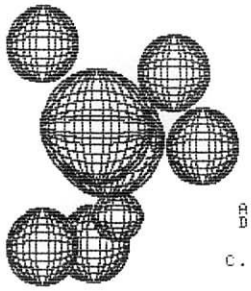
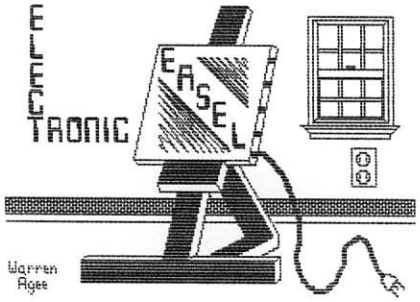
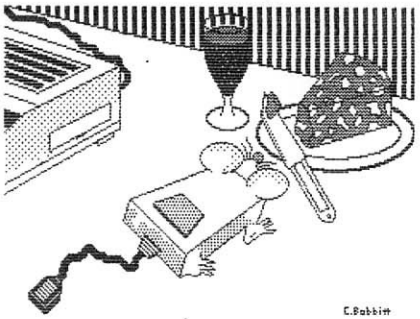
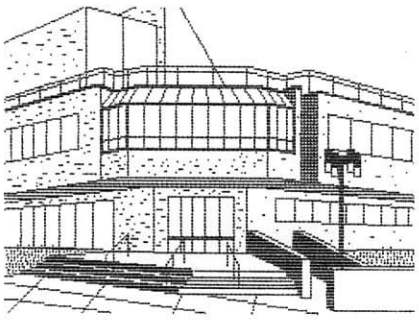
PROGRAM BITEN 93-3









ISSN 0281-1146

REDAKTÖREN

Niklas Johansson, Moråvägen 2, Hogstad, 595 93 MJÖLBY beställde 80-kolumnskortet OPA T.I.M. i juli 1992 och har ännu efter ett år (6 juli 1993) inte fått det. Han fick förra hösten ett brev från OPA där Garry Bowser förklarade att de hade problem. Samma brev finns publicerat i Micropendium Nov 92 p.07. Slutsatsen är att du inte bör beställa något från OPA innan Niklas fått T.I.M. levererad. Niklas undrar också om någon medlem har Imagewise Video Digitizer till Geneve.

Nya priser från Bud Mills Services för färdigbyggda Horizon-kort:

RAM-disk 256 kbytes	180 $
RAM-disk 512 kbytes	230 $
RAM-disk 1 Mbytes	355 $
RAM-disk 2 Mbytes	530 $
P-GRAM 72 kbytes	180 $
P-GRAM+ 192 kbytes	205 $
Tillägg för klocka	20 $

Porto 10 $ tillkommer för Europa. ■

MIDI MASTER 99

PB 91-1.32 beskriver MIDI Master 99 som nu säljs av Cecure Electronics. Fyra textfiler från Jim Peterson, Tigercub, kan fås om skiva och svarskuvert sänds till redaktören.

MIDI MASTER 99 and CASIO MT-240
Midi Master 99 consists of a cable, to connect your RS232 card to the keyboard, and a disk containing the necessary software, the documentation, and some sample music files. The documentation contains a good deal of technical material that is way over my head, but which is not necessary in order to use it. ...

MORE ABOUT MIDI MASTER 99

CONVERTING CALL SOUND TO MIDI SNF

CONVERTING MIDI MASTER 99 SNF FILES■

ANNONS — KÖPES

Thomas Åström, tel. 031-29 97 71, vill köpa diskkontroll för TI-99/4A. Allt som ger möjlighet till lagring på flexskiva är av intresse: Fristående eller Expansionsbox. ■

Redaktör: Jan Alexandersson
Medlemsregister: Claes Schibler
Programbankir: Börje Häll

Föreningens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9 NB
S-121 38 JOHANNESHOV, Sverige

Postgiro 19 83 00-6
Medlemsavgiften för 1993 är 120:-

Datainspektionens licens-nr 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. Föreningen förbehåller sig rätten att avböja annonser.

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 84-89 (per årgång)	50:-
90-92 (per årgång)	80:-
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). Se listor i PB89-3 och PB90-4.

Artiklar sändes till redaktören:
Jan Alexandersson
Springarvägen 5, 3tr
142 61 TRÅNGSUND
Tel. 08-771 0569
Ring eller skriv till mig om du har frågor om program/hårdvara

EXTENDED BASIC VDP-MINNE

av Jan Alexandersson

Av VDP-minnets 16384 bytes i XB används 1952 bytes för att hålla reda på grafiken för skärm, tecken och sprites. Det finns 6 olika tabeller för detta:
- SCREEN TABLE
- COLOR TABLE
- PATTERN TABLE
- SPRITE ATTRIBUTE TABLE
- SPRITE PATTERN TABLE
- SPRITE MOTION TABLE

Skärmtabellen (SCREEN TABLE) rymmer 24x32 tecken för att visa vad du ser på skärmen. XB har ställt in videoprocessorns VDP-reg #2 till >00 så att tabellen hamnar på >0000 - >02FF (0-767 decimalt). Det maximala utrymmet på 768 bytes används i sin helhet av XB. Kommandon som använder skärmtabellen är: PRINT, ACCEPT, CALL HCHAR, CALL VCHAR, CALL GCHAR, CALL CLEAR m.fl. Tänk på Basic-offset om du skriver och läser med assembler eller CALL POKEV.

Färgtabellen (COLOR TABLE) rymmer 32 bytes (reserverade för XB) för att bestämma förgrunds- och bakgrundsfärg på bokstäverna. En byte behövs per teckenset. XB använder dock endast teckenset 0-14 d.v.s. 15 stycken. Tabellen har utrymme även för teckenset 15-16 som inte kan användas för något annat. XB har ställt in VDP-reg #3 till >20 så att tabellen hamnar på >0800 - >081F (2048-2079 decimalt). Endast adress 2063-2079 används för teckenset 0-16. Även teckenset 15-16 ställs in vid start, Ready efter programkörning och CALL CHARSET. P.g.a. Basic-offset blir adress 2048-2062 oanvända. Färgtabellen används av kommandona CALL COLOR och CALL CHARSET.

Teckentabellen (PATTERN TABLE) används för att bestämma bokstävernas utseende med 8x8 punkter per bokstav. Det behövs 8 bytes per tecken. XB har ställt in VDP-reg #4 till >00 så att videoprocessorn ser tabellen på >0000 - >07FF (0-2047 decimalt). Som du ser så kommer teckentabellen att överlappa flera andra tabeller bl.a. skärmtabellen. Man har löst detta problem genom att endast använda 130 av de 256 möjliga tecknen. Dessutom har man förskjutit numreringen av tecknen med vad som brukar kallas Basic-offset på >60 (decimalt 96). XB-tecken 32 blir då VDP-tecken 128 medan XB-tecken 143 blir VDP-tecken 239 (XB 159 blir VDP 255). Resultatet blir att teckentabellen endast kan använda totalt 912 bytes på VDP-adresserna 1008-1919 för tecken. I princip skulle även tecken 144-159 kunna användas om inte SPRITE MOTION användes men detta kräver hjälp av assembler eftersom CALL CHAR ger felmeddelande. Tecken 158-159 är lediga även med 28 rörliga sprites. Kommandon som använder teckentabellen är CALL CHAR, CALL CHARPAT, CALL CHARSET.

SPRITE ATTRIBUTE TABLE används för att bestämma egenskaper hos en sprite: vertikal position, horisontell position, tecken, "early clock", färg. Det behövs 4 bytes per sprite. XB har ställt in VDP-reg #5 till >06 så att tabellen börjar på >0300 (decimalt 768). XB har endast plats för 28 sprites trots att VDP skulle ha klarat 32 sprites om tilldelade VDP-adresser hade varit en gnutta större. XB använder nu 112 bytes på VDP-adress 768-879. Kommandon som använder denna tabell är: CALL SPRITE, CALL LOCATE, CALL PATTERN, CALL POSITION, CALL DELSPRITE, CALL DISTANCE och CALL COLOR(#...).

SPRITE PATTERN TABLE har lagts ovanpå teckentabellen för bokstäver så att bokstavs-tecken och sprite-tecken med samma nummer blir lika trots att VDP skulle ha kunnat acceptera olika placeringar. XB har ställt in VDP-reg #6 till >00. Samma inskränkningar till tecken 30-143 finns som för teckentabellen och samma kommandon används, d.v.s. CALL CHAR m.fl.

SPRITE MOTION TABLE ligger alltid på VDP-adress >0780 - >07FF (1920-2047 decimalt) och kan inte flyttas men man kan ju låta bli att använda den.

Varje sprite behöver 4 bytes. Det finns 128 bytes tilldelat för XB så att 32 sprites får plats men eftersom SPRITE ATTRIBUTE endast rymmer 28 sprites så blir de sista 4 oanvändbara. Således 16 bytes bortkastade som skulle ha kunna använts av tecken 158-159 även om alla 28 sprites var i rörelse. Kommandon som använder denna tabell är CALL SPRITE och CALL MOTION.

REFERENSER

PB 84-5 Minnestabeller X-Basic
PB 89-4 Videoprocessor för 99/4A
PB 91-3 CALL LINK från Basic & XB
PB 91-6 Graphic1 för assembler

MG Night Mission, page 68-70:
CALL PEEK and CALL LOAD for XB
MG Explorer:
p.80 Scratch Pad RAM - XB Use
p.86 VDP RAM Extended Basic Use ■

EXTENDED BASIC SUPER SUB

av Jan Alexandersson

Du kan själv lägga till ett antal underprogram till din vanliga Extended Basic som fungerar ungefär som motsvarande i Triton Super-XB.

Följande tre kan användas även utan expansionsminne:

CALL ALL(numeriskt uttryck)
Fyller hela skärmen med ett valfritt ASCII-tecken.

```
9100 SUB ALL(X)
9110 CALL HCHAR(1,1,X,768)
9120 SUBEND
```

CALL COLORS(f,b)
Sätter för- och bakgrundsfärgen på samtliga teckenset.

```
9200 SUB COLORS(F,B)
9210 FOR I=0 TO 14 :: CALL COLOR
     (I,F,B):: NEXT I
9220 SUBEND
```

CALL KEYS("keylist",numerisk variabel)
Detta ger CALL KEY med validate. Den väntar tills någon tangent i keylistan har tryckts. Variabeln ger sedan positionen i keylistan. Ett HONK-ljud ges om fel tangent trycks.

```
9300 SUB KEYS(L$,TEST)
9310 CALL KEY(0,K,S):: IF S<1
     THEN 9310
9320 TEST=POS(L$,CHR$(K),1)
9330 IF TEST THEN SUBEXIT
9340 CALL SOUND(150,110,0)
9350 GOTO 9310
9360 SUBEND
```

Följande sex kräver att CALL LOAD fungerar vilket även betyder expansionsminne 32 kbytes eftersom CALL INIT måste göras före anropet:

CALL BYE
Som BYE men fungerar under körning.

```
9400 SUB BYE
9410 CALL LOAD(-31962,100,130)
9420 SUBEND
```

CALL NEW
Som NEW men fungerar under körning.

```
9500 SUB NEW
9510 CALL LOAD(-31962,100,124)
9520 SUBEND
```

CALL GOSPRT
Detta kommer att starta auto-rörelse av alla sprites samtidigt och är motsatsen till CALL STSPRT. AND av 191=255-64 används för att skriva in en nolla på den binära biten för auto-sprite enable.

```
9600 SUB GOSPRT
9610 CALL PEEK(-31806,X)
9620 CALL LOAD(-31806,(X AND 191))
9630 SUBEND
```

CALL STSPRT
Detta kommer att stoppa auto-rörelse av samtliga sprite och möjliggör skapande av extra stora rörliga figurer genom att flera sprites samverkar och startas med CALL GOSPRT. OR av 64 används för att skriva in en etta på den binära biten för auto-sprite disable. Du måste alltid avsluta programmet med CALL GOSPRT eftersom varken READY eller

NEW kommer att ordna enable av sprite-rörelse. Detta problem gäller även med Super-XB modulen.

```
9700 SUB STSPRT
9710 CALL PEEK(-31806,X)
9720 CALL LOAD(-31806,(X OR 64))
9730 SUBEND
```

CALL QUITON
Möjliggör QUIT med vanlig tryckning av FCTN = och är motsatsen till CALL QUITOFF. OBS att 255-16=239.

```
9800 SUB QUITON
9810 CALL PEEK(-31806,X)
9820 CALL LOAD(-31806,(X AND 239))
9830 SUBEND
```

CALL QUITOFF
Blockerar möjligheten till QUIT med FCTN = så att du inte gör QUIT av misstag.

```
9900 SUB QUITOFF
9910 CALL PEEK(-31806,X)
9920 CALL LOAD(-31806,(X OR 16))
9930 SUBEND
```

Följande fyra är egentligen assemblerprogram som laddas in genom poke med CALL LOAD och sedan CALL LINK till respektive program:

```
8000 SUB LINKLOAD
8010 CALL INIT
8020 ! Ladda assemblerprogr
8030 CALL LOAD(9500,37,14,37
,46,37,82)
8040 CALL LOAD(9506,23,108,0
,80,0,0,0,0,0,0,0,0,200,27,1
31,232,200,62,131,236,195,32
)
8050 CALL LOAD(9528,32,14,20
0,9,32,14,2,224,131,224,6,14
8,201,32,37,34,131,2,5,224,1
31,115)
8060 CALL LOAD(9550,4,96,0,9
6,193,32,22,108,6,148,2,224,
37,14,200,12,32,14,3,128,200
,11)
8070 CALL LOAD(9572,37,26,2,
224,36,250,4,193,216,1,131,1
24,4,32,37,28,0,52,16,10,200
,11)
8080 CALL LOAD(9594,37,26,2,
224,36,250,4,193,216,1,131,1
24,4,32,37,28,0,54,2,224,131
,224)
8090 CALL LOAD(9616,194,224,
37,26,4,91,2,224,36,250,2,0,
1,224,16,4,2,224,36,250,2,0)
8100 CALL LOAD(9638,1,160,4,
32,32,48,6,192,216,0,131,212
,4,193,216,1,131,124,2,224,1
31,224)
8110 CALL LOAD(9660,4,91)
8120 ! Uppdatera REF/DEF
8130 ! * Referenser
8140 ! * BEEP
8150 ! * HONK
8160 ! * SCRON
8170 ! * SCROFF
8180 CALL LOAD(16352,66,69,6
9,80,32,32,37,98,72,79,78,75
,32,32,37,120,83,67,82,79,78
,32,37,150,83,67,82,79)
8190 CALL LOAD(16376,83,67,8
2,79,70,70,37,160)
8200 ! Uppdatera pekaren
8210 CALL LOAD(8196,63,224)
8220 SUBEND
```

Observera att LOADASMBAS endast fungerar med max tre REF så jag fick tricksa något för att få med den fjärde REF.

Anropa CALL LINKLOAD innan följande fyra används. Själva underprogrammen ser ut så här:

CALL BEEP
Ger ett positivt ljud för inmatning.

```
10000 SUB BEEP
10010 CALL LINK("BEEP")
10020 SUBEND
```

CALL HONK
Ger ett negativt ljud för fel.

```
10030 SUB HONK
10040 CALL LINK("HONK")
10050 SUBEND
```

CALL SCRON
Aktiverar skärmen och är motsatsen till CALL SCROFF.

```
10060 SUB SCRON
10070 CALL LINK("SCRON")
10080 SUBEND
```

CALL SCROFF
Stänger av hela skärmen vilket kan användas när du vill ändra skärmen utan att något syns förrän den är färdig och aktiverar med CALL SCRON.

```
10090 SUB SCROFF
10100 CALL LINK("SCROFF")
10110 SUBEND
```

■

EXTENDED BASIC VERSIONER

av Jan Alexandersson

Det märkliga är att Extended Basic (XB) inte är "extended" i strikt mening eftersom konsolens Basic i GROM 1-2 inte används av Extended Basic. Motsvarande kod är således omskriven och utökat med nya funktioner och kan när det gäller CALL vara mycket snabbare för XB än för Basic. Eftersom koden är olika så finns det ibland skillnader som man inte väntar sig (se CALL CHAR och PRINT i mina XB-Tips i PB 93-1).

Extended Basic modulen har tillverkats i flera olika versioner, med förbättringar och tillägg. Ett alternativ till att köpa en ny version av XB, kan vara att använda ett bibliotek med SUB-program som anropas med CALL. Tigercub säljer sådana skivor (NUTS & BOLTS) med ett stort antal underprogram som är lagrade i MERGE-format.

Ett tredje sätt att utöka XB är med CALL LINK till assembler-rutiner som finns på flexskivor som laddas in till det låga 8 kbytes expansionsminnet med CALL LOAD("DSK2.FILNAMN"). Efter laddningen finns ett stort antal rutiner omedelbart tillgängliga med CALL LINK("RUTIN", variabel1,...). Exempel på sådana programskivor är:
- Enhanced Display Package
- STAR
- The Missing Link
- XHI och X80 (för 9938/9958)

En annan mycket intressant utökning är EZ-Keys som hjälper till vid editeringen i kommandomod. EZ-Keys är mycket bra men något svårt att beskriva för mig som aldrig använt den. EZ-Keys säljs av Asgard Software.

TI ORIGINALMODUL (PHM 3026)

Den äldsta modulen har CALL VERSION (X) med X=100. Denna modul började säljas 1:a kvartalet 1981 och har sannolikt aldrig sålts i Sverige. Den hade ett antal buggar som rättats i senare versioner. Det kan finnas program skrivna för version 100 som inte kan köras med senare versioner. Version 100 har alltid 28 aktiva sprites vilket gjorde att programskrivarna ofta ändrade detta med CALL LOAD(-31878,SPRITE) där SPRITE är det högsta sprite-nummer som används. Senare version av Extended Basic ställer in detta automatiskt så att detta CALL LOAD är helt onödigt, speciellt som det inte kan användas utan expansionsminne 32 kbytes.

Den vanliga XB-modulen med CALL VERSION(X) som svarar X=110 började säljas under 3:e kvartalet 1982. Mitt eget exemplar är märkt ATA 4482 (vecka 44). Modulen använder ROM >6000 - >7FFF med den övre 4 kbytes delen bankswitchad så att totalt 8+4=12 kbytes ROM användes. Dessutom används GROM-bankarna 3-6 med vardera 6 kbytes GROM. Totalt finns således 36 kbytes ROM+GROM.

Extended Basic levererades med en tjock manual på engelska på 224 sidor som beskriver varje kommando i alfabetisk ordning. Det tokiga är att alla exempel är skrivna för tangentbordet i TI-99 och inte för TI-99/4A. Du fick även en tunn 20 sidors översiktsinformation på sex språk inklusive svenska. Den svenska delen är dock endast tre sidor. Dessutom finns det en "TI-99/4 Extended Basic Reference Card" vikt som dragspel. Denna saknar tillägg för TI-99/4A och XB version 110. ASCII-koder upp till 126 borde vara med (använd kortet till Basic). Listan över kommandon saknar ! (tail remark), AND, OR, NOT, XOR, !@P- (pre-scan stopp) och !@P+ (pre-scan start).

Du fick även en 15 sidors "Important Product Information for TI XB". Den beskriver skillnaden mellan tangentbordet för TI-99 och TI-99/4A samt de nyheter som finns i version 110 jämfört med version 100. TI-99/4A har även små bokstäver med ASCII kod 96-126. CALL KEY har även tangentbord 3, 4 och 5 (se även beskrivningen i Basic-manualen). En annan nyhet i version 110 är pre-scan

start !@P+ och stopp !@P-. Allt detta samt rättelse av ett antal fel i den tjocka manualen finns i detta viktiga tillägg. Det kom även ett löst blad med "Addendum to TI XB" som beskriver ytterligare fel i manualen.

Det finns även XB-moduler tillverkade av Exceltec och Tenex/MICROpal. Dessa är troligen helt identiska med TI:s version 110 utan några tillägg.

MECHATRONIC EXTENDED BASIC PLUS

Denna modul som gavs ut 4:e kvartalet 1985 har utökats med nya kommandon för bl.a. minneshantering (BHCOPY, VPOKE, VPEEK, GPEEK, ALLSET, MOVE, MSAVE, MLOAD, FIND m.fl.) och grafik men är i övrigt helt kompatibel med TI:s originalversion. Den svarar CALL VERSION(X) med X=110. Anders Persson har beskivit innehållet i PB 86-5. Grafikrutinerna måste laddas med CALL APESOFT från kommandomod innan du laddar in programmet. Grafiken fungerar i Graphic1 mode genom att ett antal tecken placeras nära varandra och omdefinieras grafiskt. Det finns således risk för "out of ink" precis som i TI LOGO II. Någon äkta Graphic2 används således inte vilket gör grafikfältet begränsat i storlek. Alla dessa grafikkommandon för Apesoft använder syntaxen CALL LINK("NAMN",parameter1,parameter2, ...).

TRITON SUPER EXTENDED BASIC (BDAK)

GRAM Kracker levererades med ett fåtal nya CALL för XB: NEW, BYE, CLSALL, CLOCK, CLKOFF och CAT.

GK Utility I såldes som en fristående skiva med patchar till XB som dessutom ordnade LIST med valfri bredd (t.ex. 28), RES även av en mindre del av programmet, TRACE även till printer, markören som en smal understrykning och auto load bypass. Det finns nya möjlighet att låta markören hoppa till början eller slutet av en programrad samt flytta den vertikalt. Denna nya rörlighet för markören fungerar även under programkörningen med INPUT och ACCEPT. Andra nyheter är COPY, DEL och MOVE av delar av ett program i samband med editeringen. Nya CALL (utöver nyheterna för Gram Kracker): PEEKG, POKEG, PEEKV, POKEV, QUITON och QUITOFF.

Triton Super-XB modulen som gavs ut 2:a kvartalet 1987 svarar CALL VERSION (X) med X=120. Den innehåller samtliga funktioner som fanns i GK Utility I samt ytterligare tillägg. CALL LOAD som POKE fungerar även utan expansionsminne.

Ytterligare nya CALL: BEEP, HONK, STSPRT, GOSPRT, SCROFF, SCRON, COLORS, ALL, CHIMES, GOSUB, GOTO, KEYS, ALOCK, SHIFT, CTRL, FCTN, RESTORE och RUNPROG.

Super-XB levereras med tre manualer: Originalmanualen till TI XB utan ändringar (beskriver TI-99 & XB v.100), Tillägg för TI-99/4A & XB v.110 samt en 24 sidors manual över nyheterna i Super-XB v.120.

Super-XB har möjlighet till äkta Graphic2 (brukar kallas bitmap graphics något oegentligt) med hjälp av DRAW'N PLOT som finns inbyggt. Denna måste dock först laddas in i expansionsminnet i kommandomod med CALL FILES(2), NEW, CALL INIT och CALL DRAWNPLOT. Alla dessa grafikkommandon anropas med CALL LINK ("NAMN",param.1, param.2, ...).

Super-XB fungerar inte med Widget (omkopplare för 3 moduler) eftersom -5V inte används då detta är en GROM-emulator och inte äkta GROM.

MYARC EXTENDED BASIC Vn 2.12

Denna modul kräver Myarc RAM-disk för att kunna fungera. Det finns skärmeditor ungefär som i TI-Writer. Tre olika grafikmoder kan användas: Graphic1, Text1, Graphic2(bitmap). Den har ändrat användningen av VDP-RAM så att alla grafikmoder får plats med alla sina tabeller. Med Graphic1 kan man t.ex. använda alla 256 tecken utan begränsningar och upp till 32 sprites. TI:s originalmodul har kortat flera tabeller och låtit vissa överlappa varandra. CALL VERSION(X) ger X=200.

MYARC ADVANCED BASIC 3.0 FÖR GENEVE

CALL VERSION(X) ger X=300. Kan bl.a. beställas från Micropendium för 4 dollar. Kräver MDOS 1.23F eller 1.50H som kostar 4 dollar styck. Porto tillkommer. Kommandon listade i PB 88-1.

RICH EXTENDED BASIC

Finns nu i GRAM-version som bl.a. passar till P-GRAM och Gramulator. Eventuellt kommer en modul-version senare. Beskrivning av nya kommandon se Micropendium May 1993. Säljs av C.A.D.D. Electronics, 81 Prescott Road, RAYMOND, NH 03077, USA. Pris 25 dollar + porto.

ASGARD EXTENDED BASIC 3

Säljs på skiva för GRAM-karte av Asgard Software för 40 dollar+porto. Fungerar inte med övriga GRAM-kort (t.ex. P-GRAM) eftersom man använder 3 RAM-bankar på CPU-adress >6000. Finns även som Super Module med 256 kbytes för 100 dollar. Denna version är skriven av Winfried Winkler i Tyskland, se PB 91-4.28. Mycket av GPL är omskrivet till assembler så att den i många fall är 50 % snabbare en vanlig XB. CALL VERSION visar 150. Recension med lista över kommandon i Micropendium June 1993.

REFERENSER

NN 83-2 TI Extended Basic
PB 85-3 Kommandon i Basic och XB
PB 86-5 Mechatronic Extended Basic
PB 87-3 Super Extended Basic
PB 88-1 Star, CALL LINK för XB
PB 88-1 Enhanced Display Package
PB 88-1 Myarc Advanced Basic
PB 88-2 Triton Super Extended Basic
PB 89-2 XHI 3.4 - grafik för 9938
PB 89-2 Multimod för Super-XB
PB 89-3 Styrprogram för DM 99
PB 89-4 XHI 3.6, nyheter
PB 90-2 När datorn låser sig
PB 90-4 The Missing Link för XB
PB 91-5 Treffen in Berlin, XB3

PROGRAMBANKEN:
Basic Sort
Disk Manager 99
Enhanced Display Package
Star
XB Tools

Micropendium:
Feb 85 p.20 J&KH Super-XB,CALL LINK
May 85 p.31 Draw 'N Plot, review
Oct 85 p.44 Mechatronic XB II Plus
Dec 85 p.43 Oak Tree DEP, CALL LINK
Dec 86 p.34 GK Utility, review
Jan 87 p.37 Myarc Extended Basic II
Sep 87 p.32 Super X-Basic, review
Jan 88 p.31 EZ-keys, review
Mar 88 p.37 String Master,CALL LINK
Jul 88 p.37 Tips on using EZ-keys
Dec 88 p.40 Enhanced Display Pack
Feb 89 p.26 Super-XB and Widget
Jun 89 p.37 40-columns utilities
Sep 89 p.43 Super-XB and GK
Feb 90 p.35 XHI 3.6, review
Sep 90 p.29 The Missing Link,review
Apr 91 p.28 Rich GKXB available
Jan 92 p.34 Repair your XBcartridge
May 92 p.27 Extend. Basic Utilities
Sep 92 p.29 Rich RXB Extended Basic
Dec 92 p.21 Rich RXB and Gramulator
Apr 93 p.27 Extended Basic 3
May 93 p.25 RXB is trubo-charged XB
Jun 93 p.24 XB3 Sup. Module, review

Smart Programmer Dec 86: New XB CALL

TI*MES(UK):
No.11 p.54: Mechatronic XB II Plus
No.15 p.45: Myarc Extended Basic II
No.31 p.33: XB versions (Rambles) ■

THE TIGERCUB NUTS & BOLTS DISKS

What are they? The Nuts & Bolts disks are collections of 100 or more subprograms in merge format, ready to merge into your own programs.

And what does that mean? Well, TI-99/4A Extended Basic allows the use of user-written subprograms. And what are subprograms? You know them well. CALL COLOR, CALL SOUND, CALL HCHAR - those are all subprograms which are built into the Basic language. You can write your own subprograms, to do anything that Extended Basic is capable of, and tack them onto the end of your program to be CALLed whenever needed.

To put it in another way, using a subprogram is almost like running

one program from another - except that you can access it much faster, you can pass along any values that you want to, and you can return to where you left the first program.

Also, with a disk drive you can save programs in MERGE format and then MERGE them into a program in memory. Providing that the line numbers are different, the program which you merge in will be added to the existing program.

The variables used in a subprogram are entirely separate from those used in the main program, therefore libraries of utility subprograms can be developed in MERGE format, and merged into any program without conflict.

The Nuts and Bolts disks are libraries of such subprograms. The first disk contains 100 such subprograms, plus a tutorial on using them. Disk No. 2 contains 108, and Disk No. 3 contains 140 of them in 114 files. Nothing like them has ever been offered by anyone else for the TI-99/4A computer.

These 348 subprograms have been consecutively line-numbered with high line numbers so that they will not overwrite your program lines, and so that any number of them can be merged into a program without overwriting each other.

Advanced programming techniques have been used to make these routines as compact as possible, averaging hardly more than 3 sectors each, so that a hundred or more could be crammed onto a disk and so that they would add very litle to the length of a program. If you are learning to program, you might learn a great deal by studying these routines.

Each disk is accompanied by documentation on disk, explaining the use of each subprogram and giving a short demo routine which you can key in, run, and experiment with.

Many of these subprograms can be used by persons with almost no programming knowledge, to modify existing programs. For instance, a program written in Basic, which crashes with BAD VALUE when run in XBasic, will run with a simple CALL BXB, and CALL KILLQUIT will disable the infernal QUIT key. Many different screen character styles are available, as well as colorful wipes to replace CALL CLEAR.

However, it is the programmer who will find these disks truly invaluable. Even if you had the skill and ingenuity to develop these routines for yourself, wouldn't you rather pay just fifteen cents apiece for them?

The three Nuts & Bolts disks are available for $5 each, postpaid, from Tigercub Software, 156 Collingwood Ave., Columbus OH 43213, USA. ■

RELISTING PROGRAMS - 1

by Jim Peterson, Tigercub, USA

At the last meeting, our editor asked me about ways to convert listed programs to 28-column width, and to convert listed programs to runnable programs. A couple of days later, I had a phone call from a user asking about the same thing. And, I have received a few newsletters with reprints of an article describing a method of listing to the printer in 28-column format.

Why list in 28-column format? Because that is the way a program appears on the screen. It is much, much easier to key in a program accurately when it is published in 28-column format, because you can edit your work by checking the position of characters in relation to the line above - especially when the program contains long stretches of blanks, or long hex codes.

About that method currently being reprinted - it doesn't work. At least, it doesn't work properly with Extended Basic programs. The idea

is that you open the printer and send it ASCII codes 27 81 28, which sets the right margin at 28. You can get the same result by OPEN #1: "PIO",VARIABLE 28 .

The problem is that Extended Basic program lines can be keyed in up to 140 characters long, and can be forced considerably longer. When you LIST a program to disk, it is saved in DV/80 format. Any line longer than 80 characters is broken into separate 80 character records. When you break those records into 28-character segments, you have program lines stopping in the middle and then continuing on the next line. They can still be keyed in correctly, if you realize what has happened, but the listing will not be in screen format, which is the whole purpose of using 28 columns.

Besides, you probably don't want to output to the printer. You want to output to disk, so you can incorporate the listing into a text article, as I am about to do.

So, what to do? If you have the Triton Super Extended Basic module, it is as easy as pie. Just - LIST "DSK1.LISTING":28:1-32766 (You may omit the line numbers so just LIST "DSK1.LIST":28: Ed. note.). It will do a perfect job but the listing will be in DV/28 format, which will not load into Funnelweb. So I will now write a little program, save it, list it with my Super Extended Basic, and then load my little program to convert the DV/28 file into a DV/80 file which I will insert right here -

```
100 DISPLAY AT(10,1)ERASE AL
L:"Input file? DSK":"":"Outp
ut file? DSK" :: ACCEPT AT(1
0,16):IN$ :: ACCEPT AT(12,17
):OUT$
110 OPEN #1:"DSK"&IN$,VARIAB
LE 28,INPUT :: OPEN #2:"DSK"
&OUT$,OUTPUT
120 LINPUT #1:M$ :: PRINT #2
:M$ :: IF EOF(1)<>1 THEN 120
 ELSE CLOSE #1 :: CLOSE #2
```

But you don't have the Triton module? Well, several years ago I wrote a 28 column converter which will do the job perfectly. It will also optionally replace and transliterate those characters that get messed up when you print a program listing through the Formatter. It will even recognize unprintable blank characters which have been keyed in with the CTRL key and print their key letter underlined. That program was published in Tips From The Tigercub #18 with an upgrade in #21. It is available on my TI-PD disk #1015 and I will put it on the Spirit of 99 BBS again.

That program does require that the listing have standard line number spacing, numbered by tens from 100. If you are starting with a listing which is not in that format, this one will do the job but not as easily, because you have to first insert a carriage return at the end of each program line. To do that, load the listing into the Funnelweb Editor, press CTRL 0 to get the hollow cursor and CTRL U to get the underline cursor, go to the end of each program line with the arrow keys and press M.

```
100 DISPLAY AT(3,6)ERASE ALL
:"PROGRAM RELISTER":"":"  Wi
ll reformat a LISTed    XBas
ic program from any lineleng
th to any other length."
110 DISPLAY AT(8,1):"  Each
program line (not    file li
ne) must end in a    carriag
e return."
120 DISPLAY AT(12,1):"Input
filename?":"DSK" :: ACCEPT A
T(13,4):IF$ :: DISPLAY AT(15
,1):"Output filename?":"DSK"
 :: ACCEPT AT(16,4):OF$
130 DISPLAY AT(18,1):"Presen
t line length?" :: ACCEPT AT
(18,22)SIZE(2)VALIDATE(DIGIT
):A
140 DISPLAY AT(20,1):"Reform
at to what length?" :: ACCEP
T AT(20,26)SIZE(2)VALIDATE(D
IGIT):X :: IF X=A THEN 130
150 OPEN #1:"DSK"&IF$,INPUT
:: OPEN #2:"DSK"&OF$,OUTPUT
:: IF X<A THEN 230
160 IF EOF(1)THEN 270 :: LIN
PUT #1:M$ :: L=LEN(M$):: IF
POS(M$,CHR$(13),1)=0 THEN 18
0
170 IF P+L<X+1 THEN PRINT #2
:M$ :: P=0 :: GOTO 160 ELSE
PRINT #2:SEG$(M$,1,X-P)&CHR$
```

```
(13):SEG$(M$,X-P+1,255):: P=
0 :: GOTO 160
180 IF L<A THEN M$=M$&RPT$("
",A-L):: L=A
190 IF P=0 THEN PRINT #2:M$;
:: P=L :: GOTO 160
200 IF P+L<X THEN PRINT #2:M
$;:: P=P+L :: GOTO 160
210 IF P+L=X THEN PRINT #2:M
$&CHR$(13):: P=0 :: GOTO 160
220 PRINT #2:SEG$(M$,1,X-P)&
CHR$(13):SEG$(M$,X-P+1,255);
:: P=LEN(SEG$(M$,X-P+1,255))
:: GOTO 160
230 IF EOF(1)THEN 270 :: LIN
PUT #1:M$
240 L=LEN(M$):: IF L+P>X THE
N PRINT #2:SEG$(M$,1,X-P)&CH
R$(13):: M$=SEG$(M$,X-P+1,25
5):: P=0 :: GOTO 240
250 IF M$=CHR$(13)THEN 230
260 IF POS(M$,CHR$(13),1)<>0
 THEN PRINT #2:M$ :: P=0 ::
GOTO 230 ELSE PRINT #2:M$;::
 P=LEN(M$):: GOTO 230
270 CLOSE #1 :: CLOSE #2
```

That one is also on TI-PD 1015.

Now, about converting listings to programs, without having to key them in - well, let's save that for next month. ■

FORMATERA & TRANSLITTERERA

av Jan Alexandersson

FORMATERARE

Du behöver normalt en formaterare för att skriva ut texten från din ordbehandlare som kan vara TI-Writer eller FW-Editor 5.00. Det finns två grundversioner av formaterare: USA-Formatter och Euro-Formatter. Nästan alla kända formaterare fungerar i princip på samma sätt som USA-Formatter: TI-Wr original Formatter, FW-Formatter 4.40, RAG-Formatter, AP-Formaterare.

TI-Wr och FW 4.40 Formatter fungerar mycket lika men FW-versionen är avsevärt mera användarvänlig men jag tror att själva resultatet, utskriften, är identisk. De kan använda de vanliga formateringskommandona: .LM, .RM, .FI, .PL, .TL m.fl. Du måste dock se upp med följande tecken som inte fritt kan användas: @, &, *, ^ och punkt(.). Punkt först på en rad uppfattas som tillhörande ett formateringskommando (som inte skrivs ut) oberoende av vad som skrivs efter punkten, men det händer inget alls så länge formateraren inte hittar komandot i sin egen lista.

RAG-Formatter(finns i programbanken) är helt kompatibel med filer skrivna för TI-Wr eller FW. Det finns utöver detta ett antal kommandon som ökar möjligheterna, bl.a. kan @, &, * och ^ frigöras för generell användning. Programbiten skrivs ut med hjälp av denna formaterare.

AP-Formateraren av Anders Persson finns på TW/MP Upgrade skivan i programbanken. Den har en egen uppsättning formateringskommandon: .VM, .HM, .FY, .SL, .OD m.fl. som skiljer sig från TI-Wr original och FW. Dessa kommandon gör exakt samma sak som motsvarande kommandon som är listade ovan. I övrigt uppför den sig som USA-Formatter. Däremot är editorn på samma skiva en Euro-Editor. Du kan skriva accent ovanpå vokalen med FCTN- , . / och CTRL-/. Jag har tidigare aldrig själv använt formateraren eftersom mitt system vägrade att ladda filen. Det finns en patch i McGoverns dokumentation till Editor 5.00 som gör att du kan ladda formateraren utan TI-Wr modul. Använd en sektoreditor för att ändra FORMA1-filens första sektor byte >30 från >130A till >100A.

Euro-Formatter finns på en skiva från Charles Good där alla av McGovern rekommenderade patchar redan är gjorda så att den kan laddas från Funnelweb utan TI-Wr modul. Denna har samma formateringskommandon som TI-Wr och FW men samma problem finns med @, &, * och ^.

TAB-RADEN OCH ASCII-TECKEN >127

Editorn sparar alltid TAB-inställningarna på en rad sist i filen. USA-Editorn sparar TAB som börjar med tecken 128 (Ç) medan Euro-Editorn börjar TAB-raden med tecken 252 (η). USA-Editorns sätt används av TI-Editor, äldre FW-Editor och RAG-Editor. Se Micropendium May 89 p.13 som beskriver TAB-raden.

Formateraren ska inte skriva ut TAB-raden eftersom den endast är ett hjälpmedel för editorn. USA-Formatter betraktar alla rader som börjar med tecken 128-254 som en TAB-rad som inte ska skrivas ut. Euro-Formatter betraktar endast rader som börjar med tecken 252 som TAB-rad. Alla formaterarna klarar tecken 0-254 under förutsättning att de inte uppfattas som tillhörande en TAB-rad.

Du kan alltid använda USA-Formatter om tecken 128-254 eller 46(punkt) aldrig finns först i textraden. Du kan ordna detta genom att skriva ett betydelselöst tecken före alla dessa tecken. Jag använder CTRL-U SHIFT-T CTRL-U (ASCII-tecken 20) som då skrivs i första kolumnen i editorn men efterföljande tecken hamnar ändå i första kolumn på skrivaren. ASCII 20 betyder avsluta utdragen text men motsvarande görs ändå automatiskt av skrivaren vid slutet av varje rad. USA-Formatter klarar både USA-TAB och Euro-TAB så att de inte skrivs ut.

Du kan använda Euro-Formatter om tecken 252 eller 46(.) inte skrivs i första kolumnen på textraden. Samma lösning med ett betydelselöst tecken kan användas även här. Euro-Formatter kan inte hantera USA-TAB eftersom den skrivs ut som en vanlig textrad som börjar med tecken 128. Du måste först spara från P/E (Program Editor) eller med Print File från W/P (Word Processor) till flexskiva innan du använder Euro-Formatter. Naturligtvis klarar den Euro-TAB utan detta krångel.

TRANSLITTERERING

Du har möjlighet att beordra formateraren att byta ut ett tecken mot ett eller flera andra. Detta kallas translitterering. Om du skriver .TL 126:42 innan du använder tecken 126 (~) så kommer detta att skrivas ut som tecken 42 (*) på skrivaren. Translittereringen kan även användas för många tecken så att .TL 126:72,69,74 kommer att ordna utskrift av HEJ varje gång ~ står i texten.

Du kan även låta olika tecken byta plats men det visar sig att USA-Formatter och Euro-Formatter hanterar detta radikalt olika. USA-Formatter fungerar enkelt och rakt på sak. Följande rader kommer att byta A till B och B till A i utskriften:

```
.TL 65:66
.TL 66:65
```

Euro-Formatter kommer inte att förstå detta utan kräver att man gör utbytet via ett temporärt tecken som endast används internt i formateraren. Samma utbyte mellan A och B måste skrivas:

```
.TL 91:65
.TL 92:66
.TL 65:92
.TL 66:91
```

Detta kommer inte att förstås av USA-Formatter. Du måste välja motsvarande olika metoder om du vill permutera: A till B, B till C och C till A. Det finns således fall där translittereringarna måste skrivas olika beroende på vilken typ av formaterare som används.

Takttecknet ^ används av formateraren för att markera nödvändigt mellanslag som inte får fyllas ut av .FI (Fill). Det betyder att tak ^ eller tyskt Ü inte kan komma ut på skrivaren. Du kan tvinga fram dessa tecken med .TL 94:94 som kommer att ta bort betydelsen av fast mellanslag. Detta fungerar med båda typer av formaterare.

BACKSPACE ASCII-TECKEN 08

Formateraren kommer aldrig att sända backspace tecken 08 till skrivaren. Två backspace efter varandra kommer inte heller att backa utskriften med två positioner till vänster utan endast med en position oberoende av hur många tecken 08 du skriver in.

Formateraren gör en egen emulering

av backspace genom att sända två rader till skrivaren. Du kan se hur detta fungerar genom att låta formateraren skriva till en DSK-fil istället för PIO (RS232). Du kan även prova mot skrivaren med endast PIO (RS232) utan efterföljande .LF så ser du hur en rad blir två eller ännu flera rader. Den första raden kommer att ha alla tecken utom det som ska vara efter backspace och avslutas utan CR. Den andra raden kommer endast att innehålla det saknade tecknet efter backspace samt ett CR i slutet av raden.

Detta fungerar bra så länge du inte försöker använda kontrollkoder före backspace. Formateraren kommer nämligen att räkna kontrollkoderna när nästa bokstav efter backspace ska placeras i rätt kolumn. Den kommer då att skrivas ut för långt till höger. Kontrollkoder efter backspace fungerar ännu sämre eftersom backspace endast tar ett tecken åt gången. Även om du skriver backspace före varje tecken så hamnar varje sådant tecken på en ny utskriftsrad. ESC R (n) med backspace före varje tecken kommer att generera fyra rader till skrivaren med ESC på rad 2, R på rad 3 och (n) på rad 4. Kontrollkoden kommer inte i en sekvens till skrivaren och kommer således inte att fungera som avsett. Backspace fungerar bra så länge du bara använder vanliga tecken före och efter backspace. Det får inte finnas någon kontrollkod i hela raden före backspace. Du kan faktiskt skriva mer än två tecken ovanpå varandra om backspace finns före varje av de efterföljande tecknen.

Euro-Formatter har en unik möjlighet som saknas i USA-Formatter, nämligen backspace av ett temporärt tecken så att hela sekvensen av tecken hamnar på den andra utskriftsraden i rätt position för att hamna ovanpå avsett tecken. Detta temporära tecken kan även generera kontrollkoder.

Du har möjlighet att lura formateraren genom att sända tecken 128+8=136 till en ASCII 7-bits skrivare. Den mest signifikanta biten tas bort i skrivaren så att tecken 08 som är backspace verkligen kommer fram. Detta är nödvändigt för att välja Japan ESC R (8) på skrivaren p.g.a. problemet med backspace. Min skrivare uppfattar även ESC R (24) som Japan eftersom endast de fyra minst signifikanta bitarna läses av skrivaren.

TRANSLITTERERING AV TEXTFILER

Följande exempel visar hur en Euro-Writer fil kan skrivas ut på en ASCII 7-bits skrivare med hjälp av temporära tecken:

```
.CO Only for Euro-Formatter
.CO aaaaaa lower case with accents
.TL 148:97,08,208
.TL 149:189
.TL 150:190
.TL 151:97,08,211
.CO CHR 168-207 ARE ONLY INTERMEDIATE
.TL 189:27,82,02,123,27,82,00
.TL 190:27,82,01,064,27,82,00
.CO CIRCUMFLEX, UMLAUT, GRAVE, ACUTE
.TL 208:27,82,00,094,27,82,00
.TL 209:27,82,01,126,27,82,00
.TL 210:27,82,00,096,27,82,00
.TL 211:039
```

Den sista 00 i 189-210 ska bytas till 05 om du använder svenskt teckenset på skrivaren, ESC R (5). Exemplet visar att bokstäver som finns i något europeiskt 7-bits teckenset anropas (t.ex. ä och à) medan övriga ordnas med backspace och accent (t.ex. â och á). Av okänd anledning skriver min NEC Pinwriter P6 ut Italien 126 som ì istället för ì.

Här följer ett exempel på hur man kan lura formateraren att släppa igenom backspace som tecken 136 men detta fungerar inte med Euro-Formatter eftersom 136 där kommer att betraktas som ett temporärt tecken (från ACCENTS-3U):

```
.CO Only for USA-Formatter
.CO Only for 7-bit printer
.TL 128:67,136,44
.TL 135:27,82,01,092,27,82,00
.TL 021:27,82,02,064,27,82,00
.CO aaaaaa lower case with accents
.TL 131:97,27,82,00,136,94,27,82,00
.TL 132:27,82,02,123,27,82,00
.TL 133:27,82,01,064,27,82,00
.TL 160:097,136,039
.CO eeeeee lower case with accents
.TL 136:101,27,82,00,136,094,27,82,0
.TL 137:101,27,82,01,136,126,27,82,0
.TL 138:27,82,01,125,27,82,00
.TL 130:27,82,01,123,27,82,00
```

Jag har nu samlat ett antal translittereringsfiler på en skiva ALEX/FORMA (sänd skiva med frankerat svarskuvert till mig). En liknande fil från Charles Good (endast Euro-Writer till 7-bits skrivare) har använts som grund men har utökats och förbättrats. Jag har även gjort en lathund över tecken 128-254 med vilka tangenter som ska tryckas. Följande translittereringsfiler finns på min ALEX/FORMA skiva:

ACCENTS-1: Euro-Writer till 7-bits skrivare med Euro-Formatter.

ACCENTS-2: Euro-Writer till 8-bits skrivare med Euro-Formatter.

ACCENTS-3E: FW All Char Editor till 7-bits skrivare med Euro-Formatter.

ACCENTS-3U: FW All Char Editor till 7-bits skrivare med USA-Formatter.

ACCENTS-4: Svenska tecken <127 till 8-bits skrivare flyttade till >127.

ACCENTS-1 och -2 utgår från att du skrivit in accenterna på det mycket speciella sätt som endast finns i Euro-Writer. Om du verkligen äger en 8-bits skrivare med IBM-teckenset så bör du använda ACCENTS-2 istället för ACCENTS-1 eftersom utskriften blir mycket snyggare när flera direkta tecken finns tillgängliga.

ACCENTS-3 gör det möjligt för dig att använda den nya FW Editor 5.00 med All Char (finns nu även i 40-kolumnsversion) även om du bara har en 7-bitsskrivare. Du måste dock själv hålla reda på vilka tecken som kan translittereras och övriga måste du låta bli. På detta sätt kan du använda alla bokstäver med accent som finns i IBM-teckensetet, inklusive Ç ç Ñ ñ Æ æ som inte finns i Euro-Writer och valuta: ¢ £ ¥ R. Jag har även tagit med § som finns som 7-bits tecken 21 i IBM-setet. Tecken för grafik, grekiska och matematik kan inte användas. Av okänd anledning klarar min skrivare inte ESC R (4) för Danmark I och ESC R (9) för Norge men skriver ut ESC R (10) Danmark II som om det var Danmark I. Detta stämmer inte ens med min manual (bug!).

Jag använder själv ACCENTS-4 som flyttar de svenska bokstäverna till ASCII-koder >127. Jag har valt att bibehålla $, @, ^, och ~ som amerikanska bokstäver medan resten blir svenska.

ACCENTS-1, -2 och -3E fungerar inte om det finns andra kontrollkoder tidigare på en rad som ska skrivas ut med en vokal med accent. ACCENTS-3U fungerar däremot även tillsammans med andra kontrollkoder.

Du bör inte börja använda Euro-Writer om du inte tidigare använt den utan välj istället FW All Char.

MICROPENDIUM REFERENSER

Feb 85	Taking control of TI-Writer
May 88	Jack Sughrue PLUS!
May 88	Control characters and fonts
Jun 88	The first 32 ASCII codes
Nov 88	ESCape to six font sizes
Nov 88	Changing the CHARA1 file
May 89	TI-Writer TAB line
Jun 89	CHARA1FIX, part 1
Jul 89	CHARA1FIX, part 2
Aug 89	CHARA1FIX, part 3
Aug 89	Avoiding Formatter problems
Sep 89	Customizing Funnelwriter

Läs även artikeln om Editor i detta nummer där referenser till PB finns. ■

FW 40 & 80 KOL EDITOR 5.00

av Jan Alexandersson

Tony McGovern har nu (July/19/93) släppt ut sin nya Text Editor (ordbehandlare) version 5.00 både för 40-kolumn och 80-kolumn på skärmen. 40-kolumn kan naturligtvis skriva 80 kolumns text genom att stega tre fönster i sidled med FCTN-5. Första utgåvan av 40-kol kom i maj 1993. 80-kol som nu förbättrats ytterligare kom i sin första utgåva i dec 1992. 40-kol kan användas på en normal TI-99/4A med endast TI-origi-

nalkort i expansionsboxen och någon modul (t.ex. XB eller EA) som kan ladda Funnelweb 4.40. 80-kol måste dessutom ha videoprocessorn 9938 eller 9958. Den följande beskrivningen gäller både 40- och 80-kol om inget speciellt anges. Editor 5.00 måste laddas från någon av Funnelwebs menyer eftersom den använder inbyggda rutiner i FW 4.40.

Observera att 40-kol finns i två versioner ED/40 respektive ED/AEH där du normalt ska använda den senare. ED/AEH innehåller nu (July 93) Euro-Writer, fullt IBM-teckenset, hårddisk path dir och avsevärt förbättrad Show Directory (SD). Du måste döpa om filerna för den editor du ska använda till ED och EE. ED/AEH för 40-kol ändras till ED medan EE/AEH ändras till EE. Glöm inte att använda INSTALL/ED (och CON/ED) som patchar ED-filen innan du börjar använda editorn eftersom vissa saker endast kan konfigureras på detta sätt. Om du använder formaterare för att skriva ut DOC-filerna så måste du först göra RS /@/@@/. Formateraren kommer annars att skriva ut bokstäverna efter @ med fet stil utan något @.

Editorn kan fungera både som ordbehandlare (W/P) och program-editor (P/E) för assembler eller c-språket beroende på hur den laddas. Den ersätter således både TI-Writer och EA Editor. Denna nya version från juli 93 kan nu spara den sista raden i filen med TAB-inställningarna enligt USA TI-Writer eller Euro-Writer. Jag väljer USA-versionen som gör mina sparade filer möjliga att ladda med en äldre version av Funnelweb. Editor 5.00 kan dock alltid ladda båda typerna av TAB-inställningar.

TECKENSET PÅ SKÄRMEN

Editorn kan visa 255 olika tecken på skärmen som brukar finnas i nyare IBM-kompatibla skrivare. Följande ASCII-tecken 128-254 kan visas:

ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖÜ¢£¥Rƒ

áíóúñÑªº¿⌐¬½¼¡«»

αβΓΠΣσμτΦΘΩδ∞øε∩≡±≥≤⌠⌡÷≈°··√ⁿ²■

Min egen NEC Pinwriter P6 fungerar utmärkt. Du kan själv välja att ladda 7-bitars (ASCII 0-126) eller 8-bitars (ASCII 0-254) teckenset. Även om du har en gammal skrivare som inte klarar IBM-tecken så har editorn så många andra förbättringar som mycket väl gör det motiverat för dig att skaffa den nya versionen. Alla som använder ordbehandling med TI-99/4A bör skaffa denna nya version 5.00. Se artikel i PB 93-2 av Charles Good som mera detaljerat beskriver de flesta nyheterna.

Det finns både engelska och svenska ledtexter (ändrar även vad som skrives på kommandoraden) i menyerna samt ytterligare några europeiska språk. För varje språk finns även olika teckenset som ändrar vissa nationella tecken t.ex. ÄÖÅ på ASCII 91-93. Samma tecken finns dock även på ASCII >127 men kräver där ovanliga tryckningar av tangenter som inte är praktiskt med vanliga tecken.

Jag använder själv den engelska menytexten tillsammans med den svenska teckenfilen genom att döpa om filnamnen. Dessutom har jag ändrat den svenska teckenfilen så att paragraftecken § syns. Denna ASCII 21, som ingår i IBM-teckensetet för skrivaren, knappas in med CTRL-U SHIFT-U. Tänk bara på att detta avviker från TI-original om du behöver knappa in detta som kontrollkod till skrivare, men så höga tal som 21 används mycket sällan som attribut till andra kommandon. Jag har även kopierat följande från USA-teckenfilen:
$ ASCII 36, @ ASCII 64
^ ASCII 94, ~ ASCII 126

Detta kräver att jag själv ordnar en lämplig translitterering eftersom skärmen varken motsvarar skrivarinställning för USA eller Sverige.

Euro-Writer (E/W) ingår nu både i 40-kol och 80-kol versionen. Du väljer All-Char eller Euro-Writer vid laddning eller konfigurering. Euro-Writer ger dig möjlighet att skriva ett antal bokstäver med diverse accenter som sedan kan skrivas ut med en translitterering (.TL) fil

till en gammal 7-bitsskrivare. Varje bokstav ersätts då av ett flertal tecken till skrivaren, inklusive backspace som gör att accenterna kan skrivas ovanpå bokstaven. Detta är mycket krångligare än att använda All-Char. Nödvändig TL-fil finns inte på Editorskivan men väl på en speciell skiva från Charles Good med Euro-Formatter och translitterering. En mera komplett uppsättning translittereringsfiler finns på min egen skiva ALEX/FORMA.

PATH DIRECTORY

Hard Disk (HD) kan visa katalog för rot eller sub-directory på en hårddisk. Nytt i denna utgåva från juli 1993 är att man successivt kan gå djupare i underliggande sub-dir genom att markera i katalogen. Det går t.o.m. att backa så att närmast övergripande sub-dir visas. Denna typ av path directory fungerar även för Horizon DSK A-Z eller Corcomp RAM-disk.

FLEXSKIVOR MED PROGRAM

FW 40-kolumn Editor 5.00 ryms på en DS/SD (eller två SS/SD) och FW 80-kolumn Editor 5.00 ryms på en DS/SD (eller två SS/SD). Observera att det är två olika uppsättningar av skivor för 40- respektive 80-kolumn. Båda versionerna ryms på en DS/DD skiva varav 1187 sektorer används.

Det finns en SS/SD skiva från Charles Good med Euro-Formatter och translittereringsfiler. Skivan innehåller även en hel del beskrivande text. Dessutom finns min egen skiva ALEX/FORMA på SS/SD med ytterligare translittereringar som finns beskrivna i artikeln om formaterare i detta nummer.

Du måste även ha Funnelweb 4.40 från Oct/30/91 för att kunna använda Editorn. Dess 40-kolumns grundversion ryms på 3 st SS/SD och tilläggen för 80-kolumn ryms på ytterligare 2 st SS/SD. Detta kan sparas på större skivor DS/SD eller DS/DD om du så önskar.

Sänd skivor och frankerat svarskuvert till mig personligen så ordnar jag kopior.

REFERENSER

PB 85-1 TI-Writer/Multiplan upgrade
PB 89-1 Funnelweb 4.13
PB 90-1 TI-Writer once again
PB 90-3 RAG-Writer 4.5 Formaterare
PB 90-3 Konfigurering av FW 4.21
PB 90-5 Ordbehandling med 99/4A
PB 91-1 Funnelweb 4.31 -READ-ME
PB 91-2 HTA - TI-Writer
PB 91-6 Funnelweb 4.40
PB 93-1 FW 80-kol Editor 5.00
PB 93-2 FW Text Editor 5.00
PB 93-3 Formatera & Translitterera
PB 93-3 Talking to printer (BITS) ■

FROM BASIC TO ASSEMBLY - 4

by Bob August, Bug News, USA

Up to now we have been working in the Editor assembly environment. That means that you must load your program with the E/A module, Minimemory module or some other option 3 loader. You cannot load you program with Extended Basic. This month we are taking last months program and converting it to the Extended Basic environment so it can be loaded and called from Extended Basic with the CALL LINK statement and have it return to Extended Basic.

There are four changes that you must make. You must equate your Utilitites instead of using the reference. The TI Extended Basic equates are found in your E/A manual on page 415 through page 418. You have to add the Basic offset so your characters will appear on the screen and you need to return your program to the GPL workspace. Also you cannot assemble the program using the compress command. Use R or RL or RLT only.

We used two methods to write to the screen so you would have both routines. We put the first message on the screen without the BL (branch link). We put the second and third messages on the screen using the gosub BL. We removed the auto start feature so we can use the CALL LINK statement.

After you you assemble your program use the following Extended Basic program to load and call up your program. Use XBPRO/O as your object file name.

```
100 DISPLAY AT(12,3)ERASE AL
L:"LOADING ASSEMBLY PROGRAM"
110 CALL INIT :: CALL LOAD("
DSK1.XBPRO/O")
120 CALL LINK("START")
130 DISPLAY AT(12,2)ERASE AL
L:"YOU HAVE RETURNED TO YOUR
": :"  EXTENDED BASIC PROGRA
M"
140 DISPLAY AT(18,4):"PRESS
ANY KEY TO QUIT"
150 CALL KEY(0,K,S):: IF S=0
 THEN 150
```

When you run the above program it will automatically load and run your program as line 120 has the call link statement. After you return to Extended Basic you can re-enter your assembly program by typing CALL LINK ("START") or RUN 120. Why use the Extended Basic environment? Assembly runs faster than XBasic. You could use an assembly sort to sort your data and then go back to XBasic.

HAPPY ASSEMBLING!

```
****************************************
* BASIC TO ASSEMBLY  Lesson Number 4 *
****************************************
* Called from Extended Basic
* with a CALL LINK("START")
*
       DEF  START          Entry point label
*
KSCAN  EQU  >201C          Address for KSCAN utility
VSBW   EQU  >2020          Address for VSBW  utility
SOUND  EQU  >8400          Address for SOUND utility
GPLWS  EQU  >83E0          Beginning address of GPL workspace
*
WRKSP  BSS  32             Allocate E/A program workspace
SAV11  BSS  2              Save return address buffer
OFFSET BYTE >60            Basic offset for XB
* Messages 1 2 and 3
MSG1   TEXT 'THIS IS MESSAGE NUMBER ONE'
MSG2   TEXT 'THIS IS MESSAGE NUMBER TWO'
MSG3   TEXT 'PRESS: A For Message number one '
       TEXT '       B For Message number two '
       TEXT '                                '
       TEXT '       Enter key to quit'
*
       EVEN                Make sure start byte is even
*
START  MOV  R11,@SAV11     Save return address
       LWPI WRKSP          Load E/A workspace
* Print message one - DISPLAY AT(8,4)
M1     BL   @CLEAR         GOSUB CLEAR
       DATA 0              Data to pass to subroutine
       LI   R0,227         Screen location, start of MSG1
       LI   R3,MSG1        Message number one
       LI   R2,26          Length of message
M1LOOP MOVB *R3+,R1        Move message to R1 a byte a time
       AB   @OFFSET,R1     Add basic offset to byte in R1
       BLWP @VSBW          Write one byte to screen
```

```
       INC  R0            Move one byte to the right
       DEC  R2            Take one byte from length
       JNE  M1LOOP        If R2 is not zero goto M1LOOP
       B    @M3           GOTO M3
* Print message two - DISPLAY AT(14,4)
M2     BL   @CLEAR        GOSUB CLEAR
       DATA 0             Data to pass
       BL   @DISPLY       GOSUB DISPLY
       DATA MSG2,419,26   Message,Location,Length
* Print message three - DISPLAY AT(20,1)
M3     BL   @DISPLY       GOSUB DISPLY
       DATA MSG3,608,120  Message,Location,Length
       BL   @BEEP         Gosub BEEP
* Key scan routine - CALL KEY(0,K,S)
       LI   R0,>0300      Load R0 with 3 for CALL KEY(3,K,S)
       MOVB R0,@>8374     Move byte >03 to >8374
       CLR  @>837C        Clear status
       LI   R4,>2000                                (Ed.change)
KLOOP  BLWP @KSCAN        Scan keyboard
       CB   @>837C,R4     Check status for key press (Ed.change)
       JNE  KLOOP         IF S=0 THEN KLOOP         (Ed.change)
       MOV  @>8375,R3     Key value to R3
       CI   R3,>41        Is it 65 (ASCII A)
       JEQ  M1            If K=A then M1
       CI   R3,>42        is it 66 (ASCII B)
       JEQ  M2            If K=B then M2
       CI   R3,>0D        Is it 13 (Enter)
       JNE  KLOOP         If K<>13 then KLOOP
       LWPI GPLWS         Re-establish the GPL workspace
       MOV  @SAV11,R11    Restore return address
       CLR  @>837C        Clear the GPL status byte
       RT                 Return to Extended Basic
* Clear screen routine - CALL CLEAR
CLEAR  MOV  *R11+,R0      Move DATA 0 to R0
       LI   R1,>8000      >2000 plus >6000 OFFSET
CLOOP  BLWP @VSBW         Put space on screen
       INC  R0            R0=R0+1
       CI   R0,767        Is R0=767
       JLE  CLOOP         If less then 767 goto cloop
       RT                 Return
* Display on screen routine - DISPLAY AT
DISPLY MOV  *R11+,R2      Message to write to screen
       MOV  *R11+,R0      Screen address to start
       MOV  *R11+,R3      Length of message
DLOOP  MOVB *R2+,R1       Move one byte to R1
       AB   @OFFSET,R1    Add Basic offset
       BLWP @VSBW         Write byte to the screen
       INC  R0            Add one to R0 for new screen location
       DEC  R3            Take one byte from R3 (length)
       JNE  DLOOP         Jump if R3 not zero else
       RT                 Return to calling area
* Beep tone routine - BEEP
BEEP   LI   R10,>9100     Value for sound and turn on generator
       MOVB R10,@SOUND    Turn on beep tone sound
       LI   R0,>3000      Load delay of 12288
BLOOP  DEC  R0            Start count down
       JNE  BLOOP         If not equal to zero, loop
       LI   R10,>9F00     Value to turn off sound
       MOVB R10,@SOUND    Turn off sound generator
       RT                 Return to calling area
* End program without auto start so we can use CALL LINK
       END
```

■

SWEDLOW TI BITS * 28-29 *

by Jim Swedlow, USA

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

MAZE

This program will print mazes from very simple ones (level 0) to very complex ones (level 9). There are two bits of code worth discussing.

In line 240, you see this:
```
IF W AND 1 THEN
```

This tests to see if W is even or odd. The results are:
```
IF W AND 1 THEN <W IS ODD>
           ELSE <W IS EVEN>
```

You will also see similar code in lines 310 and 320. This is a neat trick that lets one variable serve a number of purposes. To figure it out, however, you will have to get the book that came with Miller's Graphics "Night Mission". Lots of good stuff in that book!!!

```
100 ! MAZE
110 ! BY STEVE KARASEK
120 ! MODIFIED BY J. SWEDLOW
130 ! JULY, 1989
140 !
150 DATA 19,4,9,8,6,12,4,16,
4,19,3,24,2,28,2,32,2,36,2,3
9
160 RANDOMIZE :: DIM M(40,40
):: INPUT "HOW MANY MAZES?":
Z :: PRINT
170 INPUT "LEVEL OF DIFFICUL
TY?":L :: IF L<0 OR L>9 THEN
 170 ELSE OPEN #1:"PIO"
180 DISPLAY AT(10,11)ERASE A
LL:"LEVEL ";L: :TAB(9);"Init
ializing"
190 FOR X=0 TO L :: READ S,N
 :: NEXT X
200 FOR X=1 TO N :: FOR Y=1
TO N :: M(X,Y)=0 :: NEXT Y :
: NEXT X
210 FOR X=1 TO N :: M(N+1,X)
,M(X,N+1),M(0,X),M(X,0)=16 :
: NEXT X
220 C,X,Y=1 :: DISPLAY AT(12
,9):"   ";"1 /";N^2
230 RANDOMIZE :: W=INT(RND*4
):: DX=X+(W=0)-(W=1):: DY=Y+
(W=2)-(W=3):: IF M(DX,DY)THE
N 230
240 M(X,Y)=M(X,Y)+2^W :: IF
W AND 1 THEN W=W-1 ELSE W=W+
1
250 X=DX :: Y=DY :: M(X,Y)=M
(X,Y)+2^W :: C=C+1 :: DISPLA
Y AT(12,9)SIZE(4):USING "###
#":C :: IF C=N*N THEN 280
260 IF M(X+1,Y)=0 OR M(X,Y+1
)=0 OR M(X,Y-1)=0 OR M(X-1,Y
)=0 THEN 230
270 RANDOMIZE :: X=INT(RND*N
)+1 :: Y=INT(RND*N)+1 :: IF
M(X,Y)THEN 260 ELSE 270
280 DISPLAY AT(12,9):"  Prin
ting" :: PRINT #1:CHR$(27);"
1":"Start";TAB(30);"Level: "
;L
290 PRINT #1 :: PRINT #1:"#"
;TAB(S+1);RPT$("#",S*(N-1)+1
):: S=S-1 :: S$=RPT$(" ",S):
: X$=RPT$("#",S)
300 M(N,N)=M(N,N)+8 :: FOR Y
=1 TO N :: FOR W=1 TO S :: P
RINT #1:"#";:: FOR X=1 TO N
:: PRINT #1:S$;
310 IF M(X,Y)AND 2 THEN PRIN
T #1:" ";ELSE PRINT #1:"#";
320 NEXT X :: PRINT #1 :: NE
XT W :: PRINT #1:"#";:: FOR
X=1 TO N :: IF M(X,Y)AND 8 T
HEN PRINT #1:S$;ELSE PRINT #
1:X$;
330 PRINT #1:"#";:: NEXT X :
: PRINT #1 :: NEXT Y :: S=S+
1 :: PRINT #1: :TAB(S*N-4);"
Finish":CHR$(12):: Z=Z-1 ::
IF Z>0 THEN 200
```

TALKING TO YOUR PRINTER

I had a problem recently making TI Writer print out some revisions to our club bylaws and I though you might be interested. Some background is needed first.

Some time ago I mentioned that there are almost no standards for software codes to control printer functions (font, appearance, line spacing, etc). There are, however, a few things that are uniform. This month's effort centers on two of them: carriage return and line feed.

For simplicity, I will refer to them as <CR> and <LF>. <CR> is ASCII code 13 and <LF> is 10 (or, in hex, 0D and 0A, respectively).

What do they do? Just what they say. A <CR> causes your print head to return to the beginning of the current line. A <LF> causes the print head to advance one line, in the same print position. Sending both <CR> <LF> causes the print head to move to the beginning of the next line.

This is almost universal. Many printers, however, have the ability to add a missing <CR> or <LF>. When in this mode, receipt of either command causes the print head to move to the beginning of the next line. Receiving both causes a double space. Usually this is controlled with DIP switches and is discussed in the printer's manual.

When you open a printer as "PIO" (I won't discuss RS232 names here because they act just the same), your RS232 card sends your printer a <CR> and a <LF> after every print line. You cannot control this as it is automatic.

You have some software control, however, by adding ".LF" or ".CR" to the printer name.

Here is a simple program that will show you what these extensions do:

```
100 DATA "PIO", "PIO.LF",
    "PIO.CR"
110 FOR I=1 TO 3
120 READ A$ :: OPEN #1:A$
130 PRINT #1:A$:A$
140 CLOSE #1
150 OPEN #1:"PIO"
160 PRINT #1: : :
180 CLOSE #1 :: NEXT I
```

When the printer name is PIO, your RS232 card will add a <CR> and <LF> to each line and your printer will print:

```
PIO
PIO
```

When the printer name is PIO.LF, you RS232 card will only add a <CR> so you will get:

```
PIO.LF
```

but it will be printed twice (bold). With PIO.CR, neither <CR> nor <LF> is sent so you should have:

```
PIO.CRPIO.CR
```

PRINTERS AND TI WRITER

Wait, you say, in TI Writer (or FUNNELWEB, etc), I can add <CR> to lines. Does this change things? The answer is yes and no.

The primary function of this <CR> is to tell TI Writer that you have reached the end of a paragraph. It uses it in the Editor when you cause a reformat and in the Formatter when you use FIll or FIll and ADjust.

What does it do to your printer? Well, lets say you have this text in your Editor:

```
UGOC<CR>
```

and then print it with PrintFile and PIO as your printer name. Your TI Writer would send this to your printer:

```
UGOC<CR><CR><LF>
```

In other words, it adds a <CR><LF> to whatever is on the line. The double <CR> is the same as a single <CR> as the print head can only move to the beginning of the print line once.

WHY DOES THE FORMATTER USE A ".LF"?

In short, the answer is to enable you to use bold (@) and underscore (&). Remember, when your printer name is PIO.LF, your RS232 card only adds a <CR> to each print line. Here is how it works.

If your entire line was:

```
@UGOC
```

The formatter would send the following to your printer:

```
UGOC<CR>
UGOC<CR>
```

UGOC<CR><LF>

This would cause triple printing of UGOC or bold. Underline is similar. If your entire line was:

&UGOC

The formatter would send the following to your printer:

UGOC<CR>
____<CR><LF>

On the page, UGOC would be underlined.

STRIKE OUT AND ITALICS

I am almost done with the background. As I said at the beginning, I was working on an update to our bylaws. The normal way to show changes is to strike out old text and put new text in italics. For italics, I used TransLiterate to change the open bracket "{" to start italics and the close bracket "}" to stop italics:

.TL 123:27,52
.TL 125:27,53

Whenever the Formatter runs into an open bracket "{" or ASC 123, it will send the ASC codes 27 and 52 to the printer, which switches to italics. Similarly, "}" or 125 causes the Formatter to send 27 and 53 which returns my printer to its normal font. It made editing easy as anything I wanted in italics I simply enclosed in brackets like {print this in italics}.

For strike out, I used transliterate to change the underscore to a dash:

.TL 95:45

Then, when I used the underscore command, the Formatter printed strike out. So &old &text would print as:

~~old text~~

NOW FOR THE PROBLEM (FINALLY)

When I mixed strike out and italics on the same line and the italics were first, I got garbage. Why? Because TI Writer and my printer count characters differently.

Suppose I had this line:

{new text} for &old &text

When it went through the formatter, the printout looked like this:

new text for old ~~text~~----

Why? Well, you see, TI Writer counted the open and close brackets as characters while the printer, recognizing them as control codes, didn't. So TI Writer thought it was sending this:

..new text.. for old text<CR>
--- ----<CR><LF>

The double dots ".." stand for the control codes that the brackets were translitterated into. TI Writer, as you can see, counts the number of characters sent to the printer, not the number in the Editor print line.

The problem was that the printer counts the print characters this way:

new text for old text
--- ----

The result was that the strike out was not correctly aligned.

The solution was to make sure that, in the same line, italics did not precede strike out. It could follow or be on a different line, but not precede.

So I wrote this:

&Old &Text becomes {new text}

and it printed like this:

~~Old Text~~ becomes *new text*

I left ADjust off (using only FIll) because this difference in character counts defeats the way the Formatter right justifies text.

Enjoy. ■

PROGRAMS WRITE PROGRAMS -2

by Jim Peterson, Tigercub, USA

Last month I promised you something more useful, so here it is. This routine will come in very handy for formatting screen text into neat 28-column lines, and will save the text in program lines of DATA statements. When you are ready to save, type @@@ and enter as the last line, then NEW and MERGE DSK1.LINEFILE -

```
100 !LINEWRITER to aid in fo
rmatting screen text into 28
-column format and saving it
 as DATA program lines in ME
RGE format - by Jim Peterson
110 !strings containing comm
as and quotation marks will
be ACCEPTed, and converted t
o DATA statements which RUN
correctly even though they
120 !are not enclosed in qu
otation marks!
130 CALL CLEAR :: OPEN #1:"D
SK1.LINEFILE",VARIABLE 163 :
: LN=30000
140 FOR R=1 TO 24 :: DISPLAY
 AT(R,1)SIZE(1):" " :: ACCEP
T AT(R,0)SIZE(-28):A$ :: IF
A$="@@@" THEN 180 :: B$=B$&C
HR$(200)&CHR$(LEN(A$))&A$
150 X=X+1 :: IF X/4=INT(X/4)
THEN 160 ELSE B$=B$&CHR$(179
):: GOTO 170
160 GOSUB 210 :: LN=LN+10
170 NEXT R :: X=0 :: CALL CL
EAR :: GOTO 140
180 IF B$="" THEN 200 :: IF
SEG$(B$,LEN(B$),1)=CHR$(179)
THEN B$=SEG$(B$,1,LEN(B$)-1)
190 GOSUB 210
200 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 :: END
210 PRINT #1:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&B$&CHR$(0):: B$=NU
L$ :: RETURN
```

Oh - that puzzle in last month's article? Try creating those DATA statements with this LINEWRITER program!

Now, let's get down to business and learn how to do all this. First, let's write a program that will write a program to list the token codes that you need to use to write a program that will write a program -

```
100 OPEN #1:"DSK1.TOKENLIST"
,DISPLAY ,VARIABLE 163,OUTPU
T :: FOR N=129 TO 254 :: L1=
INT(N/256):: L2=N-256*L1
110 PRINT #1:CHR$(L1)&CHR$(L
2)&CHR$(131)&CHR$(N)&CHR$(0)
:: NEXT N
120 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 :: END
```

Key that in, RUN it, then enter NEW, then MERGE DSK1.TOKENLIST. Now LIST it and you will see a list of ASCII codes 129 through 254 and their token meanings. Delete lines 171 through 175, 185, 198, 226 through 231, and 242. Change the definition of 199 to QUOTED STRING, of 200 to UNQUOTED STRING, and 201 to LINE NUMBER, and add line 255 !END OF FILE.

You don't need all those exclamation points, so change the program to a DIS/VAR 80 file by LIST "DSK1.TOKENLIST". Then key in this little routine.

```
100 OPEN #1:"DSK1.TOKENLIST"
,INPUT :: OPEN #2:"PIO" !or
whatever
110 PRINT #2:CHR$(27);"N";CH
R$(6)
120 LINPUT #1:A$ :: PRINT #2
:TAB(10);SEG$(A$,1,4)&SEG$(A
$,6,255):: IF EOF(1)<>1 THEN
 120 ELSE CLOSE #1 :: END
```

RUN it, and print out a list of all the token codes. Keep it handy, you'll be needing it. Notice that every Extended Basic statement has its own ASCII token code - even the ones you perhaps never heard of, such as LET and GO. Notice also that every keyboard symbol which affects program execution, such as + and =, has its own ASCII token code which is NOT the same as its keyboard ASCII code. And

notice that the double colon, used as a separator in Extended Basic multi-statement lines, has its own token.

Now, let's take a look at how a MERGE format program is put together. This routine will do that for you - and you will also find it very useful in debugging the MERGE programs you are going to write.

```
100 DISPLAY AT(3,5)ERASE ALL
:"D/V 163 FILE READER": :"
    by Jim Peterson": : :" T
o edit a file saved or":"cre
ated in MERGE format."
110 DISPLAY AT(12,1):"Output
 to? (S/P)S":" (S)creen":" (
P)rinter" :: ACCEPT AT(12,17
)SIZE(-1)VALIDATE("SP"):Q$
120 IF Q$="P" THEN DISPLAY A
T(14,1):"PRINTER? PIO" :: AC
CEPT AT(14,10)SIZE(-18):P$ :
: D=2 :: OPEN #2:P$
130 DATA ELSE,"::",!,IF,GO,G
OTO,GOSUB,RETURN,DEF,DIM,END
,FOR,LET,BREAK,UNBREAK,TRACE
140 DATA UNTRACE,INPUT,DATA,
RESTORE,RANDOMIZE,NEXT,READ,
STOP,DELETE,REM,ON,PRINT,CAL
L
150 DATA OPTION,OPEN,CLOSE,S
UB,DISPLAY,IMAGE,ACCEPT,ERRO
R,WARNING,SUBEXIT,SUBEND,RUN
,LINPUT
160 DATA ,,,,,THEN,TO,STEP,"
,",";",":",),(,&,,OR,AND,XOR
,NOT,=,<,>,+,-,*,/,^,
170 DATA QUOTED STRING,UNQUO
TED STRING,LINE NUMBER,EOF,A
BS,ATN,COS,EXP,INT,LOG,SGN,S
IN
180 DATA SQR,TAN,LEN,CHR$,RN
D,SEG$,POS,VAL,STR$,ASC,PI,R
EC,MAX,MIN,RPT$,,,,,,,NUMERI
C,DIGIT
190 DATA UALPHA,SIZE,ALL,USI
NG,BEEP,ERASE,AT,BASE,,VARIA
BLE,RELATIVE,INTERNAL,SEQUEN
TIAL,OUTPUT,UPDATE,APPEND
200 DATA FIXED,PERMANENT,TAB
,#,VALIDATE
210 DIM T$(126):: FOR J=1 TO
 126 :: READ T$(J):: NEXT J
:: E$(1)="LINE NOT CLOSED WI
TH CHR$(0)"
220 DISPLAY AT(16,1):"FILENA
ME? DSK" :: ACCEPT AT(16,14)
:F$
230 ON ERROR 240 :: OPEN #1:
"DSK"&F$,VARIABLE 163,INPUT
:: GOTO 250
240 DISPLAY AT(20,1):"I/O ER
ROR" :: ON ERROR STOP :: RET
URN 220
250 ON ERROR 260 :: LINPUT #
1:A$ :: X=ASC(SEG$(A$,1,1)):
: Y=ASC(SEG$(A$,2,1)):: IF X
=255 AND Y=255 THEN 410 ELSE
 270
260 PRINT #D:"FILE NOT CLOSE
D PROPERLY":"WITH CHR$(255),
CHR$(255) ?" :: STOP
270 PRINT #D:"LINE NUMBER":X
;"TIMES 256=";X*256:Y;"PLUS"
;Y;"=";X*256+Y
280 FOR J=3 TO LEN(A$)-1 ::
X=ASC(SEG$(A$,J,1))
290 IF X=201 THEN PRINT #D:X
;"LINE NUMBER" :: X=ASC(SEG$
(A$,J+1,1)):: Y=ASC(SEG$(A$,
J+2,1)):: J=J+2 :: PRINT #D:
X;"TIMES 256=";X*256:Y;"PLUS
";Y;"=";X*256+Y
300 IF X=199 THEN PRINT #D:X
;"QUOTED STRING" ELSE IF X=2
00 THEN PRINT #D:X;"UNQUOTED
 STRING" ELSE GOTO 360
310 J=J+1 :: X=ASC(SEG$(A$,J
,1)):: PRINT #D:X;"OF";X;"CH
ARACTERS"
320 ON ERROR 340 :: FOR L=1
TO X :: Y=ASC(SEG$(A$,J+L,1)
):: PRINT #D:Y;CHR$(Y):: IF
Y<32 OR Y>126 THEN PRINT #D:
"UNPRINTABLE CHAR - ERROR?"
330 NEXT L :: J=J+X :: GOTO
370
340 PRINT #D:"ERROR! INSUFFI
CIENT BYTES IN":"STRING" ::
IF ASC(SEG$(A$,LEN(A$),1))<>
0 THEN PRINT #D:E$(1)
350 ON ERROR STOP :: RETURN
250
360 IF X<129 THEN PRINT #D:X
;CHR$(X);" VARIABLE NAME" EL
SE PRINT #D:X;T$(X-128)
370 CALL KEY(0,K,S):: IF S=0
 THEN 390
380 CALL KEY(0,K2,S2):: IF S
2<1 THEN 380
390 NEXT J :: IF ASC(SEG$(A$
,J,1))=0 THEN PRINT #D:"0 EN
D OF LINE" ELSE PRINT #D:E$(
1)
400 GOTO 250
410 PRINT #D:X:X;"END OF FIL
E" :: CLOSE #1 :: STOP
```

Next month - how to do it! ■

IBM-teckenset: hjärter, ruter, klöver, spader med tecken 03-06 = ♥ ♦ ♣ ♠

TIPS FROM THE TIGERCUB #66

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213, USA

****

My three Nuts & Bolts disks, each containing 100 or more subprograms, have been reduced to $5.00 each. I am out of printed documentation so it will be supplied on disk.

My TI-PD library now has well over 500 disks of fairware (by author's permission only) and public domain, all arranged by category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #5 and the latest supplement is available for $1 which is deductible from the first order.

In Tips #65 I said that the TI could calculate to 14-digit accuracy, rather than the 8-digit accuracy of a PC. Actually the number in memory is calculated to 13- or 14-digit accuracy, depending on the number, but is rounded to 10 digits on the screen display, or shown in exponential notation if the number is extremely large or extremely small. If you want to see the complete number, this routine will show the normal screen display and the full number in memory. To see the complete range of numbers our little TI can handle, try inputting -
-9.9999999999999E127 and
-1.0000000000000E-128 and
1.0000000000000E-128 and
9.9999999999999E127.

```
100 OPEN #1:"DSK1.INT2",INTE
RNAL,RELATIVE,UPDATE
110 INPUT N
120 PRINT #1,REC 1:N
130 INPUT #1,REC 1:N$
140 X=ASC(SEG$(N$,1,1)):: Y=
ASC(SEG$(N$,2,1)):: IF Y>99
THEN Y=256-Y :: N$=SEG$(N$,1
,1)&CHR$(Y)&SEG$(N$,3,255)
150 FOR J=2 TO LEN(N$):: X$=
STR$(ASC(SEG$(N$,J,1)))
160 IF ASC(SEG$(N$,J,1))<10
THEN X$="0"&X$
170 P$=P$&X$ :: NEXT J
180 IF X<63 THEN Y$="."&RPT$
("00",63-X)&P$ :: GOTO 230
190 IF X>191 THEN Y$="."&RPT
$("00",X-192)&P$ :: GOTO 230
200 IF X>185 THEN Y$=SEG$(P$
,1,14-(X-185)*2)&"."&SEG$(P$
,14-(X-185)*2+1,255):: GOTO
230
210 Y$=SEG$(P$,1,(X-63)*2)&"
."&SEG$(P$,(X-63)*2+1,255)
220 IF ASC(Y$)=48 THEN Y$=SE
G$(Y$,2,255)
230 IF N<0 THEN Y$="-"&Y$
240 PRINT TAB(2);N :: PRINT
TAB(3);Y$ :: P$="" :: GOTO 1
10
```

But even the smart little TI has its limits. Try this- X=2/3-1/3-1/3 :: PRINT X. See the TI User's Reference Guide page III-13 for the explanation of all this.

Solving an equation such as X^X/X-X would be very difficult to solve by mathematical means, but our computer can find the answer quickly by systematic trial and error, to the 14-point limit of its accuracy.

```
100 DISPLAY AT(3,1)ERASE ALL
:"This program will solve ev
enthe most difficult equatio
nswith one variable."
110 DISPLAY AT(6,1):"Put you
r own equation in    line 21
0, using A for the   known v
alue and X for the   unknown
 value."
120 DISPLAY AT(24,6):"PRESS
ANY KEY" :: DISPLAY AT(24,6)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 120
130 CALL CLEAR
140 DISPLAY AT(8,1):"KNOWN V
ALUE? " :: ACCEPT AT(8,14):C
150 X=1 :: DISPLAY AT(12,1):
""
160 GOSUB 210
170 IF A<C THEN Y=X :: X=X*2
 :: GOSUB 210 :: GOTO 170 EL
SE 190
180 IF A>C THEN Y=X :: X=X/2
 :: GOSUB 210 :: GOTO 180
190 Z=(ABS(X-Y))/2 :: Y=X ::
 IF A<C THEN X=X+Z ELSE X=X-
Z
200 GOSUB 210 :: GOTO 190
210 A=X^X/X-X/2
220 IF A<C THEN DISPLAY AT(1
1,5):X ELSE IF A>C THEN DISP
LAY AT(13,5):X
230 IF A=C OR A=B THEN DISPL
AY AT(12,5)ERASE ALL:X :: GO
TO 140 ELSE B=A :: RETURN
```

In Recreational and Educational Computing, published 8 times a year at 909 Violet Terrace, Clarks Summit PA 18411, $27 per year, I found a neat routine to find the greatest common divisor and least common multiple of any two numbers - so I converted it to TI Basic and modified to do the same with multiple numbers.

```
100 CALL CLEAR :: PRINT "PRO
GRAM TO FIND THE GREATESTCOM
MON DIVISOR AND LEAST    COM
MON MULTIPLE OF ANY      NUM
BER OF NUMBERS.": :
110 DIM N(100)
120 PRINT "INPUT ZERO WHEN F
INISHED": :
130 T=T+1 :: INPUT "NUMBER "
&STR$(T)&"? ":N(T):: IF N(T)
=0 THEN 140 ELSE IF N(T)<>IN
T(N(T))OR N(T)<1 THEN T=T-1
:: GOTO 130 ELSE 130
140 AA=N(1):: GCD=N(2)
150 GOSUB 170 :: FOR J=3 TO
T-1 :: AA=N(J):: GCD=ABS(GCD
):: GOSUB 170 :: NEXT J
160 GOTO 210
170 R=AA-INT(AA/(GCD+ABS(GCD
=0)))*GCD
180 IF R<2 THEN GCD=R+GCD*(1
-R):: GCD=GCD*ABS(GCD>0)+ABS
(GCD=0):: GOTO 200
190 AA=GCD :: GCD=R :: GOTO
170
200 RETURN
210 PRINT "THE GREATEST COMM
ON DIVISOR OF YOUR";T-1;"NUM
BERS IS";GCD
220 L=N(1)*N(2)/GCD :: FOR J
=3 TO T-1
230 IF L/N(J)<>INT(L/N(J))TH
EN L=L*N(J)
240 NEXT J
```

```
250 LL=L/2 :: FOR J=1 TO T-1
 :: IF LL/N(J)<>INT(LL/N(J))
THEN J=T-1 :: GOTO 270
260 NEXT J :: L=LL :: GOTO 2
50
270 PRINT "AND THE LOWEST CO
MMON       MULTIPLE IS";L
```

Joy Warner called from the L.A. group, and mentioned that it would be nice to have a program to print out a page of math problems, and a page of answers. So here is one that will randomly create any number of either addition or subtraction problems, within any specified range of numbers, and output the desired number of copies to a printer in two columns of expanded print, numbered in sequence, plus a numbered answer sheet to make it easy for the teacher.

```
100 DISPLAY AT(1,4)ERASE ALL
:"MATH PROBLEM PRINTER" !by
Jim Peterson
110 DIM A(200),H(200),L(200)
:: OPEN #1:"PIO" :: PRINT #1
:CHR$(27)&"@"&CHR$(27)&"W"&C
HR$(1);
120 M$(1)="ADDITION" :: M$(2
)="SUBTRACTION" :: D$(1)="+
" :: D$(2)="- " :: ON$=CHR$(
27)&"-"&CHR$(1):: OFF$=CHR$(
27)&"-"&CHR$(0)
130 DISPLAY AT(3,1):"Do you
want?":"":"1. "&M$(1):"2. "&
M$(2):: ACCEPT AT(3,14)VALID
ATE("12")SIZE(1)BEEP:C
140 DISPLAY AT(8,1):"Range o
f numbers?":"From":"To" :: A
CCEPT AT(9,6)VALIDATE(DIGIT)
BEEP:LN :: ACCEPT AT(10,4)VA
LIDATE(DIGIT)BEEP:HN :: IF L
N>=HN THEN 140 ELSE HN=HN-LN
150 DISPLAY AT(13,1):"How ma
ny problems?" :: ACCEPT AT(1
3,20)VALIDATE(DIGIT)BEEP:P
160 DISPLAY AT(15,1):"How ma
ny copies?" :: ACCEPT AT(15,
18)VALIDATE(DIGIT)BEEP:CC
170 FOR J=1 TO P :: GOSUB 29
0 :: H(J)=N1 :: L(J)=N2
180 IF C=1 THEN A(J)=H(J)+L(
J)ELSE A(J)=H(J)-L(J)
190 NEXT J
200 FOR J=1 TO CC :: GOSUB 3
10 :: FOR K=1 TO P STEP 2
210 T1$=STR$(K)&".    "&STR$
(H(K)):: T2$=STR$(K+1)&".
 "&STR$(H(K+1))
220 PRINT #1:TAB(15-LEN(T1$)
);T1$;TAB(35-LEN(T2$));T2$
230 T1$=D$(C)&STR$(L(K)):: T
2$=D$(C)&STR$(L(K+1))
240 PRINT #1:TAB(15-LEN(T1$)
);ON$&T1$&OFF$&RPT$(" ",20-L
EN(T2$))&ON$&T2$&OFF$
250 PRINT #1:"":"":"" :: IF
K/19=INT(K/19)THEN PRINT #1:
CHR$(12);
260 NEXT K :: PRINT #1:CHR$(
12);:: NEXT J
270 PRINT #1:TAB(16);"ANSWER
S":"":""
280 FOR J=1 TO P STEP 2 :: P
RINT #1:TAB(6);STR$(J)&".  "
;A(J);TAB(26);STR$(J+1)&".
";A(J+1):: NEXT J :: STOP
290 RANDOMIZE :: N1=INT(RND*
HN+LN):: N2=INT(RND*HN+LN)::
 IF N1=N2 THEN 290
300 IF C=2 AND N2>N1 THEN T=
N2 :: N2=N1 :: N1=T :: RETUR
N ELSE RETURN
310 PRINT #1:"       "&M$(C)
&" PROBLEM PRINTER":"":"":""
:"" :: RETURN
```

And this one will do the same with multiplication problems.

```
100 DISPLAY AT(1,4)ERASE ALL
:"MULTIPLICATION PROBLEMS":"
         PRINTER" !by Jim P
eterson
110 DIM A(200),H(200),L(200)
:: OPEN #1:"PIO" :: PRINT #1
:CHR$(27)&"@"&CHR$(27)&"W"&C
HR$(1);
120 ON$=CHR$(27)&"-"&CHR$(1)
:: OFF$=CHR$(27)&"-"&CHR$(0)
130 DISPLAY AT(8,1):"Range o
f multiplicand?":"From":"To"
 :: ACCEPT AT(9,6)VALIDATE(D
IGIT)BEEP:L1 :: ACCEPT AT(10
,4)VALIDATE(DIGIT)BEEP:H1 ::
 IF L1>=H1 THEN 130 ELSE H1=
H1-L1
140 DISPLAY AT(12,1):"Range
of multiplier?":"From":"To"
:: ACCEPT AT(13,6)VALIDATE(D
IGIT)BEEP:L2 :: ACCEPT AT(14
,4)VALIDATE(DIGIT)BEEP:H2
150 IF L2>=H2 THEN 140 ELSE
R=LEN(STR$(H2)):: H2=H2-L2
160 DISPLAY AT(16,1):"How ma
ny problems?" :: ACCEPT AT(1
6,20)VALIDATE(DIGIT)BEEP:P
170 DISPLAY AT(18,1):"How ma
ny copies?" :: ACCEPT AT(18,
18)VALIDATE(DIGIT)BEEP:CC
180 FOR J=1 TO P :: GOSUB 31
0 :: H(J)=N1 :: L(J)=N2
190 A(J)=H(J)*L(J)
200 NEXT J
210 FOR J=1 TO CC :: GOSUB 3
20 :: FOR K=1 TO P STEP 2
220 T1$=STR$(K)&".    "&STR$
(H(K)):: T2$=STR$(K+1)&".
 "&STR$(H(K+1))
230 PRINT #1:TAB(15-LEN(T1$)
);T1$;TAB(35-LEN(T2$));T2$
240 T1$="X "&STR$(L(K)):: T2
$="X "&STR$(L(K+1))
250 PRINT #1:TAB(15-LEN(T1$)
);ON$&T1$&OFF$&RPT$(" ",20-L
EN(T2$))&ON$&T2$&OFF$
260 FOR S=1 TO R+3 :: PRINT
#1:"" :: NEXT S
270 LC=LC+5+R :: RC=LC+5+R :
: IF RC>=60 AND K<P THEN PRI
NT #1:CHR$(12):: LC=5
280 NEXT K :: PRINT #1:CHR$(
12);:: NEXT J
290 PRINT #1:TAB(16);"ANSWER
S":"":""
300 FOR J=1 TO P STEP 2 :: P
RINT #1:TAB(3);STR$(J)&".  "
;A(J);TAB(23);STR$(J+1)&".
";A(J+1):: NEXT J :: PRINT #
1:CHR$(12):: STOP
310 RANDOMIZE :: N1=INT(RND*
H1+L1):: N2=INT(RND*H2+L2)::
 RETURN
320 PRINT #1:"       MULTIP
LICATION PROBLEMS":"":"":"":
"" :: LC=5 :: RETURN
```

And division -

```
100 DISPLAY AT(1,6)ERASE ALL
:"DIVISION PROBLEMS":"
    PRINTER" !by Jim Peterso
n
110 DIM A(200,2),H(200),L(20
0):: OPEN #1:"PIO" :: PRINT
#1:CHR$(27)&"@"&CHR$(27)&"W"
&CHR$(1);
120 DISPLAY AT(8,1):"Range o
f dividend?":"From":"To" ::
ACCEPT AT(9,6)VALIDATE(DIGIT
)BEEP:L1 :: ACCEPT AT(10,4)V
ALIDATE(DIGIT)BEEP:H1 :: IF
L1>=H1 THEN 120
130 DISPLAY AT(12,1):"Range
of divisor?":"From":"To" ::
ACCEPT AT(13,6)VALIDATE(DIGI
T)BEEP:L2 :: ACCEPT AT(14,4)
VALIDATE(DIGIT)BEEP:H2
140 IF L2>=H2 THEN 130 ELSE
R=LEN(STR$(INT(H1/H2)))*2 ::
```

```
 H2=H2-L2 :: H1=H1-L1
150 DISPLAY AT(16,1):"How ma
ny problems?" :: ACCEPT AT(1
6,20)VALIDATE(DIGIT)BEEP:P
160 DISPLAY AT(18,1):"How ma
ny copies?" :: ACCEPT AT(18,
18)VALIDATE(DIGIT)BEEP:CC
170 FOR J=1 TO P :: GOSUB 31
0 :: H(J)=N1 :: L(J)=N2
180 A(J,1)=INT(H(J)/L(J))::
A(J,2)=H(J)-A(J,1)*L(J)
190 NEXT J
200 FOR J=1 TO CC :: GOSUB 3
20 :: FOR K=1 TO P STEP 2
210 LC=LC+1 :: T1$=STR$(K)&"
.    "&RPT$(" ",LEN(STR$(L(K
))))&RPT$("_",LEN(STR$(H(K))
))
220 T2$=STR$(K+1)&".    "&RP
T$(" ",LEN(STR$(L(K+1))))&RP
T$("_",LEN(STR$(H(K+1))))
230 PRINT #1:TAB(15-LEN(T1$)
);T1$;TAB(35-LEN(T2$));T2$
240 T1$=STR$(L(K))&"¦"&STR$(
H(K)):: T2$=STR$(L(K+1))&"¦"
&STR$(H(K+1))
250 LC=LC+1 :: PRINT #1:TAB(
15-LEN(T1$));T1$;TAB(35-LEN(
T2$));T2$
260 FOR S=1 TO R+3 :: LC=LC+
1 :: PRINT #1:"" :: NEXT S
270 IF 66-LC<5+R AND K<P THE
N PRINT #1:CHR$(12):: LC=5
280 NEXT K :: PRINT #1:CHR$(
12);:: NEXT J
290 PRINT #1:TAB(16);"ANSWER
S":"":""
300 FOR J=1 TO P :: PRINT #1
:TAB(3);STR$(J)&".  ";A(J,1)
;"REMAINDER ";A(J,2):: NEXT
J :: PRINT #1:CHR$(12):: STO
P
310 RANDOMIZE :: N1=INT(RND*
H1+L1):: N2=INT(RND*H2+L2)::
 RETURN
320 PRINT #1:"          DIV
ISION PROBLEMS":"":"":"":""
:: LC=5 :: RETURN
```

Bud Wright wrote this one for Irwin Hott, so he could listen to lower case text with the Speech Synthesizer. Imbed it with ALSAVE, access it with CALL LINK("CAPS",A$) and it will instantly convert any lower case letters to upper case. I found it invaluable in writing keyword search programs.

```
* CAPS/S BY BUD WRIGHT
* VERSION 1.1 10/17/86
STRREF EQU >2014
STRASG EQU >2010
MREG   BSS 32
STRBUF BYTE 255
       BSS 255
       DEF CAPS
CAPS   LWPI MREG
       CLR R0
       LI R1,1
       SETO @STRBUF
       LI R2,STRBUF
       BLWP @STRREF
       MOVB @STRBUF,R2
       SRL R2,8
       JEQ CAPOUT
       LI R1,STRBUF+1
CAPS2  MOVB *R1,R3
       SRL R3,8
       CI R3,96
       JGT CAPS1
CAPS3  SWPB R3
       MOVB R3,*R1+
       DEC R2
       JNE CAPS2
       CLR R0
       LI R1,1
       LI R2,STRBUF
       BLWP @STRASG
CAPOUT LWPI >83E0
       B @>006A
CAPS1  CI R3,122
       JGT CAPS3
       AI R3,-32
       JMP CAPS3
       END
```

Memory full, Jim Peterson ■

TITLE SCREEN

by Bob August, Bug News, USA

You can spice up those programs with fancy title screens. These are easy to do and they make your program more interesting. In Extended basic you could use sprites roaming around on the screen or simply put a fancy border around the screen. Remember that graphics uses two columns on the left and two columns on the right that are not used by Basic or Extended Basic. You can use these and not lose any of your usable screen.

We have encluded two possible title screens for your use. The first one is in Basic and the second one is in Extended Basic.

Basic Version

```
100 REM TITLE SCREEN
110 CALL CLEAR
120 CALL CHAR(128,"001100110
0110011")
130 CALL SCREEN(16)
140 CALL COLOR(13,1,7)
150 PRINT TAB(5);"BREA 99er'
s PRESENTS":::::TAB(6);"A NE
W TITLE SCREEN":::::::" pre
ss any key to continue"
160 CALL HCHAR(24,1,128,64)
170 CALL VCHAR(1,31,128,96)
180 CALL KEY(0,K,S)
190 IF S=0 THEN 180
200 REM Program continues
```

Extended Basic Version

```
100 ! TITLE SCREEN
110 CALL SCREEN(16):: CALL C
OLOR(13,1,7):: CALL CHAR(128
,"0011001100110011")
120 DISPLAY AT(10,5)ERASE AL
L:"BREA 99er's PRESENTS": :
: : :TAB(6);"A NEW TITLE SCR
EEN": : : : : : : :" press a
ny key to continue"
130 CALL HCHAR(24,1,128,64):
: CALL VCHAR(1,31,128,64)::
CALL KEY(0,K,S):: IF S=0 THE
N 130
140 ! Program continues
```

Type in the program and run it. Then try changing the: CALL CHAR (128,"00110011 00110011") to CALL CHAR(128, "FF81CEA5A5CE81FF") or CALL CHAR(128, "FF81BDA5 A5BD81FF") or CALL CHAR(128, "FFC3A59999A5C3FF")

Now change the call color statement to CALL COLOR (13,7,1) and run each of the examples and you will discover that it is easy the make fancy borders. Give it a try.

HAPPY PROGRAMMING! ■

CHAR & SPRITE EDITOR

Detta program ger dig möjlighet att rita ett tecken med 8x8 eller 16x16 punkter som sedan översätts till hex-kod för CALL CHAR. Det är förbättrat jämfört med originalet från Compute!

Vi har tidigare publicerat två liknande program i PB:
SPR-MAKER för XB PB 84-1
SPR-MAKER2 för XB PB 86-1
Båda dessa använder joystick och har rutmönster för 16x16 punkter men även en markering av gränsen mellan de fyra tecknen.

Andra liknande program är:

CHARDEF från Progr. Aids I, Basic, saknar rutmönster, 8x8 punkter.

DEFCHAR från Compute! 83-05 (Regena Guide to TI/99/4A), Basic, med rutmönster, 8x8 punkter.

EDITSPRITE: Compute! 83-09 (Compute! Collection), XB, saknar rutmönster, 16x16 punkter.

SPR-EDIT: Compute! Sound & Graphics, XB, med rutmönster, 8x8 och 16x16 punkter.

SPRITER: 99'er 82-11 (Best of 99'er Volume 1), XB, saknar rutmönster, 16x16 punkter.

SUPERFONT: Compute! 85-06 (Compute! Collection), XB.

SPRITE BUILDER:
Micropendium Aug 1989.

```
100 REM CHAR EDITOR
105 REM BY L.LONG
110 REM COMPUTE 83-09 XB
115 REM REVJan Alexandersson
120 DIM BIN(16,16)
125 CALL CHAR(100,"000000000
0000000FFFFFFFFFFFFFFFFFFFC
3C3C3C3FFFF"):: CALL COLOR(9
,2,16)
130 CALL CLEAR :: SCREEN=1 :
: COLOR=7 :: CHAR=100
140 INPUT "1 SINGLE  2 DOUB
LE ":Q :: IF Q<>1 AND Q<>2 T
HEN 130
180 CALL CLEAR
190 DISPLAY AT(1,10):"CHAR E
DITOR"
200 FOR R=1 TO 8*Q :: CALL H
CHAR(3+R,4,100,8*Q):: NEXT R
210 CALL MAGNIFY(1):: IF KEY
=84 THEN 217 :: CALL SCREEN(
8)
217 CALL DELSPRITE(ALL)
220 CALL SPRITE(#28,102,14,2
5,25)
225 CALL HCHAR(21,1,32,64)
240 DISPLAY AT(22,2):"E=UP X
=DOWN S=LEFT D=RIGHT":" 1=PI
XEL ON  0=PIXEL OFF":" P=DIS
PLAY SPRITE"
260 R=1 :: C=3
270 CALL KEY(3,KEY,STA):: IF
 STA=0 THEN 270
272 IF KEY=48 THEN CHAR=100
274 IF KEY=49 THEN CHAR=101
280 IF KEY=83 THEN C=C-1 ::
GOTO 320
290 IF KEY=68 THEN C=C+1 ::
GOTO 320
300 IF KEY=69 THEN R=R-1 ::
GOTO 320
310 IF KEY=88 THEN R=R+1 ::
GOTO 320
312 IF KEY=80 THEN 470
320 IF C<3 THEN C=2+8*Q
330 IF C>2+8*Q THEN C=3
340 IF R<1 THEN R=8*Q
350 IF R>8*Q THEN R=1
380 CALL LOCATE(#28,(8*R)+17
,8*C+1)
420 CALL HCHAR(3+R,1+C,CHAR)
430 CALL SOUND(20,200,5)
460 GOTO 270
470 CALL DELSPRITE(ALL)
480 CALL HCHAR(21,1,32,128)
490 DISPLAY AT(22,2):"PLEASE
 WAIT WHILE I THINK"
500 FOR R=1 TO 8*Q
510 FOR C=1 TO 8*Q
520 CALL GCHAR(3+R,3+C,GC)
540 BIN(R,C)=GC-100
550 NEXT C
560 NEXT R
570 HEX$="0123456789ABCDEF"
580 M$=""
590 FOR R=1 TO 8*Q
600 LOW=BIN(R,5)*8+BIN(R,6)*
4+BIN(R,7)*2+BIN(R,8)+1
610 HIGH=BIN(R,1)*8+BIN(R,2)
*4+BIN(R,3)*2+BIN(R,4)+1
620 M$=M$&SEG$(HEX$,HIGH,1)&
SEG$(HEX$,LOW,1)
630 NEXT R
635 IF Q=1 THEN 690
640 FOR R=1 TO 16
650 LOW=BIN(R,13)*8+BIN(R,14
)*4+BIN(R,15)*2+BIN(R,16)+1
660 HIGH=BIN(R,9)*8+BIN(R,10
)*4+BIN(R,11)*2+BIN(R,12)+1
670 M$=M$&SEG$(HEX$,HIGH,1)&
SEG$(HEX$,LOW,1)
680 NEXT R
690 CALL CHAR(104,M$)
700 CALL MAGNIFY(2*Q-1)
710 MM=2*Q-1 :: M=2*Q
730 CALL SPRITE(#1,104,COLOR
,50,170,0,0)
740 DISPLAY AT(21,2):"C COLO
R  M MAGNIFY  T EDIT"
750 DISPLAY AT(22,2):"A ERAS
E  Q QUIT  B BACKGRD"
760 DISPLAY AT(23,2):"E=UP X
=DOWN S=LEFT D=RIGHT"
770 DISPLAY AT(24,8):"L LIST
S PROGRAM"
780 CALL KEY(3,KEY,STA)
790 IF KEY=76 THEN 1010
800 IF KEY=81 THEN END
810 IF KEY=65 THEN 130
812 IF KEY=66 THEN GOSUB 121
0
815 IF KEY=84 THEN 210
820 IF KEY=77 THEN 940
830 IF KEY=67 THEN 1160
840 IF KEY=83 THEN H=H-2
850 IF KEY=68 THEN H=H+2
860 IF KEY=69 THEN V=V-2
870 IF KEY=88 THEN V=V+2
880 IF V>120 THEN V=120
890 IF V<-120 THEN V=-120
900 IF H>120 THEN H=120
910 IF H<-120 THEN H=-120
920 CALL MOTION(#1,V,H)
930 GOTO 780
940 CALL MAGNIFY(M)
950 MM=M
960 IF M=2*Q-1 THEN M=2*Q EL
SE M=2*Q-1
970 FOR D=1 TO 20 :: NEXT D
980 GOTO 780
1000 REM PROGRAM LISTER
1010 CALL CHAR(110,"00282828
")
1020 CALL CLEAR :: CALL DELS
PRITE(ALL)
1030 PRINT "      PROGRAM LI
STING": :
1050 CALL HCHAR(24,2,62):: P
RINT "100 CALL CHAR(96,n";::
 FOR W=1 TO 16*Q*Q :: PRINT
SEG$(M$,W,1);:: NEXT W :: PR
INT "n";")"
1055 CALL HCHAR(24,2,62):: P
RINT "110 CALL SCREEN(";STR$
```

```
(SCREEN);")"
1060 CALL HCHAR(24,2,62):: P
RINT "120 CALL MAGNIFY(";STR
$(MM);")"
1070 CALL HCHAR(24,2,62):: P
RINT "130 CALL SPRITE(#1,96,
";STR$(COLOR);",";"150";",";
"150,";STR$(V);",";STR$(H);"
)"
1150 PRINT : : : : :
1155 DISPLAY AT(21,3):"A - E
RASE   Q - QUIT"
1156 CALL KEY(3,KEY,STA):: I
F STA=0 THEN 1156
1157 IF KEY=81 THEN END
1158 IF KEY=65 THEN 130 ELSE
 1156
1160 COLOR=COLOR+1 :: IF COL
OR>16 THEN 1180
1170 CALL COLOR(#1,COLOR)::
GOTO 780
1180 COLOR=2 :: CALL COLOR(#
1,COLOR):: GOTO 780
1200 REM SCREEN COLOR
1210 SCREEN=SCREEN+1 :: IF S
CREEN=17 THEN SCREEN=2
1220 CALL SCREEN(SCREEN):: R
ETURN                      ■
```

```
100 REM ******************
110 REM *                *
120 REM *     LOTTO      *
130 REM *                *
140 REM *      AV        *
150 REM *                *
160 REM * BENGT FAHLGREN *
170 REM *                *
180 REM *    C 1985      *
190 REM *                *
200 REM ******************
210 GOSUB 8000
220 DIM A(36),B(36),M(28),TA
L(7)
230 CALL SCREEN(8)
240 FOR I=1 TO 8
250 CALL COLOR(I,2,8)
260 NEXT I
270 CALL CLEAR
280 CALL COLOR(9,1,8)
290 CALL COLOR(10,2,8)
300 CALL COLOR(13,1,8)
310 CALL COLOR(14,2,8)
320 CALL CHAR(96,"")
330 CALL CHAR(104,"00FFFF")
340 CALL CHAR(128,"FFFFFFFFF
FFFFFFF")
350 CALL CHAR(136,"")
360 DATA 96,128,96,96,96,96,
128,128,128,128,96,128,128,1
28,128,128,96,128,128,128,12
8,128,96,128,128,128,128
370 DATA 96
380 DATA 96,128,96,96,96,96,
128,96,96,128,96,96,96,128,9
6,96,96,96,96,128,96,96,96,1
28,96,96,128,96
390 DATA 96,128,128,128,128,
96,128,128,128,128,96,96,96,
128,96,96,96,96,96,128,96,96
,96,128,128,128,128,96
400 FOR I=1 TO 28
410 READ M(I)
420 H1$=H1$&CHR$(M(I))
430 NEXT I
440 FOR I=1 TO 28
450 READ M(I)
460 H2$=H2$&CHR$(M(I))
470 NEXT I
480 FOR I=1 TO 28
490 READ M(I)
500 H5$=H5$&CHR$(M(I))
510 NEXT I
520 PRINT "````````````````
``````````"
530 PRINT H1$:H2$:H2$:H2$:H5
$
540 PRINT "````````````````
``````````"
550 PRINT : : : : : : : : :
: : : : : : :
560 CALL COLOR(13,4,4)
570 CALL COLOR(9,16,16)
580 RAD=9
590 KOL=8
600 Z$="C 1985  B.FAHLGREN"
610 GOSUB 2460
620 REM
630 REM  MENY UTSKRIFT
640 REM
650 CALL HCHAR(12,1,32,416)
660 RAD=12
670 KOL=11
680 Z$="--MENY--"
690 GOSUB 2460
700 RAD=14
710 KOL=6
720 Z$="SIFFERVAL:"
730 GOSUB 2460
740 RAD=16
750 KOL=10
760 Z$="1)  START"
770 GOSUB 2460
780 RAD=18
790 KOL=10
800 Z$="2)  SYSTEM"
810 GOSUB 2460
820 RAD=20
830 KOL=10
840 Z$="3)  INFORMATION"
850 GOSUB 2460
860 RAD=22
870 KOL=10
880 Z$="4)  KLAR"
890 GOSUB 2460
900 REM
910 REM VAL AV PROCEDUR
920 REM
930 CALL KEY(3,K,S)
940 IF S=0 THEN 930
950 IF (K=49)+(K=50)+(K=51)+
(K=52)THEN 960 ELSE 930
960 ON K-48 GOTO 1390,1030,1
060,2520
970 REM
980 REM
990 REM
1000 REM
1010 REM
1020 REM
1030 REM START SYSTEM
1040 REM SLUT  SYSTEM
1050 REM START INFO
1060 CALL HCHAR(12,1,32,416)
1070 RAD=12
1080 KOL=9
1090 Z$="--INFORMATION--"
1100 GOSUB 2460
1110 RAD=13
1120 KOL=3
1130 Z$="DETTA PROGRAM SIMUL
ERAR EN"
1140 GOSUB 2460
1150 RAD=14
1160 KOL=3
1170 Z$="RIKTIG LOTTO-DRAGNI
NG."
1180 GOSUB 2460
1190 RAD=15
1200 KOL=3
1210 Z$="PROGRAMMET V[LJER B
ARA BLAND"
1220 GOSUB 2460
1230 RAD=16
1240 KOL=3
1250 Z$="DOM TAL SOM 'LIGGER
' KVAR I"
1260 GOSUB 2460
1270 RAD=17
1280 KOL=3
1290 Z$="HATTEN. PROGRAMMET
KAN K\RAS"
1291 GOSUB 2460
1300 RAD=18
1310 KOL=3
1320 Z$="I B]DE TI-BASIC OCH
 X-BASIC"
1325 GOSUB 2460
1330 RAD=21
1331 KOL=6
1332 Z$="*** MYCKET N\JE ***
"
1333 GOSUB 2460
1334 RAD=24
1335 KOL=3
1336 Z$="KLAR? (J/N)"
```

```
1340 GOSUB 2460
1342 CALL KEY(3,K,S)
1343 IF S=0 THEN 1342
1344 IF (K=74)+(K=78)THEN 13
45 ELSE 1343
1345 IF K=74 THEN 650
1346 IF K=78 THEN 1342
1350 GOTO 650
1360 REM SLUT INFO
1370 REM
1380 REM
1390 CALL HCHAR(12,1,32,384)
1400 RAD=12
1410 KOL=3
1420 Z$=" 1   2   3   4   5
  6   7"
1430 GOSUB 2460
1440 CALL HCHAR(13,1,104,32)
1450 REM
1460 REM  INITTIERA
1470 RANDOMIZE
1480 REM
1490 REM ANTAL VALMOEJLIGHET
1500 LIMIT=35
1510 CALL COLOR(13,10,10)
1520 CALL HCHAR(24,1,32,32)
1530 CALL HCHAR(14,1,32,32)
1540 REM MAERK KULORNA
1550 REM (35 ST) MED SIFF.
1560 REM 1 T.O.M. 35
1570 FOR P=1 TO 35
1580 A(P)=P
1590 NEXT P
1600 REM
1610 REM STARTA DRAGNINGEN
1620 REM VAELJ 7 KULOR
1630 FOR I=1 TO 7
1640 SLUMP=INT(RND*LIMIT)+1
1650 REM
1660 REM TAG DE KULOR MED
1670 REM LAEGRE NUMMER AEN
1680 REM SLUMP OCH LAEGG
1690 REM DESSA I FAELT B
1700 FOR J=1 TO ABS(SLUMP-1)
1710 B(J)=A(J)
1720 NEXT J
1730 REM
1740 REM TAG DE KULOR MED
1750 REM HOEGRE NUMMER AEN
1760 REM SLUMP OCH LAEGG
1770 REM DESSA I FAELT B
1780 REM
1790 FOR K=SLUMP TO LIMIT
1800 B(K)=A(K+1)
1810 NEXT K
1820 TAL(I)=A(SLUMP)
1830 A(INT(LIMIT))=0
1840 REM
1850 REM FAELT A = FAELT B
1860 REM
1870 FOR L=1 TO LIMIT
1880 A(L)=B(L)
1890 NEXT L
1900 REM
1910 REM MINSKA ANT KULOR
1920 REM MED 1
1930 REM
1940 LIMIT=LIMIT-1
1950 NEXT I
1960 REM
1970 REM
1980 REM SORTERING AV DE
1990 REM DRAGNA TALEN
2000 CALL COLOR(13,11,11)
2010 FOR I=1 TO 6
2020 FOR J=I TO 7
2030 IF TAL(I)<=TAL(J)THEN 2
070
2040 TEMP=TAL(I)
2050 TAL(I)=TAL(J)
2060 TAL(J)=TEMP
2070 NEXT J
2080 NEXT I
2090 CALL COLOR(13,4,4)
2100 REM UTSKRIFT AV DE SORT
2110 REM TALEN
2120 REM
2130 REM
2140 RAD=14
2150 KOL=4
2160 Z$=STR$(TAL(1))
2170 GOSUB 2460
2180 KOL=8
2190 Z$=STR$(TAL(2))
2200 GOSUB 2460
2210 KOL=12
2220 Z$=STR$(TAL(3))
2230 GOSUB 2460
2240 KOL=16
2250 Z$=STR$(TAL(4))
2260 GOSUB 2460
2270 KOL=20
2280 Z$=STR$(TAL(5))
2290 GOSUB 2460
2300 KOL=24
2310 Z$=STR$(TAL(6))
2320 GOSUB 2460
2330 KOL=28
2340 Z$=STR$(TAL(7))
2350 GOSUB 2460
2360 RAD=24
2370 KOL=3
2380 Z$="FLER RADER? (J/N)"
2390 GOSUB 2460
2400 CALL KEY(3,K,S)
2410 IF S=0 THEN 2400
2420 IF (K=74)+(K=78)THEN 24
30 ELSE 2400
2430 IF K=74 THEN 1500
2440 IF K=78 THEN 650
2450 REM MARKORSTYRNING
2460 FOR Z=1 TO LEN(Z$)
2470 CALL VCHAR(RAD,KOL,ASC(
SEG$(Z$,Z,1)))
2480 KOL=1-KOL*(KOL<>32)
2490 RAD=RAD-(KOL=1)
2500 NEXT Z
2510 RETURN
2520 CALL CLEAR
2530 END
8000 CALL CHAR(91,"002800384
47C4444")
8010 CALL CHAR(92,"0028007C4
444447C")
8020 CALL CHAR(93,"003828384
47C4444")
8030 RETURN
```

■

---

### MYARC HFDC PÅ CRU >1800

*av Jan Alexandersson*

Ja har flyttat min HFDC från CRU >1800 till CRU >1200 och då fungerar Myarc DM V igen. Trolig orsak AVPC CRU >1400?

---

### PENSIONSBERÄKNING OCH CC-40, TI-74, TI-99/4A

CC-40 och TI-74 Basic är mycket lik TI-99/4A XB så att det är enkelt att ändra från originalet till TI-99/4A. De ändringar som behövdes var efter INPUT ledtext : istället för ; och satsseparator :: istället för :. Dessutom har PRINT-formatet ändrats med USING eller INT.

CC-40 och TI-74 har tidigare behandlats i följande äldre nummer av Programbiten och Nittinian. År 1983 var Programbiten och Nittinian två helt olika tidningar:

| | |
|---|---|
| PB 83-1.08 | CC-40 Bo.Nordlin |
| NN 83-1.07 | CC-40 Gustavsson |
| PB 86-5.14 | TI-74 C.Schibler |
| PB 87-1.15 | TI-74 Annons |
| PB 88-1.15 | TI-74 A.Persson |
| PB 90-4.28 | PC-Interface till TI-74 |
| PB 90-4.30 | Förnyelse av PB |
| PB 92-2.02 | Artiklar från Charles Good |

Charles Good från Lima Ohio User Group har skrivit många artiklar om CC-40 och TI-74. BB&P May 1992 var ett specialnummer om CC-40 och TI-74. BB&P Adress: Lima OH 99/4A User Group, P.O. Box 647, Venedocia, OH 45894, USA.

BB&P Apr 1993 innehåller en artikel om nytillverkade Hexbus periferienheter:

- 99/4A Hexbus-interface för anslutning av TI-99/4A till periferienheter som ursprungligen har tillverkats för CC-40/TI-74.

- 5,25 tums DS/DD Hexbus diskkontroll.

- Hexbus Video interface för anslutning av CC-40/TI-74 till extern kompositmonitor.

BB&P May 1993: PC-Interface for CC-40 and TI-74.

```
100 REM Pensionsberakningar
v1.0 . Harry Nilsson 911223
110 REM *** OBS endast ungef
arliga resultat ***
120 REM Rev. for TI-99/4A XB
130 REM Original CC-40/TI-74
140 CALL CLEAR
150 INPUT "Manadskostn.? ":M
K :: M=MK*12 ! Pensions prem
ie/man
160 INPUT "Infl.? ":I :: I=I
/100
170 INPUT "Aterbaring, Ranta
 ? ":R :: R=R/100
180 INPUT "Antal ar ? ":N !
Antal ar fram till pens.alde
r
190 INPUT "Marginalskatt ? "
:MS :: MS=MS/100 ! I procent
200 PRINT
210 FOR X=1 TO N
220 MX=(M/(1+I)^X)*(1-MS)::
K=K+MX
230 SN=(SN+M)*(1+R):: PRINT
USING "## #####  ########":X,
MX,SN
240 NEXT X
250 PRINT
260 KN=M*N*(1-MS):: PRINT "T
ot.kostn.nom= ";KN ! Total p
remiekostn.
270 PRINT "Tot.kostn.d.pv=";
INT(K)! Total premiekostn. i
 dagens penningvarde
280 PRINT "Tot.summa nom= ";
INT(SN)! Summa pens.utbetaln
ing
290 S=SN/(1+I)^N :: PRINT "T
ot.summa d.pv=";INT(S)! Pens
.utb. dagens penningvarde
300 END
```

■

## ALPHA BLAST — SPEL XB

*av Dave Miller, USA*

Detta spel går ut på att markera rätt alfabetisk ordning av fyra slumpmässiga bokstäver genom att dra joysticken mot bokstäverna i tur och ordning. Du har en begränsad tid att välja bokstav och tiden blir kortare för varje ny uppsättning om fyra bokstäver. Du kan istället använda ESDX-tangenterna om du laddar in programmet SUB JOYST i PB 93-2.05 med MERGE. Du kan själv välja andra tangenter genom att ändra i SUB JOYST.

```
70 REM ALPHA BLAST
80 REM BY D.MILLER
90 REM COMPUTE 83-11 XB JS
100 GOSUB 510
110 RANDOMIZE
120 DIM N(3)
130 CALL CLEAR :: CALL SCREE
N(16)
140 CALL HCHAR(8,5,120,24)::
 DISPLAY AT(10,4):"A L P H A
 -- B L A S T" :: CALL HCHAR
(12,5,120,24)
150 CALL MAGNIFY(2):: FOR L=
1 TO 28
160 CALL SPRITE(#L,INT(RND*2
6)+65,INT(RND*13)+3,INT(RND*
24)*8+1,INT(RND*32)*8+1,INT(
RND*60)-30,INT(RND*60)-30)
170 IF L=25 THEN DISPLAY AT(
21,10):"GET READY!"
180 NEXT L :: CALL DELSPRITE
(ALL):: CALL CLEAR :: HS=0
190 CALL COLOR(12,6,1)
200 DISPLAY AT(1,6):"HIGH SC
ORE:";HS :: U,R,SC=0
210 U=U+.03*SGN(1-U):: R=R+1
 :: DISPLAY AT(5,14):"ROUND
#";R :: DISPLAY AT(2,6):"SCO
RE:     ";SC
220 FOR I=6 TO 21 :: CALL HC
HAR(I,6,128):: NEXT I
230 FOR I=5 TO 7 STEP 2 :: C
ALL VCHAR(5,I,95,17):: NEXT
I
240 FOR I=3 TO 9 STEP 6 :: C
ALL VCHAR(4,I,120,20):: NEXT
 I :: CALL HCHAR(4,4,120,5):
: CALL HCHAR(23,4,120,5)
250 FOR I=0 TO 3
260 N(I)=INT(RND*26)+65
270 FOR J=0 TO I-1 :: IF N(J
)=N(I)THEN 260
280 NEXT J :: NEXT I
290 CALL SPRITE(#6,42,3,97,1
53)
300 CALL SPRITE(#2,N(0),14,5
7,153):: CALL SPRITE(#3,N(1)
,14,97,201):: CALL SPRITE(#4
,N(2),14,137,153):: CALL SPR
ITE(#5,N(3),14,97,105)
310 ROW=21 :: A,B,C,D=-1
320 T=0
330 CALL JOYST(1,X,Y):: IF A
BS(X)+ABS(Y)<>4 THEN CALL HC
HAR(ROW,6,32):: ROW=ROW-U ::
 IF ROW<5 THEN 400 ELSE 330
340 IF (X=0)*(Y=4)*(A)THEN C
ALL PATTERN(#2,32,#6,43):: V
(T),A=0 :: GOTO 390
350 IF (X=4)*(Y=0)*(B)THEN C
ALL PATTERN(#3,32,#6,43):: V
(T)=1 :: B=0 :: GOTO 390
360 IF (X=0)*(Y=-4)*(C)THEN
CALL PATTERN(#4,32,#6,43)::
V(T)=2 :: C=0 :: GOTO 390
370 IF (X=-4)*(Y=0)*(D)THEN
CALL PATTERN(#5,32,#6,43)::
V(T)=3 :: D=0 :: GOTO 390
380 CALL HCHAR(ROW,6,32):: R
OW=ROW-U :: IF ROW<5 THEN 40
0 ELSE 330
390 CALL SOUND(-10,200,2)::
CALL PATTERN(#6,42):: T=T+1
:: IF T=4 THEN 450 ELSE 330
400 DISPLAY AT(22,11):"YOUR
TIME IS UP!"
410 CALL SOUND(800,110,5,120
,5):: FOR I=1 TO 200 :: NEXT
 I
420 DISPLAY AT(24,10):"PLAY
AGAIN(Y/N)?" :: IF SC>HS THE
N HS=SC
430 CALL KEY(3,KEY,ST):: IF
ST=0 THEN 430
440 IF KEY=89 THEN CALL CLEA
R :: CALL DELSPRITE(ALL):: G
OTO 200 ELSE END
450 REM EVALUATE ANSWERS
460 FOR T=0 TO 2 :: IF N(V(T
))<N(V(T+1))THEN 480
470 SC=SC-INT(1.5*R*ROW):: G
OTO 490
480 SC=SC+INT(R*ROW)
490 NEXT T
500 CALL DELSPRITE(ALL):: GO
TO 210
510 REM CHAR
520 CALL COLOR(14,9,1)
530 CALL CHAR(120,"007E7E7E7
E7E7E",128,"")
540 CALL COLOR(12,6,10,13,1,
9)
550 RETURN
```

■