7,1):"SECOND LINE"
DISPLAY AT(9,1):"THIRD LINE"
DISPLAY AT(11,1):"FOURTH LINE"
```

Another way to do this is to combine your DISPLAY's into one command:

```
DISPLAY AT(5,1):"FIRST LINE": :
"SECOND LINE": : "THIRD LINE": :
"FOURTH LINE"
```

Note that the print separator bet-

ween lines is : :. I have seen (and used) :"":. The double quote was inserted to tell the computer to display a blank line. This is not necessary, : : works just as well and uses less memory. In XB you must type a space between the colons or your 4A will think that you are inputting a line separator.

This also works with PRINT statements in both BASIC and XB.

PROMPTS FOR INPUT

You have probably used a prompt with the INPUT command like:

```
INPUT "INPUT NAME":N$
```

The limitations here are frustrating. You can't use a string variable, if you want to leave blank line then you must use a lot of spaces, etc, etc. Sometimes I use a PRINT statement for the prompt and then follow with INPUT. By accident I discovered that INPUT respects any trailing print separator on the preceding PRINT command:

```
PRINT A$;TAB(20);
INPUT B$
```

The INPUT ? will appear at the 20th column.

PRINTING STRINGS

If you are printing two strings, you would normally use a semi-colon (>) between them. When you do this, your 99/4A looks at the current location on the screen and the length of the next print item. If it is longer than the space left on the current line, the string starts at the beginning of the next line.

This can result in annoying blanks in your right hand margin. There is another way:

```
PRINT A$&B$
```

Regardless of the length of A\$, B\$ will start in the next empty column. This may or may not give you the result you wanted, but it does give you options. ■

TIPS FROM THE TIGERCUB

#53

Copyright 1988

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213
USA

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Someone wanted a program to teach how to make change.

```

100 CALL CLEAR
110 DEF IS(X)="$"&SEG$(STR$(X),1,POS(STR$(X),".",1)+2)
120 CALL COLOR(1,2,8,2,2,8,3,2,8,4,2,8,5,2,8,6,2,8,7,2,8,8,2,8,9,2,8,10,2,8,11,2,8,12,2,8)
130 CALL SCREEN(5):: D$=RPT$(" ",112)!programmed by Jim Peterson Apr. 1988 for the public domain
140 GOTO 180
150 CALL KEY :: CALL CLEAR : CALL SOUND
160 A,P,C,G,T,F,O,Q,D,N,S,TT,X,B$,QQ,K,M$,J
170 !@P-
180 DISPLAY AT(2,8)ERASE ALL:"MAKING CHANGE":;:" by Jim Peterson" :: RANDOMIZE : CALL KEY(3,K,S)
190 DISPLAY AT(6,1):"Do you want to":;:" (1) Input amounts": (2) Use random amounts " :: ACCEPT AT(6,16)SIZE(1)VALIDATE("12")BEEP:A :: CALL CLEAR
200 IF A=2 THEN 240 :: DISPLAY AT(2,1):"Price of item? $":;::: :: ACCEPT AT(2,17)VALIDATE(NUMERIC)BEEP:P :: IF P<0 THEN 200
210 DISPLAY AT(4,1):"Amount offered by customer?": "$" :: ACCEPT AT(5,2)VALIDATE(NUMERIC)BEEP:C
220 IF C<>INT(C)AND P-INT(P)<>C-INT(C)THEN DISPLAY AT(23,1):"Even dollars please!" : GOTO 210

```

```

230 IF C<P THEN DISPLAY AT(23,1):"Not enough!" :: GOTO 210 ELSE IF C=P THEN DISPLAY AT(23,1):"No change needed!" :: GOTO 200 ELSE GOTO 260
240 RANDOMIZE :: P=20*RND+.01 :: P=INT(P*100)/100 :: DISPLAY AT(2,1):"Price of item " :: DISPLAY AT(2,16):IS(P+.001)
250 C=INT(4*RND+1)*5 :: IF C<=P OR C=15 THEN 250 :: DISPLAY AT(4,1):"Customer offers $" :: DISPLAY AT(4,18):STR$(C)
260 G=C :: C=C-P :: T=INT(C/10):: C=C-T*10 :: F=INT(C/5):: C=C-F*5 :: O=INT(C):: C=C-O :: Q=INT(C/.25):: C=C-Q*.25 :: D=INT(C/.1):: C=C-D*.1 :: N=INT(C/.05)
270 C=C-N*.05 :: X=C*100 :: TT=0
280 DISPLAY AT(7,1):"OK, start adding change from the price until you reach the customer's amount." :: TT=P+.001
290 IF X=0 THEN 330 :: GOSUB 640 :: IF B$<>"P" THEN M$="No, give pennies to reach "&IS(TT+X*.01):: GOSUB 670
300 DISPLAY AT(11,1):"How many pennies?" :: ACCEPT AT(11,19)VALIDATE(NUMERIC)BEEP:QQ
310 IF QQ=X THEN 320 ELSE IF TT+QQ*.01>G THEN GOSUB 680 :: GOTO 300 ELSE GOSUB 690 : GOTO 300
320 DISPLAY AT(7,1):"You have reached "&IS(TT+X*.01):;::: :: TT=TT+X*.01
330 IF N=0 THEN 380 :: GOSUB 640
340 IF B$<>"N" THEN M$="No, if the cents portion is .10 or .20 or .35 or .45 or .60 or .70 or .85, add a nickel" :: GOSUB 670
350 DISPLAY AT(11,1):"How many nickels?" :: ACCEPT AT(11,19)VALIDATE(NUMERIC)BEEP:QQ
360 IF QQ=N THEN 370 ELSE IF TT+QQ*.05>G THEN GOSUB 680 :: GOTO 350 ELSE GOSUB 690 : GOTO 350
370 DISPLAY AT(7,1):"You have reached "&IS(TT+N*.05):;::: :: TT=TT+N*.05
380 IF D=0 THEN 430 :: GOSUB 640
390 IF B$<>"D" THEN M$="No,

```

```

add dimes to reach .25 or .
50 or .75 or .00" :: GOSUB 6
70
400 DISPLAY AT(11,1):"How ma
ny dimes?" :: ACCEPT AT(11,1
9)VALIDATE(NUMERIC)BEEP:QQ
410 IF QQ=D THEN 420 ELSE IF
TT+QQ*.1>G THEN GOSUB 680 :
: GOTO 400 ELSE GOSUB 690 ::
GOTO 400
420 DISPLAY AT(7,1):"You hav
e reached "&I$(TT+D*.1):;::;
:: :: TT=TT+D*.1
430 IF Q=0 THEN 480 :: GOSUB
640
440 IF B$<>"Q" THEN M$="No,
add quarters to reach an eve
n dollar." :: GOSUB 670
450 DISPLAY AT(11,1):"How ma
ny quarters?" :: ACCEPT AT(1
1,20)VALIDATE(NUMERIC)BEEP:Q
Q
460 IF QQ=Q THEN 470 ELSE IF
TT+QQ*.25>G THEN GOSUB 680
:: GOTO 450 ELSE GOSUB 690 :
: GOTO 450
470 DISPLAY AT(7,1):"You hav
e reached "&I$(TT+Q*.25):;::;
:: :: TT=TT+Q*.25
480 IF O=0 THEN 530 :: GOSUB
640
490 IF B$<>"O" THEN M$="No,
add dollars to reach a mult
iple of five dollars." :: GO
SUB 670
500 DISPLAY AT(11,1):"How ma
ny dollars?" :: ACCEPT AT(11
,19)VALIDATE(NUMERIC)BEEP:QQ
510 IF QQ=O THEN 520 ELSE IF
TT+QQ>G THEN GOSUB 680 :: G
OTO 500 ELSE GOSUB 690 :: GO
TO 500
520 DISPLAY AT(7,1):"You hav
e reached "&I$(TT+O):;::;::;
:: :: TT=TT+O
530 IF F=0 THEN 580 :: GOSUB
640
540 IF B$<>"F" THEN M$="No,
add a five dollar bill to r
each a multiple of ten." ::
GOSUB 670
550 DISPLAY AT(11,1):"How ma
ny fives?" :: ACCEPT AT(11,1
7)VALIDATE(NUMERIC)BEEP:QQ
560 IF QQ=F THEN 570 ELSE IF
TT+QQ*5>G THEN GOSUB 680 ::
GOTO 550 ELSE GOSUB 690 ::
GOTO 550
570 DISPLAY AT(7,1):"You hav
e reached "&I$(TT+F*5):: TT=
TT+F*5
580 IF T=0 THEN 620 :: GOSUB

```

```

640
590 IF B$<>"T" THEN M$="No,
add ten dollar bills to reac
h"&I$(G):: GOSUB 670
600 DISPLAY AT(11,1):"How ma
ny tens?" :: ACCEPT AT(11,16
)VALIDATE(NUMERIC)BEEP:QQ ::
IF QQ=T THEN 620
610 IF QQ>T THEN GOSUB 680 :
: GOTO 600 ELSE DISPLAY AT(1
4,1)BEEP:"That's not enough!
" :: GOTO 600
620 DISPLAY AT(7,1)BEEP:"You
gave the correct change!":"
":"":"
630 DISPLAY AT(14,1):"
PRESS ANY KEY" :: DISPLAY A
T(14,1):" press any ke
y" :: CALL KEY(3,K,S):: IF S
=0 THEN 630 ELSE 200
640 DISPLAY AT(11,1):"Will y
ou now give":" (P)ennies":"
(N)ickels":" (D)imes":" (Q)u
arters":" (O)ne dollar bills
":" (F)ive dollar bills":" (
T)en dollar bills"
650 DISPLAY AT(19,1):""""""
":"""""""
660 ACCEPT AT(11,19)SIZE(1)V
ALIDATE("PNDQOFT")BEEP:B$ ::
DISPLAY AT(11,1):""""""""
":""""""""" :: RETURN
670 FOR J=1 TO 5 :: DISPLAY
AT(20,1):D$ :: DISPLAY AT(20
,1)BEEP:M$ :: NEXT J :: RETU
RN
680 CALL SCREEN(7):: FOR J=1
TO 15 :: CALL SOUND(-99,110
,0,-4,0):: DISPLAY AT(14,1):
"" :: DISPLAY AT(14,1):"You
gave too much change!" :: NE
XT J :: CALL SCREEN(5):: RET
URN
690 DISPLAY AT(14,1):"No, th
at's wrong!" :: RETURN

```

And here's a one-screen tinygram - you could convert this to speech and it would do a better job of making change than most clerks do nowadays!

```

1 !*****
!*      CHANGEMAKER      *
!*      TINYGRAM         *
!*      by Jim Peterson  *
!*****
100 CALL CLEAR :: S$(1)="S"
110 DEF P$(X)="$"&SEG$(STR$(
X),1,LEN(STR$(X))-1)
120 DATA 10,5,1,.25,.1,.05,.
01,CENT,NICKEL,DIME,QUARTER,

```

```

DOLLAR BILL,FIVE DOLLAR BILL
,TEN DOLLAR BILL
130 PRINT TAB(8);"CHANGEMAKE
R";: : INPUT "PRICE? ":P :
: INPUT "AMOUNT OFFERED? ":B
: : IF B<P THEN 130
140 PRINT : : C=B-P : : FOR J=
1 TO 7 : : READ X(J):: A(J)=I
NT(C/X(J)):: C=C-A(J)*X(J)::
NEXT J : : Z=P+.001
150 PRINT P$(P+.001);" OUT O
F ";P$(B+.001):: : : FOR J=7
TO 1 STEP -1 : : READ A$ : :
Z=Z+A(J)*X(J):: IF A(J)>0 TH
EN PRINT "AND";A(J);A$&S$(-(
A(J)>1))&" IS ";P$(Z)::
160 NEXT J : : PRINT : : RESTO
RE 120 : : GOTO 130

```

The Extended Basic Manual did a very poor job of showing us how to use USING. My thanks to Karl Romstedt for telling me how to do it with DISPLAY AT - put a semicolon directly before USING -

```

100 CALL CLEAR : : DISPLAY AT
(12,5):USING "$##.##":1.23

```

Other commands can go either before the AT or after the parameters -

```

110 DISPLAY ERASE ALL BEEP A
T(12,5):USING "$##.##":1.23
120 DISPLAY AT(12,5)ERASE AL
L BEEP:USING "$##.##":1.23

```

However, to output to a printer, put a comma before USING -

```

120 OPEN #1:"PIO" : : PRINT #
1,USING "$##.##":1.23

```

The trouble with PRINT USING "\$##.##" is that it will print nothing but asterisks if the integer contains more digits than the number of # left of the decimal, and will leave blanks between the \$ and the first digit if the integer contains less digits than the number of ## left of the decimal. This algorithm will correctly print dollars and cents values of ANY size, rounded off to the nearest cent and with the dollar sign directly before the first digit or decimal.

```

100 INPUT A : : PRINT USING "
$"&SEG$(RPT$("#",LEN(STR$(IN
T(A))))),1-(INT(A)=0),255)&".

```

```

##":A : : GOTO 100

```

Thanks to Ed Machonis for some of these improvements to the Printall program which was published in Tips #45 -

```

171 DISPLAY AT(8,12):"V.1.2"
190 DISPLAY AT(18,7):"TURN P
RINTER ON!";:"SET TOP OF FO
RM HALF INCH BELOW PERFS"
200 DISPLAY AT(23,8):"PRESS
ANY KEY" : : DISPLAY AT(23,8)
:"press any key" : : CALL KEY
(0,K,S):: IF S=0 THEN 200 EL
SE CALL CLEAR

```

```

330 IF P=1 AND SS$<>"Y" THEN
DISPLAY AT(12,1):"EMPHASIZE
D? (Y/N) Y" : : ACCEPT AT(12,
19)VALIDATE("YN")SIZE(-1)BEE
P:ES : : IF ES="Y" THEN PRINT
#1:CHR$(27);"E";

```

```

390 IF NC=1 THEN 410 : : AV=I
NT(TA/(NC-1)):: DISPLAY AT(1
2,1)ERASE ALL:"COLUMN SEPARA
TION?": "MINIMUM 2": "MAXIMUM
"&STR$(AV)&" AVAILABLE ": "2"
400 ACCEPT AT(15,1)VALIDATE(
DIGIT)SIZE(-2)BEEP:CS : : IF
CS<2 OR CS>AV THEN 400 ELSE
SS=RPT$(" ",CS)
450 LSP=12 : : DISPLAY AT(10,
1): " ": " ": "LINES PER PAGE?
60": " ": " ": " ": " : : ACCEP
T AT(12,17)VALIDATE(DIGIT)SI
ZE(-3):LP : : IF LP<61 THEN 4
90

```

```

460 !DELETE

```

```

470 !DELETE

```

```

480 LSP=72/(LP/10):: PRINT #
1:CHR$(27);"A";CHR$(LSP)

```

```

510 DISPLAY AT(15,1):STR$(LP
)&" lines per page": "with "&
STR$(INT(LSP))&"/72 line spa
cing"

```

```

640 ! DELETE!

```

```

650 IF LEN(M$(IP))<=CW THEN
670 : : T$=SEG$(M$(IP),1,CW):
: CALL SOUND(1000,110,0,-4,0
):: DISPLAY AT(12,1):M$(IP);
" OVER";CW;"CHARACTERS": "TRU
NCATED TO ";T$:"OK?"

```

```

660 CALL KEY(3,K,S):: IF S=0
THEN 660 ELSE IF K<>89 THEN
STOP ELSE M$(IP)=T$

```

MEMORY FULL!

Jim Peterson ■

(0) Innehållsförteckning

(1) Introduktion och Konventioner.....	1
(2) Att Komma Igång.....	2
(3) Topp Meny.....	3
(4) Sysinfo Meny.....	3
(5) Laddnings Meny.....	3
(6) Yttre Enheters Meny.....	4
(7) Färg Meny.....	4
(8) Meny Meny.....	5
(9) XB List Meny.....	6
(10) UL List Meny.....	7
(11) Sysinfo Meny - Spara SYSICON.....	8
(12) Installations Meny.....	8
(13) Avslutande Steg.....	9
(14) Hur Fwb Konfigurerar Fungerar.....	9
(15) Uppgradering av Funnelweb.....	9
(16) Funnelweb Meny Översikt.....	11
(17) Version 4.1x Fwb Filer.....	13
(18) Version 4.2x Fwb Filer.....	14

(1) Introduktion och Konventioner

- a. Dessa instruktioner presenteras som service till TI-99/4A användarna av UGOC / användargruppen i Orange County i Kalifornien. De kompletterar FUNNELWEB dokumentationen. Denna utgåva täcker versionerna 4.10 till 4.21.

Kommentarer och förslag kan sändas till:

UGOC c/o
Jim Swedlow
7301 Kirby Way
Stanton, CA 90680

Om du vill ha ett svar, var snäll och bifoga ett adresserat kuvert och internationellt svarsporto.

- b. Skriv ut och läs FUNNELWEB dokumentationen. Använd Formateraren, skriv ut "-READ-ME" och alla filer som börjar med FWD0C.

- c. Dessa instruktioner förutsätter att du har två diskettstationer och att du vill ladda FUNNELWEB från DSK1. Om din utrustning ser annorlunda ut behöver du anpassa dig till detta förhållande. Där så är möjligt har förslag till anpassning till andra applikationer tagits med.

- d. Om du behöver trycka på en tangent visar "<>" tecknen detta. T ex "Tryck <ENTER>". Om två knappar ska tryckas ned på samma gång visas det så här "Tryck <CTRL C>". <CTRL C> betyder att du ska hålla nere <CTRL> och sedan trycka på <C>. Släpp båda knapparna samtidigt.

- e. I boxar där du anger information (t ex ett filnamn), anger du att du är klar genom att trycka på <ENTER>. Detta anges som "Enter the correct file name". Förslagen indata visas inom citationstecken ("DSKn, etc). Ange INTE citationstecknen. Alltså: "n" i "DSKn" står för diskettstationens nummer (DSK1, DSK2 etc).

- f. I Configure's menyer gör du dina val genom att trycka på den första bokstaven. T ex Sysinfo meny har tre val:

Sysinfo
Quit
Install

- Tryck <S> för Sysinfo, <Q> för Quit eller <I> för Install. Sådana val visas som "Tryck <Q>uit" eller "Tryck <Q> för Återgång.

En förteckning över FUNNELWEB's menyer återfinns på sidan 16.

- g. <CTRL C> används för att backa från den aktuella meny till föregående. Du kan använda <BACK> eller <FCRN 9> i stället för <CTRL C>. <CTRL C> fungerar även i DM 1000 ist f <BACK>.

När dessa instruktioner ber dig att trycka på <CTRL C> för att återgå till föregående meny kan det ibland vara nödvändigt att trycka <CTRL C> mer än en gång.

- h. På många ställen i Konfigurerings programmet kan du trycka <?> eller <FCRN I> för att få upp fönster med hjälptext. Du bör använda denna möjlighet ofta första gången du konfigurerar FUNNELWEB.

- i. På de flesta ställen i Configure programmet finns även snabbkatalogen tillgänglig. Tryck <FCRN 7> för att nå denna funktion.

- j. Följande förkortningar används i dessa instruktioner:

FWB FUNNELWEB
EA Editor Assembler
TI Wr TI Writer
XB Extended Basic

(2) Att Komma Igång

- a. Gör en arbetskopiera av Fwb-disketten. Använd DM1000 för att kopiera de filer du behöver på en tom formaterad diskett. En förteckning över filer och några förslag till arrangemang finns i steg 17 (version 4.1x) och steg 18 (version 4.2x).

Konfigurer bara din arbetskopiera. Behåll en omodifierad kopia "för den händelse". Filerna CF och CG MASE finns på din arbetsdiskett för att du ska kunna konfigurerar Fwb. Dessa kan tas bort när du är klar.

- b. Ladda Fwb med din arbetsdiskett i Drive 1. Om du använder XB kommer Fwb att automatladdas när du väljer Extended Basic. Om du använder EA-modulen välj "5. Run program File" och ange filnamnet "DSK1.FW".
- c. Om du laddade från XB, välj Configure på XB menyn. Om du laddade från EA, välj User List och sedan Configure. Om Configure inte finns på någon av dessa menyer kan du ladda den genom LOADER funktionen (val 2). Filnamnet är "DSK1.CF".

(3) Topp Meny

- a. Tryck på någon tangent för att komma förbi startbildan.
- b. Tryck <S> för att modifiera systeminformationen (Sysinfo).

(4) Sysinfo Meny

- a. Tryck <I> för att ladda systemets konfigureringsfil. Ange filnamnet ("DSK1.SYSCON"). Du kan ha flera konfigureringsfiler med de namn du själv väljer. För att skapa en måste du först aktivera konfigureringsprogrammet, ladda konfigureringsfilen och sedan installera den (se steg 12).

- b. Tryck <E> för att editera systemkonfigureringen.

(5) Laddnings Meny

- a. Tryck <L> för att komma till laddningsmenyn.

- b. Boot Tracking bör vara ON om du laddar från en diskettdrive. RAM diskanvändare bör stänga av den eftersom Fwb inte kan spåra laddning från RAM diskar. Tryck på för att skifta mellan Boot Tracking ON och OFF.

- c. Siffran som står efter "TI Writer side" är det drivenummer där de filer som utgör TI Writer återfinns. Denna siffra används om Boot Tracking är OFF eller inte fungerar.

Detsamma gäller för "Edit/Asm side". Normalt är båda 1. För att ändra dem tryck på <T> eller <E> och ange sedan drivenummer.

- d. "Working Drive" är den drive på vilken du vill ha din datadisk. Om du har dubbla diskdrivar är det normalt 2. Tryck <W> för att ändra "Working Drive".

Version 4.1x: Ange drivenumret (t ex "2").

Version 4.2x: Ange det kompletta drivenamnet (t ex "DSK2.>").

- e. Om "UL Immediate" är ON, kommer User List att vara den första meny som visas när du laddar via EA. Om du inte vill ha det så sätter du detta värde OFF. Tryck på <U> för att växla mellan ON och OFF.

FUNNELWEB KONFIGURERINGS INSTRUKTIONER - UGOC UTGAVA 2.0 - SIDA 3

Om du sätter detta ON kan du ladda från EA och få den meny som du utformat själv att visas först.

- f. I version 4.2x, kan du slå Hard Disk Path ON och OFF genom att trycka på <H>. När detta är ON kan du välja bland ett antal vägar (paths) för Fwb filerna. Försätt att trycka på <H> tills du har det riktiga namnet på vägen. Om du inte har en håddisk ska detta värde vara OFF.

- g. När alla värden är riktiga tryck på <CTRL C> för att komma tillbaka till Edit Meny.

(6) Yttre Enheters Meny

- a. Tryck <D> för att komma till Devices Menu.

- b. Tryck <E> för Editor Printer. Om du har en parallell printer anger du "PIO". Om du har en seriell printer se till att alla värden är riktiga för din printer.

- c. Tryck <F> för Formatter Printer. Om du har en parallell printer anger du "PIO.LF". Om du har en seriell printer se till att alla värden är riktiga för din printer och att ".LF" finns med.

- d. Object, Work and Program filnamnen är defaults för olika Fwb-funktioner. Om du anger ett namn ska drivenumret vara med (t ex "DSK1.LADDFIL"). Om du lämnar fältet blankt kommer det drivenummer som du angav för "Working Drive" i laddningsmenyn att användas (om du t ex angav working drive som 2 blir det "DSK2.>"). Om du vill att defaultvärdet ska vara en annan drive än working drive kan du ange "DSKn". Varje filnamn du markerar i snabbkatalogen kommer att lägga sig över defaultvärdet.

- e. Tryck <O> för att uppdatera objektfilen. Detta är defaultnamnet på filen när det behövs en objekt (DF80) EA fil (t ex när du laddar en objektfil).

- f. Tryck <W> för att uppdatera Work Fil namnet. Om du vill lägga ett namn här blir detta default när du använder LoadFile i Editorn.

- g. Tryck <P> för att uppdatera Programfilen. Detta är det defaultvärde som används vid laddning av EA programfiler (TI pgm, GPL Pgm, E/A Pgm mm).

- h. När alla värden är korrekta tryck <CTRL C> för återgång till Edit meny.

(7) Färg Meny

- a. Tryck <C> för att komma till Färgmenyn.

- b. Markören kommer att befinna sig på det första färgvalet. Detta är de bildfärger som visas när Fwb laddas. De andra färgerna är de som du går runt mellan när du trycker <CTRL 3> i Editorn eller 0 (noll) på

FUNNELWEB KONFIGURERINGS INSTRUKTIONER - UGOC UTGAVA 2.0 - SIDA 4

de flesta bilderna där du kan välja genom att ange en siffra.

Du har flera valmöjligheter i detta läge:

<E>dit: Ändra aktuell färg
 <N>ext: Flytta till nästa färg på listan
 ack: Flytta till föregående färg på listan
 <X>chg: Byta mellan två uppsättningar av färger
 <R>edo: Återställa färgerna till det läge de var när du började
 <V>iew: Se på aktuell färgkombination

- c. När alla färger är korrekta tryck <CTRL C> för att komma tillbaka till Edit meny.

(8) Meny Meny

- a. Tryck <M> för att editera TI Writer och EA menyerna.
 b. Tryck <T> för att editera alternativerna på TI Writer Meny och <E> för att editera Editor Assembler meny. Du kan ändra alternativet 4 t o m 7 på båda menyerna.
 c. För varje rad på meny har du dessa alternativ: <E>dit, <N>ext, ack och <R>edo. Dessa är desamma som i färgerna (se steg 7b).
 d. Om du <E>diterar en Menyrad måste du ange följande information:

- i. NAME: Detta är det namn som visas på meny. Det kan vara vad du vill upp till 10 tecken.
 ii. FILE NAME: Detta anges omedelbart till höger om meny namnet och får bestå av högst 2 tecken. Detta är namnet på den fil som kommer att laddas när du trycker på motsvarande menyvalstangent. Du kan inte ange "DSK". Fwb får drivnumret från Boot Tracking eller från laddningsinformationen (se stegen 5b och 5c).
 iii. TYPE: Här talar du om för Fwb vilken filtyp det är. Flytta den skuggade linjen mellan alternativen genom att trycka på <N>ext eller ack. Du har följande alternativ:

TIW PGM: Emulerar alternativ 3 på TI Writer Meny. Filen måste vara i EA program format.

GPL PGM: Detta används för de flesta program som laddas av EA "Run Program File" alternativet.

E/A PGM: Detta är för EA "Run Program File" alternativet

SCRIPT: Detta ger dig möjlighet att skriva en textfil för att ladda en serie av objektfiler (DF 80). Se FWD/UTIL för detaljer.

LOW MEM: Laddar objekt (DF 80) filer till det låga minnet. Se FWD/UTIL för detaljer.

LD/RUN: Använd detta för de flesta "Load and Run" objektfiler

FUNNELWEB KONFIGURERINGS INSTRUKTIONER - UGOC UTGAVA 2.0 - SIDA 5

(DF 80).

- e. När du är nöjd med TI Writer och EA menyerna trycker du <CTRL C> för att komma tillbaka till Edit meny.

(9) XB List Meny

- a. Tryck <X> för att ändra den meny som visas efter att du laddat Fwb från XB. De tre första alternativen ("TI Writer", "Edit/Asm" och "XB Return") kan inte ändras. OBS: i version 4.2x har "TI Writer" ändrats till "Text Edit".

- b. Nu har du fyra alternativ:

<E>DIT: Använd detta alternativ för att editera aktuell ingång (dvs XB-listan i SYSCON filen).

<F>ETCH LIST: Använd detta alternativ för att hämta meny i LOAD programmet. Normalt är detta desamma som finns i SYSCON filen.

<M>AKE RESERVE: Om du <F>etch (hämtar) en lista tryck <M> för att göra den till en reservlista. Om du senare trycker på <R>edo kommer denna lista att återställas (bra om du gjort bort dig!).

<X>CHG BUFFERS: Återställer den tidigare listan som du lagt i reserv.

- c. Under den initiala konfigureringen bör du trycka <E>dit.

- d. <E>dit visar alternativet i XB meny. Du har de vanliga tangenterna aktiva för att flytta dig mellan alternativen: <Edit>, <N>ext, ack och <R>edo. Se steg 7b för definitionerna.

- e. Om du väljer att <E>ditera en ingång måste du ange följande information:

- i. NAME: Detta är det namn som visas på menybilden. Detta kan vara vad du vill upp till 10 tecken.

- ii. BOOT TRACKING ON/OFF: Behåll Boot Tracking ON om filen kommer att finnas i den drive från vilken du ladda in Fwb. Sätt den OFF om du vill ladda från en annan drive. Boot Tracking fungerar inte med RAMdiskar.

OBS: I Version 4.2x, visas detta som BOOT PATH YES/NO.

- iii. Nästa parameter beror på version:

Version 4.1x: Tryck <S> för att skifta SECONDARY mellan NO eller YES. Om driven där denna fil finns är en annan än laddningsdriven och om du svarar YES på denna fråga kommer Fwb att leta efter den användarfil som angetts i steg 5c.

Version 4.2x: Om Boot Path är YES, får du frågan SET SIDE. Genom att trycka <S> skiftar valet mellan T-W och E/A. Detta är den drive som angetts i laddnings meny (steg 5) och är det

FUNNELWEB KONFIGURERINGS INSTRUKTIONER - UGOC UTGAVA 2.0 - SIDA 6

ställe där Fwb letar efter filen om Boot Tracking misslyckas. Om Boot Path är NO påminns du om att ange hela filnamnet.

- iv. REMINDER NO/YES: Om detta är YES kommer Fwb att ge dig en blänkare om att sätta in disketten med filen på. Om det är NO kommer Fwb att försöka läsa filen direkt. Sätt detta till YES om disketten med filen INTE finns i den angivna driven när du trycker på siffran för valet.

- v. FILENAME: Namnet på den fil Fwb ska ladda.

Version 4.1xi: Knappa in hela drive och filnamnet (t ex "DSK1.MYFILE").

Version 4.2xi: Om Boot Path är NO, ange hela drive och filnamnet (t ex "DSK1.MYFILE"). Om Boot Path är YES ange endast filnamnet utan "DSK1" (t ex "MYFILE").

- vi. TYPE: Här talar du om för Fwb vilken typ av fil det är. Flytta mellan alternativen med <N>ext och ack. Du kan inte lämna detta fält blankt. Du har de alternativ som angetts i steg 8d iii samt:

XB PROM: Detta är standard XB Program.

XB RETN: Detta ger återgång till XB "ready" bilden. Det fungerar som NEW kommandot.

- g. När du är nöjd med XB menyn tryck <CTRL C> för återgång till Edit menyn.

(10) UL List Meny

- a. Tryck <U> för att editera användarlistan

- b. Det finns minst två användarlistor att editera. Den första är filen "UL" och är den du ser när du trycker på "8. User List".

Den andra är filen "DS" och innehåller Disk Utilities alternativen i TI Writer menyn ("3. Disk Util").

Du kan skapa användarlistor. Huvudlistan kan kalla på en annan användarlista genom att att du gör denna till ett alternativ i den första. Enligt Fwb:s konventioner bör efterföljande listor kallas UM, UN etc (eller DT, DU etc).

- c. Från huvud användarlistans meny kan du göra följande val:

<E>dit Entries
<F>etch List
<M>ake Reserve
<X>chg Buffers
<S>ave UL File

Du bör följa dessa instruktioner för VARJE användarlista. Som ett

minimum bör du modifiera UL och DS.

- d. Tryck <F> för att hämta användarlistan. Ange korrekt filnamn ("DSK1.UL", "DSK1.DS" etc).

- e. Tryck <M> för att lägga den inhämtade listan som reserv.

- f. Tryck <E> för att editera användarlistan. Editeringen går till på samma sätt som med XB menyn (steg 9e) utom att filtyperna "XB Program" och "XB Return" inte är tillgängliga.

OBS I version 4.20 och 4.21 visas dessa alternativ men du kan inte komma åt dem

- g. Tryck <CTRL C> för att gå tillbaka till User List Meny.

- h. Tryck <S> för att spara Användarlistan. Använd korrekt filnamn ("DSK1.UL", "DSK1.DS" etc). Du MASTE spara dina förändringar i detta läge.

- i. Du kommer sedan att få frågan om detta är <M>ain (huvud) UL eller <O>ther (annan) UL. Main User List är de som man kommer till direkt från Fwb menyn (normalt UL och DS). Other User Lists är de som man kommer till via Main User List.

- j. <L>oad, <E>dit och <S>ave alla andra användarlistor som du kommer att använda. Varje användarlista ska sparas under ett eget filnamn - förändringar i en påverkar inte de andra.

- k. När du är klar med editeringen av användarlistorna trycker du <CTRL C> för återgång till Edit meny.

(11) Sysinfo Meny - Hur man sparar SYSCON filen

- a. Tryck <CTRL C> för att komma tillbaka till Sysinfo meny

- b. Tryck <S> för att spara SYSCON filen. Ange filnamnet ("DSK1.SYSCON") eller vilket namn du väljer).

- c. Tryck <CTRL C> för att återgå till Topps Meny

(12) Installations Meny

- a. Tryck <I> för att kalla upp Installations meny.

- b. Tryck <L> för "LOAD XB/XBII". Filnamnet bör vara "DSK1.LOAD". Andra det om så är nödvändigt

- c. Tryck <ENTER> för att ladda källfilen.

- d. När så anges tryck <ENTER> för att spara utgående fil.

- e. Tryck <F> för "FW/UTIL1 others". Filnamnet bör vara "DSK1.FW". Andra om så behövs. Fwb laddar antingen "FW" eller "UTIL1". Många

program letar emellertid efter FW för att ladda Fwb.

- f. Tryck <ENTER> för att ladda källfilen.
- g. När så anges tryck <ENTER> för att spara utgående fil.
- h. Tryck <CTRL C> för återgång till Topp meny.

(13) Avslutande steg

- a. Tryck <Q> för att gå ur.
- b. Gå ur Fwb och ladda in det igen. De ändringar du gjort syns inte förrän du laddat in programmen igen. (se steg 14).
- c. Kontrollera alla menyalternativen för att se att de fungerar och att de ser ut på det sätt du vill ha dem. Om du går tillbaka till Configure för att ändra något behöver du bara ändra just den del du verkligen vill ändra. Efter att du <S>parat SYSCON filen måste du <I>nstallera alla ändringarna i LOAD och FW (se steg 12).
- d. Ladda in DM 1000. Tryck på <FCFN 3> på den första bilden. Se till att printernamnet är riktigt. Om du ändrat någon information svara <Y> för att spara till disketten.

- e. Du är nästan klar. Gör en backupkopia av din egen konfigurerade Fwb diskett. Om din arbetskopia blir kvaddad behöver du inte gå igenom alla dessa steg en gång till för att återskapa din konfiguration.

Du har nu tre disketter: den icke konfigurerade huvudkopian, arbetskopian och backup kopian av arbetsdisketten. Skrivskydda huvud- och backupkopiorna.

- f. Nu är du klar.

(14) Hur Funnelweb Konfigurering fungerar.

- a. SYSCON filen innehåller information om hur du vill att Fwb ska fungera. När Fwb laddas in läser det INTE någon information på SYSCON. Du använder istället Configure programmet för att läsa informationen i SYSCON och att installera funktionerna i LOAD och FW programmen.

- b. Under installationsprocessen ändras Configure i själva verket programmen på disketten. Det ändrar INTE programmet i minnet. Detta är skälet till varför du i steg 13b måste gå ur Fwb och ladda in det igen.

(15) Uppgradering av Funnelweb.

- a. SYSCON filerna är helt kompatibla inom 4.1x serien och inom 4.2x serien. 4.2x versionerna kan läsa 4.1x SYSCON filer men du måste

FUNNELWEB KONFIGURERINGS INSTRUKTIONER - UGOC UTGAVA 2.0 - SIDA 9

spara dem med version 4.2x Configure programmet.

- b. Instruktionerna nedan hjälper dig att gå från en äldre version av Fwb (som kommer att kallas GAMMAL) till en nyare version (kallas NY).
- c. Sätt skrivskydd på din diskett med din GAMLA version.
- d. Sätt in din NYA arbetskopia i drive 1 och din GAMLA arbetskopia i drive 2. Om du har ett system med en drive måste du byta disketter.
- e. Ladda in Fwb från drive 1 och gå in i Configure programmet.
- f. När du laddar SYSCON byt till drive 2 (t ex "DSK2.SYSCON").
- g. Gå igenom den normala konfigureringsprocessen på vanligt sätt.
- h. För användarlistor måste du:
 - <F>etch "DSK2.UL" (eller vilken lista du nu editerar.)
 - <W>ake Reserve
 - <F>etch "DSK1.UL"
 - <X>change buffers
 - <E>ditera användarlistan om nödvändigt.
 - <S>ave "DSK1.UL"
- i. Följ steg 15 för alla användarlistor.
- j. <S>para SYSCON. Se till att filens namn är "DSK1.SYSCON".
- k. Installera konfigureringen i LOAD och FW.

TI*MES

TI-99/4a USERS GROUP(UK) ger ut tidningen TI*MES fyra gånger per år med drygt 60 sidor per nummer. Den är mycket innehållsrik med många nyheter och även artiklar om BASIC. Du kan bli medlem för 15 pund som betalas till:

TI-99/4a Users Group(UK), c/o Peter Walker,
24 Bacons Drive, CUFFLEY, Herts,
EN6 4DU, Storbritannien

FUNNELWEB KONFIGURERINGS INSTRUKTIONER - UGOC UTGAVA 2.0 - SIDA 10

(16) FUNNELWEB MENY ÖVERSIKT (fortsättning)

MENU MENY : XB LIST MENY

```

MENU MENY      : XB LIST MENY
                :
TI Writer side : Edit entries
Edit/Asm side  : Fetch list
                : Make reserve
                : Xchg buffers
Edit Choices:

```

EDIT MENU

```
Name
File name
Type: TIW Pgm
      GPL Pgm
      E/A Pgm
      Script
      Low Mem
      Ld/Run
```

Edit Choices

Boot Tracking ON/OFF
Secondary NO/YES
Reminder NO/YES

4.2x Boot Path YES/NO
Set Side TI/EA
Reminder NO/YES

USER LIST MENY	:	Filemn
:	:	
Edit entries	:	Type: TIW Pgm
Fetch list	:	GPL Pgm
Make reserve	:	E/A Pgm
KXchg Buffers	:	Script
Save UL file	:	Low Mem
:	:	L/R A/M
SAVE MENY	:	XB Pgrm
:	:	XB Retn

```

USER LIST MENY
      Edit entries
      Fetch list
      Make reserve
      Check Buffers
      Save UL file
      SAVE MENY
      Main UL
      Other UL
      EDIT MENY
      Denna är densamma
      som XB Edit M
      utom att XB P
      och XB Retn i
      är tillgänglig

```

Denna är densamma
som XB Edit Menyn
utom att XB Pgrm
och XB Retn inte
är tillgängliga

(17) FUNNELWEB VERSION 4.1x FILER

File	Ändamål	**Anv på** SSSD DSSD Disk Disk	Anm
-READ-ME	Dokumentation		
AS/AT	Assembler		1
C99PFI:O	C99 Utility		2
CF/CG	Konfigurerings Pgm	D D	3
CHARA1	TI Wr Tecken	S D	
CHARA2	EA Tecken		1
CP	c99 Laddare		2
CT8RAM	Laddare för Modul		2
DP	Disk Patch	S D	
DS	Användarfil m Diskedit	S D	
EA	EA Laddare	S D	
ED/EE	TI Wr Editor	S D	
FMSAVE	Save Funktion	S D	1
FO/FP	TI Wr Formaterare	S D	
FWD0C/EASM	Dokumentation		
FWD0C/LOAD	Dokumentation		
FWD0C/REPT	Dokumentation		
FWD0C/TIWR	Dokumentation		
FWD0C/UTIL	Dokumentation		
FWSAVE	Save Funktion		1
LDFW	Alt Ladd Program		4
LH	Line Hunter		1
LL	Low Mem Laddare	S D	
LOAD	XB Load Program	S D	
MG/MH	DM 1000	S D	
QD	Snabb Katalog	S D	
SAVIT	Exempel på Scriptfil	S D	
SL	Script Laddare	S D	
SYSCON	System Konfiguration	S D	
UL	Användarlista	S D	
UTIL1	Laddar Fwb fr EA	S D	5
XB4THLD	Forth Laddare	S D	2

1. Dessa filer används normalt endast för Assembler programmering.
2. Används med angivna program. Se FWD0C/UTIL för detaljer. Ta dem med bara om det är nödvändigt.
3. CF och CG måste finnas på din diskett närhelst du vill ändra konfigurationen. SSSD användare kan ev vilja ta bort dessa filer efter konfigurationen av Fwb för att skapa utrymme för andra program.
4. LDFW kan användas för att ladda Fwb från BASIC med EA, XB II eller Mini Memory. Se FWD0C/UTIL för detaljer.
5. Denna fil kan benämnas UTIL1 eller FW. FW rekommenderas.

(18) FUNNELWEB VERSION 4.2x FILER

File	Ändamål	**Anv på** SSSD DSSD Disk Disk	Anm
-READ-ME	Dokumentation		
AS/AT	Assembler		1
C1	TI Wr Tecken	S D	
C2	EA Tecken		1
C99PFI:O	C99 Utility		2
CF/CG	Konfigurerings Pgm	D D	3
CP	c99 Laddare		2
CT8RAM	Laddare för RAM modul		2
DP	Disk Patch	S D	
DR	Katalog Översikt	S D	
DS	Användarfil m Diskedit	S D	
EA	EA Laddare	S D	
ED/EE	TI Wr Editor	S D	
FO/FP	TI Wr Formaterare	S D	
FSAVE	Save Funktion	S D	1
FW	Laddar Fwb fr EA	S D	4
FWD0C/EASM	Dokumentation		
FWD0C/DREV	Dokumentation		
FWD0C/LOAD	Dokumentation		
FWD0C/REPT	Dokumentation		
FWD0C/TIWR	Dokumentation		
FWD0C/UTIL	Dokumentation		
LDFW	Alt laddn program		5
LH	Line Hunter		1
LL	Low Mem Laddare	S D	
LOAD	XB Load Program	S D	
MG/MH	DM 1000	S D	
QD	Snabbkatalog	S D	
SCRIPT	Ex på Script fil	S D	
SL	Script Laddare	S D	
SYSCON	System Konfiguration	S D	
UL	Användarlista	S D	
XB4THLD	Forth Laddare	S D	2

1. Dessa filer används normalt endast för Assembler Programmering
2. Används med angivna program. Se FWD0C/UTIL för detaljer. Ta med endast om de behövs.
3. CF och CG måste finnas på disketten närhelst du ska ändra konfigurationen. SSSD kan ev vilja ta bort dessa filer efter konfigurationen av Fwb för att bereda plats på disketten för andra program
4. Denna fil kan heta UTIL1 eller FW. FW rekommenderas
5. LDFW kan användas för att ladda Fwb från BASIC i EA, XB II eller Mini Memory. Se FWD0C/UTIL för detaljer.