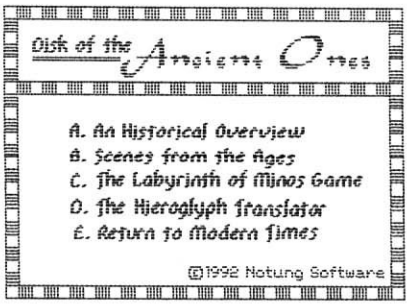
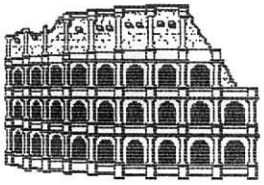
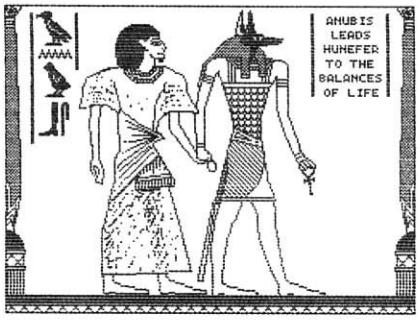
PROGRAM BITEN 93-2







ISSN 0281-1146

REDAKTÖREN

Thomas Kolk har nu sålt sin Myarc Geneve till: Niklas Johansson, Moråvägen 2, Hogstad, 595 93 MJÖLBY

Niklas fick även ett modem (HEATH 1200) tillsammans med Geneve. Han behöver hjälp med att använda detta mot en BBS. Du som har några råd och tips kan kontakta Niklas direkt.

Frank Aylstock, Brea Users Group, beskriver i Micropendium Feb 93 hur man kan använda DSK3 trots att diskdriven endast har byglar för DSK1 och DSK2. Du ordnar DSK3 genom att ansluta pinne 14 (för DSK3) från kabelsidan till pinne 10 (för DSK1) på drive-sidan. Bryt samtidigt förbindelsen pinne 10 - 10 och pinne 14 -14. Bygla som DSK1. DSK4 kan ordnas med pinne 6 mot pinne 12. ■

HORIZON PROBLEM

av Jan Alexandersson

Horizon RAM-disk kan i vissa ogynnsamma situationer förstöra katalogen på en hårddisk eller flexskiva. Enligt Charles Good, Lima Ohio UG (March 1993), är orsaken att ROS 8.14 eller 8.14B samverkar med andra kort och program som ger CALL FILES (4) eller högre. Problemet beskrevs först som "I GOTCHA!" virus trots att det inte är ett virus utan beror på att kortkonstruktörerna inte har följt TI:s regler till 100 %.

Kort som kan ge problem är Myarc HFDC, Dijit AVPC, Mechatronic 80 col, OPA TIM eller OPA POP-cart. Det finns även vissa Horizon 2000 och 3000 kort med konstruktionsfel (felaktig tändspänning på LED) som kan ge problem även utan andra kort.

Program som använder CALL FILES(4) internt är Archiver, TI-Base, FW Assembler, BOOT före dec 89 och DM-1000.

Det finns patchar till ROS som undanröjer problemet. Bud Mills Services byter ut felaktig ROS mot ROS 8.14C om ROS-skivan, svarskuvert och returporto skickas in. Du kan även få en kopia av själva patcharna till ROS 8.14 (inte hela programmet) om du kontaktar redaktören. ■

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

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

Postgiro 19 83 00-6
Medlemsavgiften för 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 TRÄNGSUND
Tel. 08-771 0569
(Ring eller skriv till mig om du har frågor om program eller hårdvara)

ÅRSREDOVISNING FÖR ÅR 1992

FÖRENINGEN PROGRAMBITEN
ÅRSBERÄTTELSE 1993-03-06
STYRELSENS ÅRSREDOVISNING

Verksamheten 1992 har bedrivits i form av tidningsutgivning och programförmedling. Antalet medlemmar har minskat från 55 till 47.

STYRELSE OCH FUNKTIONÄRER

Föreningens styrelse:
Claes Schibler, ordf. och registerf.
John Hanssen, kassör
Åke Olsson, sekreterare
Jan Alexandersson, redaktör
Börje Häll, programbankir
Peter Olsson

FÖRENINGENS VERKSAMHET

Fyra nummer av tidningen Programbiten har framställts.

Arvode till funktionärer har ej utgått.

BALANSRÄKNING

INGÅENDE BALANS 1992

TILLGÅNGAR		SKULDER	
POSTGIRO	10341.37	LEV.SKULDER	00.00
VARULAGER	00.00	FÖRBET.MEDL.AVG.	120.00
SKATTEFORDRINGAR	00.00	EGET KAPITAL	10221.37
SUMMA	10341.37	SUMMA	10341.37

UTGÅENDE BALANS 1992

TILLGÅNGAR		SKULDER	
POSTGIRO	9733.74	LEVERANTÖRSSKULDER	00.00
VARULAGER	00.00	FÖRBET.MEDL.AVG.	1560.00
SKATTEFORDR.	00.00	EGET KAPITAL	8173.74
SUMMA	9733.74	SUMMA	9733.74

RESULTATRÄKNING

INTÄKTER		KOSTNADER	
MEDLEMSAVGIFTER	6840.00	TRYCKKOSTNADER	4281.00
PROGRAMFÖRSÄLJN.	1090.00	PROGRAMINKÖP	00.00
RÄNTOR	316.90	LOKALHYROR	350.00
ÅTERBET.PORTO	374.90	FÖRBR.MAT/PORTO	740.60
ÅRETS FÖRLUST	607.33	PORTO/KUVERT	3857.53
SUMMA	9229.13	SUMMA	9229.13

STOCKHOLM SOM OVAN
CLAES SCHIBLER JOHN HANSSEN

BASIC OCH XB TIPS - 2

av Jan Alexandersson

CALL CHAR

Om du har omdefinierat tecken 128-159 kommer dessa inte att försvinna när programmet är slut. NEW kommer inte heller att sudda ut dessa.

Om du använder kassett kan du utnyttja denna finess men om du istället använder flexskiva kommer tecknen att ändras till något oönskat. De nollställs nämligen inte utan får ett slumpartat innehåll.

CALL HCHAR och PRINT

Det är en viktig skillnad mellan Basic och Extended Basic om CALL HCHAR(24,X,Y) åtföljs av PRINT. Prova följande:

```
100 CALL HCHAR(24,3,65)
110 PRINT :"B"
```

I Basic får man A och B under varandra som önskat. I Extended Basic kommer först A att skrivas som omedelbart suddas av B. Detta är saker som man måste tänka på när program ska flyttas från Basic till Extended Basic.

CALL CHARSET

Detta återställer endast ASCII-kod 32-95 till ursprungligt CHAR men ej tecken 96-143. Samtidigt återställes COLOR till ursprungligt värde för alla ASCII-koder 30-143 vilket är mycket ologiskt.

CALL KEY

Om du använder delat tangentbord d.v.s. CALL KEY(1,KEY,STA) eller CALL KEY(2,KEY,STA) är det ej tillåtet att testa IF KEY=0 THEN ... i Basic utan man måste skriva IF KEY+1 =1 THEN ... Det går dock bra att testa andra tangentvärden än 0 på vanligt sätt. I Extended Basic är det även tillåtet att testa KEY=0.

CALL KEY(0,KEY,STA) testar endast tidigare tangentbord 3, 4 eller 5 men ej tangentbord 1 och 2. I normala program ska man alltid använda CALL KEY(3,KEY,STA) så slipper man extra programrader som tittar på J och j respektive N och n. Anledningen till att 0 står i alla programlistningar är att man vill kunna köra i både TI-99/4 och TI-99/4A. Vi som bara har 99/4A bör skriva 3.

CALL KEY har den egenheten att det även påverkar möjligheten för INPUT och ACCEPT att mata in små bokstäver. Detta ger en sorts VALIDATE även i Basic. Prova följande program med ALPHA LOCK uppe:

```
100 CALL KEY(5,K,S)
110 INPUT A$
120 PRINT A$
130 CALL KEY(3,K,S)
140 INPUT A$
150 PRINT A$
160 GOTO 100
```

Du ser att varannan INPUT-sats accepterar små bokstäver. Genom att ändra från tangentbord 5 till 3 får du VALIDATE även i vanlig Basic.

VÄNTA PÅ JA ELLER NEJ MED CALL KEY

Ett vanligt sätt att skriva detta är följande:

```
100 PRINT "FORTSÄTTA J/N"
110 CALL KEY(0,K,S)
120 IF (K=78)+(K=110) THEN 140
130 IF (K=74)+(K=106) THEN 100 ELSE
    110
140 END
```

Samma resultat får du med CALL KEY (5,K,S). Om du använder tangentbord 3 kan man skriva detta mycket enklare. Du kan göra så här:

```
100 PRINT "FORTSÄTTA J/N"
110 CALL KEY(3,K,S)
120 ON 1-(K=78)-2*(K=74) GOTO
    110,140,100
140 END
```

På rad 120 används ett logiskt uttryck som blir 1 om varken J eller N har tryckts. Om K=78 är sannt blir uttrycket 1+1=2 eftersom sannt betecknas med -1 för TI-99/4A.

Det är alltid bättre att ha CALL KEY i menyer än INPUT så att man slipper att trycka på ENTER.

Du kan även använda POS för att välja JA och NEJ:

```
100 PRINT "FORTSÄTTA J/N"
110 CALL KEY(3,K,S)
120 IF S<1 THEN 110
130 ON 1+POS("JN",CHR$(K),1)GOTO
    110,100,140
140 END
```

AVBRYTA PÅGÅENDE CALL KEY LOOP

En ytterligare användning av CALL KEY är brytande av en pågående loop. Följande exempel visar detta:

```
100 PRINT "DET SNURRAR RUNT"
110 CALL KEY(3,K,S)
120 IF S=0 THEN 140
130 GOSUB 150
140 GOTO 100
150 CALL KEY(3,K,S)
160 IF S<1 THEN 150
170 RETURN
```

Om du trycker tangenten en gång så stannar PRINT-loopen. Tryck en gång till och det snurrar igen. Det är viktigt att både S=-1 och S=1 gör att man hoppar till subrutinen på rad 150. Om du inte gör detta är risken stor att att du aldrig får stopp på snurran oberoende av hur länge du trycker. Problemet uppstår när du snabbt efter det du tryckt igång vill stoppa snurran.

TI har misslyckats med detta när det gäller LIST i Extended Basic. Du måste där vänta någon extra rad så att CALL KEY eller motsvarande har passerats en gång efter starten utan att någon tangent har tryckts.

VÄNTAN PÅ UPPREPAD TANGENTTRYCKNING

Ofta görs många tangenttryckningar efter varandra när man passerar olika CALL KEY-satser. Det är då lämpligt att testa S<1 istället för S=0. Du kan annars få irriterande dubbeltryckningar. Exempel:

```
100 CALL KEY(3,K,S)
110 IF S<1 THEN 100
120 A$=A$&CHR$(K)
130 IF K<>13 THEN 100
140 PRINT A$
```

CALL JOYST

Om man saknar joystick eller tycker att programmet fungerar bättre med tangenttryckningar, kan man ändra underprogrammet i Extended Basic på följande sätt:

```
9000 !@P+
9010 SUB JOYST(NR,X,Y)
9020 CALL KEY(NR,K,S)
9030 X=-4*(K=3)+4*(K=2)
9040 Y=-4*(K=5)+4*(K=0)
9050 SUBEND
```

Detta underprogram kan man knappa in i slutet av ett gammalt program. Någon ändring i själva huvudprogrammet behöver inte göras. Du behöver således inte leta upp alla olika programrader där CALL JOYST finns. Det är ingen risk med att NR, X, Y, K och S även används för annat i huvudprogrammet. K och S är lokala variabler medan NR, X och Y endast är utfyllnad för att markera ett överfört talvärde.

I vissa program har man testat FIRE-knappen på joysticken genom att testa om någon tangent, vilken som helst, har tryckts. Du måste då ändra denna rad med CALL KEY så att endast K=18 som motsvarar FIRE testas.

Om man har tillgång till flexskiva går det att spara ovanstående program med SAVE DSK1.JOYST,MERGE. När man sedan har ett färdigt program i datorn som man vill ändra från joystick till tangenter, måste man först ta reda på om radnummer 9000-9050 redan används i programmet. Skriv LIST 9000. Om datorn då tar fram en rad med lägre nummer än 9000 är allt bra. Använd annars RES av huvudprogrammet eller underprogrammet så att de ej kolliderar. Skriv sedan MERGE DSK1.JOYST.

Om man endast har Basic eller programmet inte fungerar i Extended Basic måste alla rader med CALL JOYST letas upp och ersättas med två nya rader:

```
xxxx NR=1
xxxy GOSUB 9010
```

I slutet av programmet skriv följande:

```
9010 REM JOYST(NR,X,Y)
9020 CALL KEY(NR,K,S)
```

```
9030 X=-4*(K=3)+4*(K=2)
9040 Y=-4*(K=5)+4*(K+1=1)
9050 RETURN
```

Jämfört med det tidigare programmet behöver du endast ändra SUB till REM och SUBEND till RETURN samt testa K+1=1 istället för K=0 p.g.a. en bug i Basic. I Basic kan variablerna kollidera med andra i huvudprogrammet. Det är viktigt att K inte lagrar något värde för t.ex. FIRE från joystick som senare ska användas. Vid problem använd andra variabelnamn. Om du vill använda joystick 2 skriv xxxx NR=2 eller om programmet använder en variabel skriv xxxx NR=VARIABEL.

Inget av de beskrivna programmen tar hänsyn till diagonalerna för JOYST. Den som använder dessa kan modifiera programmet. Det är dock ganska krångligt att komma ihåg var diagonalerna sitter i snabba spelprogram.

CALL SOUND

Datorn är konstruerad i USA där frekvensen i elnätet är 60 Hz. Eftersom vi endast har 50 Hz kommer datorn att räkna för långsamt. CALL SOUND(1000,FRE,VOL) kommer att hålla på (60/50)*1000 ms d.v.s. 1,2 sek. På motsvarande sätt blir den längsta tillåtna tiden 5,12 sek om man skriver CALL SOUND(4250,FRE,VOL).

Denna förlängning med 20 % gäller endast om datorn bara har CALL SOUND att tänka på. I praktiken vill man ha ljud och musik samtidigt som grafiken förändras eller datorn gör uträkningar. Datorn hinner då inte med att räkna ned ljudkommandot utan man får en avsevärd ytterligare förlängning. I värsta fall tar det dubbelt så lång tid för CALL SOUND när det följs av andra beräkningar. I praktiken är det svårt att förutse tiden. Man måste istället prova sig fram med stoppur och ändra siffervärdet tills det stämmer.

Det finns ett sätt för programmet att ta reda på om datorn körs på 50 eller 60 Hz genom CALL PEEK på CPU-adress >000C (decimalt 12). Dess värde är >30 (decimalt 48) med 50 Hz och >28 med 60 Hz. Följande program ställer in varaktigheten för CALL SOUND till 0,5 sek oberoende av var programmet körs. RS232-kortet gör på liknande sätt för att få rätt bithastighet.

```
100 CALL PEEK(12,A)
110 T=50/(50-10*(A=48))
120 CALL SOUND(T*500,440,0)
```

CALL VERSION

CALL VERSION(X) returnerar X=110 i Extended Basic medan Triton Super-XB ger X=120. Det finns ett annat sätt att testa om programmet körs i vanlig Basic eller Extended Basic:

```
100 RANDOMIZE(0)
110 IF INT(RND*10)=8 THEN 140
120 PRINT "Extended Basic"
130 END
140 PRINT "Basic"
150 END
```

CALL SAY

Om du gör ett program för Speech Synthesizer bör du se till att samma program kan köras oberoende av om den verkligen är ansluten. I början av programmet lägger du in CALL PEEK (-28672,SP). Alla satser med tal inleds sedan med IF SP THEN CALL SAY(ORD$).

CALL FILES

Underprogrammet CALL FILES finns i diskkontrollkortet. Det bestämmer hur många filer som samtidigt kan vara öppna. Det ställs alltid in på CALL FILES(3) vid "power-up" av diskkontrollkortet. Du kan både öka och minska detta genom att skriva CALL FILES i kommandomod innan du laddar in programmet. CALL FILES ockuperar VDP-RAM i förhållande till det maximala antalet filer som anges av värdet efter CALL FILES. Värdet kan vara 1-9. Skriv alltid NEW efter CALL FILES.

Du kan inte skriva CALL FILES(0) eftersom det ger "Syntax Error" men du kan istället poka in det med CALL INIT följt av CALL LOAD(-31888,63, 255) och NEW. Du har ingen användning av CALL FILES(0) med TI:s eller Corcomps diskkontrollkort, men om du

har Myarcs diskkontrollkort, Myarc HFDC eller Horizon RAM-disk med ROS 8 kan du frigöra VDP-RAM till programkörningen. Myarc och Horizon använder nämligen RAM inne på kortet som diskbuffert så att VDP-RAM inte behövs. Horizon ROS 7 eller tidigare använder dock VDP-RAM som diskbuffert.

ARTIKLAR PÅ SVENSKA I PROGRAMBITEN

84-3.04 Utmaningen: Funktioner, procedurer och SUB-program
84-4.04 Utmaningen: Telefonregister
84-4.26 Tips och tricks
84-5.12 CALL FILES(0)
85-3.06 Kommandon i Basic och XB
85-4.13 Minnesutrymme
85-4.25 CALL SAY
85-4.26 CALL KEY
86-1.03 Garbage collection
86-1.04 ACCEPT med mer än 28 tecken
87-2.13 OPEN av en P-markerad fil
87-4, 88-1, 90-1--6, 91-1 XB-skola
88-2.16 Tillbaka till Basic
90-2--3 Datafiler utan tårar
91-2--3 99/4A från grunden
93-1.08 Basic och XB Tips - 1

Artikeln "Tillbaka till Basic" i PB 88-2 kan betraktas som del 3 av denna artikelserie men kommer inte att publiceras på nytt eftersom inga ändringar behöver göras. Du kan fortfarande köpa äldre nummer av Nittinian/Programbiten, se sidan 2 i tidningen.

ARTIKLAR PÅ ENGELSKA I PROGRAMBITEN

89-4.19 Extended Basic
90-1 -- 91-1 Swedlow Extended Basic
90-4, 90-5, 91-3 HTA Basic & XB
91-3 -- 93-1 Fast Extended Basic! ■

P-GRAM OCH P-GRAM+

av Jan Alexandersson

P-GRAM med 72 kbytes från Sept 1988 har hela tiden kostat 200 $(färdig). P-GRAM+ med 192 kbytes som kom i Nov 1989 har kostat byggd med klocka:

Nov 1989	300 $
Apr 1990	290 $
Nov 1990	280 $
Jul 1991	250 $
Mar 1992	230 $

LÄSA KLOCKA FRÅN P-GRAM

Klockan kan läsas på två olika sätt:

- Som Corcomp med OPEN #1:"CLOCK" och INPUT #1:WEEK$,DATE$,TIME$

- Som MBP från CPU-adresserna:
c>8640 tiotusendelar
(avrundat till tusendelar)
c>8642 hundradelar
c>8644 sekunder
c>8646 minuter
c>8648 timmar
c>864A veckodag
c>864C dag
c>864E månad

```
100 REM READ PGRAM CLOCK1
130 REM READ AS CORCOMP CLK
140 OPEN #1:"CLOCK"
150 INPUT #1:W$,D$,T$
160 CLOSE #1
170 CALL CLEAR
180 PRINT "AS CORCOMP CLOCK"
:"DAY OF WEEK ";W$:"DATE ";D
$:"TIME ";T$: : : :
190 REM READ AS MBP CLOCK
200 DEF BD(X)=X-6*INT(X/16)
210 CALL PEEK(-31168,T,@,H,@
,A,@,B,@,C,@,D,@,E,@,F)
220 PRINT "AS MBP CLOCK"
230 PRINT "1/10000",BD(T)
240 PRINT "1/100",BD(H)
250 PRINT "SECOND",BD(A)
260 PRINT "MINUTE",BD(B)
270 PRINT "HOUR",BD(C)
280 PRINT "DAY OF WEEK",BD(D)
290 PRINT "DAY",BD(E)
300 PRINT "MONTH",BD(F)
```

Enligt Charles Good kan du läsa in klockan till TI-Writers editor med Load File av 0 1 1 CLOCK som ger detta resultat: 0,01/03/93,19:19:16 (veckodag,datum,klockslag).

ATT JÄMFÖRA TVÅ TIDER

Om du läser dessa värden vid två o-

lika tillfällen och räknar ut skillnaden så kan det ibland ge fel värde. Detta beror på att det inte går att läsa t.ex. sekund, hundradel och tiotusendel exakt samtidigt. Programmet visar att det ibland blir ±0.01 sek och mera sällan ±1 sek.

```
100 REM READ PGRAM CLOCK2
110 REM TIME DIFFERENCE
120 REM ERROR, NOT GOOD
150 DEF BD(X)=X-6*INT(X/16)
160 CALL PEEK(-31168,TA,@,HA,@,MA)
170 CALL PEEK(-31168,TB,@,HB,@,MB)
180 M1=BD(MA)
190 M2=BD(MB)
200 H1=BD(HA)
210 H2=BD(HB)
220 T1=BD(TA)
230 T2=BD(TB)
240 PRINT STR$(M1);TAB(4);ST
R$(H1);TAB(7);STR$(T1);
250 PRINT TAB(12);STR$(M2);T
AB(15);STR$(H2);TAB(18);STR$(T2);
260 PRINT TAB(22);M2+H2/100+
T2/10000-M1-H1/100-T1/10000
270 PRINT
280 GOTO 160
```

Följande assembler-program kan anropas från Extended Basic. Det läser klockan två gånger och väljer endast det andra värdet om närmast lägre tidsenhet är noll. På detta sätt undanröjes osäkerheten i avläsningen.

```
********************************
* Read P-GRAM Clock TWO times
* Convert BCD to decimal
* Compare and select reading
* for XB:
* CALL LINK("CLK",H,M,S,HUN,TUS)
* Jan Alexandersson, Sweden
* 1993-01-31
********************************

       DEF  CLK

CIF    EQU  >0020
FAC    EQU  >834A
STATUS EQU  >837C
NUMASG EQU  >2008
XMLLNK EQU  >2018
GPLWS  EQU  >83E0

WS     BSS  >20
BUF1   BSS  8
BUF2   BSS  8
BUF3   BSS  8

CLK    LWPI WS       load workspace

****** Initial values for read
       LI   R4,BUF1  select buffer
       LI   R0,2     read 2 times
LOOP2  LI   R3,>8648 hour address
       LI   R1,5     read 5 values
****** Read clock value
LOOP1  CLR  R2
       MOVB *R3,R2
       SWPB R2
****** Convert BCD to decimal
       MOV  R2,R6
       SRL  R6,4     divide by 16
       LI   R5,6
       MPY  R5,R6    multiply by 6
       S    R7,R2    subtract
****** Store value in buffer
       SWPB R2
       MOVB R2,*R4+
       DECT R3       new clock addr.
       DEC  R1       next value
       JNE  LOOP1
****** Read second values
       LI   R4,BUF2  select buffer
       DEC  R0
       JNE  LOOP2

****** Compare the 2 readings
       LI   R1,4
       LI   R3,BUF1   compare start
       LI   R4,BUF1   read start
       LI   R5,BUF2   read start
       LI   R6,BUF3   write start
LOOP3  INC  R3
       MOVB *R3,*R3  test if zero
       JNE  NEXT
       MOVB *R5+,*R6+ store value
       INC  R4
       JMP  TEST
NEXT   MOVB *R4+,*R6+ store value
       INC  R5
TEST   DEC  R1
       JNE  LOOP3
       MOVB *R4,*R6   store 1/10000

****** Output values to Basic
       LI   R1,1      first variable
       LI   R6,BUF3   select buffer
LOOP4  CLR  R2
       MOVB *R6+,R2   read value
       SWPB R2
       MOV  R2,@FAC
       BLWP @XMLLNK
       DATA CIF     integer/floating
       CLR  R0        not an array
       BLWP @NUMASG value to Basic
       INC  R1        next variable
       CI   R1,6
       JNE  LOOP4

****** Return to Extended Basic
       CLR  R1
       MOVB R1,@STATUS
```

```
       LWPI GPLWS
       RT

       END
```

```
100 REM READ PGRAM CLOCK3
110 REM TIME DIFFERENCE
120 REM LINK WITH COMPARE
150 CALL INIT
160 CALL LOAD("DSK2.PG-CLK3/O")
170 CALL LINK("CLK",HOUR1,M1
,S1,H1,T1)
180 CALL LINK("CLK",HOUR2,M2
,S2,H2,T2)
190 PRINT STR$(S1);TAB(4);ST
R$(H1);TAB(7);STR$(T1);
200 PRINT TAB(12);STR$(S2);T
AB(15);STR$(H2);TAB(18);STR$(T2);
210 PRINT TAB(22);HOUR2*60*6
0+M2*60+S2+H2/100+T2/10000-H
OUR1*60*60-M1*60-S1-H1/100-T
1/10000
220 PRINT
230 GOTO 170
```

P-GRAM+ MED 4 GRAM-SIDOR

P-GRAM+ kan ha 4 sidor (pages) med GRAM. Jag har själv:

Sida 1 Super-XB på R1-R2 + G3-G7
Sida 2 Personal Record Keeping G3-G6
+ Funnelweb G7 + GMENU G3
Sida 3 Multimod G3-G6 (E/A + DM III
+ TIWR) + RAGDBAE G7
Sida 4 Multiplan G3-G7

Personal Record Keeping kan inte laddas som modul beroende på att Basic-program tydligen måste köras från GRAM-sida 1. Det är dock möjligt att starta Basic och sedan anropa CALL från både Editor/Assembler och Personal Record Keeping i samma Basic-program. Alla mina egna Basic-program för PRK kan köras utan problem.

Funnelweb är endast en laddare för FW-filen som jag har på Horizon RAM-disk vilket gör det lätt att ändra inställningarna.

De fyra GRAM-sidorna använder olika adresser för att läsa data (GRMRD) och skriva adress (GRMWA):

Sida	GRMRD	GRMWA
1	>9800	>9C02
2	>9804	>9C06
3	>9808	>9C0A
4	>980C	>9C0E

REFERENSER

PB 85-4 Maximem
PB 86-3 Maximem
PB 87-1 GRAM Kracker
PB 88-4 Bud Mills: P-GRAM
PB 89-2 Multimod, John Guion
PB 89-3 GRAM-patch för PRK
PB 90-2 RAG Multiplan 4.0 för GRAM
PB 90-4 RAG Multiplan GRAM-bankar
PB 91-4 RAG Linker: GRAM flag words
PB 92-2 P-GRAM PLUS with clock
PB 92-2 Chicago: OPA POP-CART
PB 92-3 Mera om P-GRAM PLUS kortet
PB 92-3 OPA SOB (GROM 0 och 1)
PB 92-4 P-GRAM+ kortet
PB 93-2 Multiplan versioner

Micropendium:

Jul 85 p.22 Programming SUPER CART
Sep 85 p.12 GRAM Kracker to debut
Oct 85 p.14 Firm markets MAXIMEM
Nov 85 p.14 GRAM Kracker
May 86 p.39 GRAM Kracker, review
Jun 86 p.39 Maximem, review
Aug 86 p.38 GPL Assembler, GPL Linker, TI Intern
Sep 86 p.26 GPL tutorial, McCormick
Sep 86 p.38 GRAM Karte, review
Sep 86 p.44 Add DM1000 to GRAM
Nov 86 p.26 General GRAM/RAM loader
Dec 86 p.31 Halt for GRAM Kracker
Dec 86 p.34 GRAM Packer, GK Utility
Jan 87 p.22 GPL Access peripherals
Jan 88 p.34 Gramulator designed
Jan 88 p.38 GRAM Kracker batteries
May 88 p.33 Gramulator shipment
Aug 88 p.40 Gramulator, review
Sep 88 p.38 Horizon releases P-GRAM
Dec 88 p.38 P-GRAM card, review
Sep 89 p.43 Super-XB and GK
Apr 90 p.24 GRAM Boxes
May 90 p.40 P-GRAM PLUS card
Jun 90 p. 8 GRAM devices still out
Apr 91 p.28 Rich GKXB available
Jul 91 p.29 Multiplan 4.0 patches
May 92 p.25 OS/99 version 3, review
Sep 92 p.29 Rich RXB Extended Basic
Dec 92 p.21 Rich RXB and Gramulator
Dec 92 p.27 Compare XB-AL-GPL
Feb 93 p.06 OS 99's extendability
Feb 93 p.25 OS 99 V4 released

BITS,BYTES&PIXELS(Lima Ohio UG)

Dec 91 Chicago Fair: OPA POP-CART
Jan 92 The P-GRAM Plus with clock
Jan 92 PGRAM Clock utility software
Jan 92 Load File 0 1 1 CLOCK
Apr 92 GRAM User Menu System (GUMS)
Apr 92 Souped up supercarts ■

MULTIPLAN VERSIONER

av Jan Alexandersson

Microsoft Multiplan v 1.04 tillhör de originalmoduler PHM 3113 som TI tog fram 1982. Den levererades med en modul, en flexskiva och en manual på 240 sidor i en stor ringpärm. Multiplan är ett kalkylprogram som har ett stort antal (255 rader, 63 kolumner) rutor (celler) där du kan mata in tal, text eller formler. Cellerna påverkar varandra med formler som du själv matar in. Multiplan kan liknas vid ett programmeringsspråk där in- och utdata alltid sker via cellerna.

Multiplan har tagits fram i flera förbättrade versioner på flexskiva. Du måste dock alltid ha originalmodulen eftersom skivorna endast har ändringar:

- TI original-version (med modulen)

- TI:s förbättrade version (finns i programbanken tillsammans med ny TI-Writer). Se PB 85-1.

- RAG Standard-version 4.00 (finns i programbanken)

- RAG GRAM-version 4.00 (finns i programbanken)

- 80-kolumnsversion för videoprocessorn 9938 eller 9958

Systemskivan TIMP laddas med DSK.TIMP.FILNAMN från lägsta DSK på lägsta CRU-nummer som normalt är DSK1. Det fungerar dåligt att ladda från DSK2 eftersom laddaren söker runt bland skivenheterna flera gånger under varje laddning.

Den nya versionen som beskrevs i PB 85-1 är snabbare och dessutom har Anders Persson ordnat en modifierad teckenfil SW-MPCHAR med svenska bokstäver som ska döpas om till MPCHAR på den system-diskett du använder.

RAG Standard-version är 50 % snabbare än ovanstående förbättrade version. Den kan även permanent ändra default för DSK, printer och skärmfärger. Filerna kan laddas på normalt sätt via DSK.TIMP.FILNAMN av modulen. Det är även möjligt att ladda Multiplan via filen MPLOAD från hårddisk eller RAM-disk. Modulen måste givetvis finnas på plats även i detta fall. På detta sätt kan du även ladda Multiplan från DSK2 eller DSK3.

RAG GRAM-version fungerar på samma sätt som RAG Standard under körningen men vissa filer (MPINTR, MPCHAR, OVERLAY) lagras i GRAM och modul-RAM så att de kan laddas mycket snabbt. Om du även har GRAM 1-2 så kan MPBASE och MPDATA lagras där. Med P-GRAM som endast har GRAM 3-7 så måste du således ändå ha MPBASE och MPDATA på en systemskiva. Det verkar inte finnas så stora skäl att använda GRAM-versionen när det inte heller går att ladda in Anders Perssons SW-MPCHAR (den finns ju i GRAM från början).

80-kolumnsversionen laddas via modulen på samma sätt som originalversionen. Multiplan med 80 kolumner är det program som har mest att vinna på 80 kolumner. Katalog av flexskiva kan visas i fyra spalter med vardera 14 rader. Katalog visar en del skräptecken som tycks bero på någon läsbuffert nu när den synliga skärmen har dubbelt så stort RAM-minne. Sänd en skiva med returporto till mig så ordnar jag en kopia.

Du kan ändra den skivenhet som Transfer Load och Save använder med (O)ptions, TAB(CTRL-2) och Setup till t.ex. DSK2.

UTSKRIFT FRÅN MULTIPLAN

Du kan ordna utskrift till valfri utport t.ex. RS232 eller PIO på följande sätt (se manualen sid 14):

- välj (P)rint

- välj (O)ptions

- area: R1:255 (default, kan ändras)

- TAB(CTRL-2)

- setup: PIO(eller RS232.softswitch)

- välj (M)argins

- print length: 58 (60 med förlängd)

- page length: 70 (72 med förlängd)

ATT ÄNDRA SKÄRMFÄRGERNA

Du kan ändra skärmfärgerna innan system-filerna laddas. Tryck på mellanslagstangenten minst två gånger. Upprepade tryckningar stegar igenom ett antal olika färg-kombinationer. Detta fungerar med båda TI-versionerna och 80-kolumns-version. RAG-versionen ställer självt in skärmfärgen som över-trumfar ovanstående tryckning av mellanslag.

MULTIPLAN MED RAMDISK

Multiplan kan alltid oberoende av version laddas från RAM-disk om denna placeras på CRU >1000. RAM-disken måste även döpas till TIMP.

Med hjälp av RAG-versionens MPLOAD kan du även ladda från höga CRU-adresser och valfria skivnamn. SAVE och LOAD av filer på höga CRU-adresser fungerar endast med RAG-versionen.

MULTIPLAN MED MYARC HFDC

Du kan ladda system-filerna, oberoende av version, från en subdirectory WDS1.DSK.TIMP om kortet finns på CRU >1000 eller >1100. På CRU >1000 kan system-skivan även laddas från den första flexskiveenheten DSK5 om skivan heter TIMP. Med RAG-versionens MPLOAD kan system-filerna placeras på valfri sub-directory eller DSK.

Problem med HFDC på CRU >1000:

- Katalog från DSK 1-3(TI) och DSK 5-8(Myarc) fungerar ej

- SAVE och LOAD från DSK 1-3(TI) och RAM-disk(DSK 4 och 9) fungerar ej

Problem med HFDC på CRU >1800:

- Katalog från DSK 5-8(Myarc) fungerar ej

- SAVE till RAM-disk(DSK4 och 9) fungerar ej

RAG-versionen med HFDC på CRU >1800 fungerar alltid med SAVE och LOAD. Katalog fungerar utom för DSK 5-8.

Jag kan inte ladda Myarcs DM V på något sätt med CRU >1800 så jag måste nog sätta tillbaka den till CRU >1000 men då fungerar Multiplan sämre. Inte ens RAG-versionen klarar då SAVE och LOAD från DSK 1-3 (TI).

MULTIPLAN MED P-GRAM+

P-GRAM+ har 4x40 kbytes GRAM. GRAM-sida 1 (page 1) fungerar som själva modulen. Det naturliga är att ha Extended Basic på GRAM-sida 1 så att Multiplan måste placeras på GRAM-sida 2-4. Det finns en speciell patch-fil (PATCH_MP) som ändrar absoluta adresser i MPINTR-filen. För GRAM-sida 2 ändras >9800 (GRMRD) till >9804 och >9C02 (GRMWA) till >9C06.

RAG-versionen fungerar även med GRAM-sida 2 om PATCH_MP har använts men endast vid laddning via modulen. MPLOAD för laddning från hårddisk/ RAM-disk fungerar inte på GRAM-sida 2. Jag får konstiga ljud så att jag måste stänga av Multiplan.

REFERENSER

NN 83-3.21 Multiplan
NN 83-4.21 Mer om Multiplan
PB 85-1.20 Multiplan upgrade
PB 90-2.03 Multiplan v 4.00 RAG
PB 90-4.25 Multiplan GRAM
PB 90-5.23 Multiplan SYLK files - 1
PB 90-6.03 DSK.SKIVA.FILNAMN
PB 90-6.12 Multiplan SYLK files - 2
PB 91-4.04 MP DIS/VAR 80 files with record length >80 (BITS)
PB 92-1.04 Multiplan application paying your bills (BITS)
PB 92-3.21 SYLK files - 2, rättelse
PB 93-2.15 Multiplan Tips (HTA) ■

FROM BASIC TO ASSEMBLY - 3

by Bob August, Bug News, USA

This month we added two more routines to our program. You can take last months source code and just update it with the new procedures. We also used the CALL KEY(3,K,S) as promised. We now have four routines.

1. Key scan routine to accept a keyboard entry.

2. Clear screen routine to clear the screen.

3. Screen display routine to display a message on the screen.

4. Beep tone routine to give you a beep tone.

In this months program we set up a buffer to save the return address in. We need this when we start jumping around in our program so this month we added it in the program. We also added the MOV R11, @SAV11 at the start of our program. Register 11 is the counting register and always contains the current address of your program. When we start it contains the start address which we save in the SAV11 buffer. Before exiting our program we move this address back to register 11 and the exit. We also used a GOSUB to clear the screen, a GOSUB to write to the screen and a GOSUB for the beep tone. The eqivalent in Basic would be:

```
100 REM Basic to assembly 3
110 REM Print message one
120 GOSUB 290
130 GOSUB 320
140 GOTO 190
150 REM Print message two
160 GOSUB 290
170 GOSUB 340
180 REM Print message three
190 GOSUB 360
200 GOSUB 390
210 REM Call Key routine
220 CALL KEY(3,K,S)
230 IF S=0 THEN 220
240 IF K=65 THEN 120
250 IF K=66 THEN 160
260 IF K=13 THEN 270 ELSE 220
270 STOP
280 REM Clear screen routine
290 CALL CLEAR
300 RETURN
310 REM Print routines
320 PRINT TAB(4);"THIS IS ME
SSAGE NUMBER ONE"::::::::::::
330 RETURN
340 PRINT TAB(4);"THIS IS ME
SSAGE NUMBER TWO"::::::
350 RETURN
360 PRINT "PRESS: A For Mess
age # one":TAB(8);"B for Mes
sage # two"::TAB(8);"Enter k
ey to quit"
370 RETURN
380 REM Beep tone routine
390 CALL SOUND(150,1500,0)
400 RETURN
410 END
```

This month we used one new command that we didn't use last month.

MOVB = Move byte form here to there.

Our call key routine has been changed to add 3 so the keyboard will accept both lower and upper case as upper case. For a call key zero you would place byte 00 at >8374. You can put a 00, 01, 02, 03, 04 or 05 at >8374 depending on what key routine you want.

In our display routine we used a new comand MOV *R11+,R0 which needs some explaining. Remember R11 has the current address of your program. The asterisk in front of the R11 means move the data at the address in register 11 into register zero. This is called indirect addressing. The plus sign after the R11 means increment register 11 by 2. This then makes register 11 point to the address of the next data item. The beep tone is straight foreward and the remarks explain it. The only thing you should note is that we had to reference SOUND at the start of our program.

HAPPY ASSEMBLING!

```
*************************************
* BASIC TO ASSEMBLY  Lesson Number 3 *
*************************************
*
       DEF  START          Entry point label
*
       REF  VSBW,VMBW          Utilities
       REF  SOUND,KSCAN        Utilities
*
WRKSP  BSS  32             Allocate workspace
SAV11  BSS  2              Buffer for return address
* 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 its even
*
START  MOV  R11,@SAV11     Save return address
       LWPI WRKSP          Load workspace
* Print message one - DISPLAY AT(8,4)
M1     BL   @CLEAR         GOSUB CLEAR
       BL   @DISPLY        GOSUB DISPLY
       DATA 227,MSG1,26    Location,Message,Length
       B    @M3            GOTO M3
* Print message two - DISPLAY AT(14,4)
M2     BL   @CLEAR         GOSUB CLEAR
       BL   @DISPLY        GOSUB DISPLY
       DATA 419,MSG2,26    Location,Message,Length
* Print message three - DISPLAY AT(20,1)
M3     BL   @DISPLY        GOSUB DISPLY
       DATA 608,MSG3,120 Location,Message,Length
       BL   @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
       CLR  @>837C         Clear status
       MOV  @SAV11,R11     Get return address
       BLWP @0             FCTN Quit
* Clear screen routine - CALL CLEAR
CLEAR  LI   R1,>2000       R1=32
       CLR  R0             Make R0 zero
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+,R0     Message location in VDP
       MOV  *R11+,R1     Message to write
       MOV  *R11+,R2     Length of message
       BLWP @VMBW        Put it on the screen
       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 with auto start
       END  START                                          ■
```

PROGRAMS WRITE PROGRAMS -1

by Jim Peterson, Tigercub, USA

Way back in 1982, in the old 99'er Magazine, Vol. 1 Nos. 3 and 4, John Clulow wrote two articles entitled "How To Write a Basic Program That Writes Basic Programs". At that time I thought I would never understand what he was writing about!

But really, it's simple. Have you ever LISTed a program to the disk, and noticed that the resulting D/V80 file took up many more sectors than the program itself? That is because the TI saves programs in a compacted form, with each statement represented by a single token ASCII.

When a program is saved in MERGE format, by SAVE DSK (filename),MERGE it is saved in this same compacted form, but in a D/V 163 file. And of course a D/V file can be created by a program - so you can write a program which will create a D/V 163 file in the form of a program, and then MERGE that file into memory and RUN it as a program, and SAVE it as a program.

You ask, why use this roundabout way of writing a program? Why not just key it in directly? Well, for one thing you can write program lines that could not possibly be keyed in directly. As for instance, the famous "line zero". Key this in, run it with a disk in drive 1, then enter MERGE DSK1.ZERO and LIST the result.

```
100 M$="BETCHA CAN'T DELETE
THIS!"
110 OPEN #1:"DSK1.ZERO",VARI
ABLE 163,OUTPUT :: PRINT #1:
CHR$(0)&CHR$(0)&CHR$(131)&CH
R$(200)&CHR$(LEN(M$))&M$&CHR
$(0)
120 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 :: END
```

Actually, there is an easy way to delete that line - but no way to key it in directly.

Here's another one - the full ASCII string.

```
100 OPEN #1:"DSK1.FULLSTRING
",VARIABLE 163,OUTPUT
110 LN=100 :: GOSUB 190 :: A
$=L$&"M$"&CHR$(190)
120 FOR J=1 TO 127 :: C$=C$&
CHR$(J):: NEXT J :: A$=A$&CH
R$(199)&CHR$(127)&C$&CHR$(0)
130 PRINT #1:A$
140 GOSUB 190 :: B$=L$&"M2$"
&CHR$(190)
150 FOR J=128 TO 255 :: D$=D
$&CHR$(J):: NEXT J :: B$=B$&
CHR$(199)&CHR$(128)&D$&CHR$(
0)
160 PRINT #1:B$
170 GOSUB 190 :: F$=L$&"M$"&
CHR$(190)&"M$"&CHR$(184)&"M2
$"&CHR$(0)
180 PRINT #1:F$ :: PRINT #1:
CHR$(255)&CHR$(255):: CLOSE
```

```
#1 :: END
190 L$=CHR$(INT(LN/256))&CHR
$(LN-256*INT(LN/256)):: LN=L
N+10 :: RETURN
```

Run that, then enter NEW, then MERGE DSK1.FULLSTRING. The string contains every ASCII from 0 to 255 in sequence, and there is no way to enter many of the unprintable ASCII codes from the keyboard. You can of course create that string in a program -

```
FOR J=0 TO 255 :: M$=M$&CHR$
(J):: NEXT J
```

but it saves a few seconds to have it predefined.

The full ASCII string is very useful for a quick shuffle without duplication. For instance, to scramble the numbers 200-250, -

```
100 M$="
          !""#$%&'()*+,-./
0123456789:;<=>?@ABCDEFGHIJK
LMNOPQRSTUVWXYZ[\]^_`abcdefg
hijklmnopqrstuvwxyz{|}~ "
110 M2$="
                              "
120 M$=M$&M2$
130 M$=SEG$(M$,200,50)
140 L=LEN(M$):: RANDOMIZE ::
 X=INT(L*RND+1):: N=ASC(SEG$
(M$,X,1)):: M$=SEG$(M$,1,X-1
)&SEG$(M$,X+1,255)
150 PRINT N;:: IF LEN(M$)=0
THEN STOP ELSE 140
```

One more example - can you run this program and get these results? You won't even be able to key in that last line!

```
>LIST
100 FOR J=1 TO 7 :: READ M$
:: PRINT M$ :: NEXT J
30000 DATA AAAAAAAAAAAAAAAAA
AAAAAAAAAAA,BBBBBBBBBBBBB,BB
BBBBBBBBBBB,CCCCCCCCCCCCC,
           DDDDDDDDDDDDDDD
30010 DATA "TESTING",,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,""TEST
ING""
>RUN
AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBB,BBBBBBBBBBBBBB
CCCCCCCCCCCCC
             DDDDDDDDDDDDDD
"TESTING"
,,,,,,,,,,,,,,,,,,,,,,,,,,,,
""TESTING""
```

■

HTA – MULTIPLAN TIPS

from Bill Sponchia, Canada

1. To speed up Multiplan there are two ways:

i) turn the "recalculation" feature off. Go into "Options" and set "Recalc" to "NO". This should be done not only when inputting data but also when assembling the sheet (setting up the formulae, etc.) To calculate the sheet then press "FCTN 8"

Warning: If the Recalc is turned off then the computer will automatically do a recalculation when you save the sheet; however if asked to print the sheet the computer WILL NOT recalculate first thus the printed sheet will be wrong. A good habit to get into is to always do a recalculation (FCTN 8) before printing.

ii) keep the spreadsheet as small as possible. Although some sheets must be large most can be divided up into smaller sheets. For example if you have one large sheet showing "Revenue" and "Expenses" then you should consider making two small sheets; one for Revenues and one for Expenses. The extra time spent setting up two sheets will be more than offset by the savings in recalculation time.

2. A spreadsheet can be Printed to disk which can be loaded into the TI-Writer for merging with another document, etc.

WARNING - be careful to use a different file name than the one used to save the Multiplan file as Multiplan will not warn you if you are about to overwrite an existing file when printing to disk.

If you set the margins greater than 80 Multiplan will write the DV80 file wherein each record is longer than 80 characters. If you attempt to read this file with a BASIC program your system will produce a strange error code and lock up.

TI-Writer will read the file but will only input the first 80 characters.

3. If you have Names that you wish to eliminate (un-name) this can be done by going into Name and deleting the reference in the "to refer to" field.

4. If you have "linked" the spreadsheet to another to eliminate this link all you must do is to erase the "Name" field. With a link (or reference) to nothing the link is broken and deleted.

5. To change the target of an Xternal copy just specify the new cell (or cells) in the "to:" field. Since each cell (or range) on the supporting sheet can have only one target (on the dependent sheet), the old link will be replaced by the new.

6. If your RECALC is OFF, you can recalculate a single cell by setting the pointer to the cell, press "E" for Edit, then press ENTER. Only that cell will be recalculated.

Note: If that cell relies on another which also had to be calculated then the information used from that other cell may be uncalculated and therefore the cell you just recalculated will be wrong.

7. To increase the response time of MP you should file copy the following files in the following order: OVERLAY, MPHLP, MPCHAR, MPDATA, MPINTR and then MPBASE

8. When you become familiar with MP you can eliminate the MPHLP (Help) file from your work disk and there by free 158 sectors.

9. If you choose to lock the formulas in a worksheet make an unlocked backup "BEFORE YOU LOCK THEM". Locking the formulas is easy; however unlocking requires you to do it piece by piece.

10. If someone else will be doing data entry on a complex worksheet, it is a good idea to have them working with a locked copy. This avoids problems such as having someone enter data into a cell containing formulas or information you use elsewhere.

11. Use relative references wherever possible. This allows for copying of formulas without editing. Editing of formulas is both time consuming and prone to error.

12. References in formulas should be done by "pointing". This method is simple, creates relative references and is subject to less errors.

13. Once you start scrolling, you can release the FCTN or CTRL key and just keep the ARROW key depressed.

14. On a data disk with more than 18 files you can catalog (display filenames) the additional files (remember the SD command only shows the first 18 files) by placing the cell pointer on the last filename then pressing REDO (FCTN 8). After the screen has been redrawn and displays "TRANSFER LOAD filename" (where filename is the one you have just highlighted) the message line will now display "Enter a filename (arrow for directory)". Now press FCTN down-arrow and the last filename from the previous screen plus the additional files will be displayed.

15. If there is a rectangular area that will be used frequently on your worksheet, consider giving it a Name. You may then refer it by this name and thus will speed up the moving around the worksheet.

16. Do you have one large file and wish to make it into two or more smaller parts? It's easy, first you give a name to the section you wish to move to the second sheet and then save the large file to disk. Then load in the blank template and make use of the "external" copy command. Make sure you tell it "no" to linking.

■

KEYBOARD ECHO - TI-99/4A

by Bob August, Bug News, USA

This will help you learn the keyboard so you can improve your touch typing skills, in Basic and XB.

BASIC VERSION

```
100 REM KEYBOARD ECHO
110 REM IN TI BASIC BY
120 REM R.W. AUGUST
130 CALL CLEAR
140 CALL SCREEN(8)
150 REM  SOMETHING FOR ENTER
160 CALL CHAR(141,"FF00FF00FF00FF")
170 REM MAKE LEFT ARROW
180 CALL CHAR(136,"002060FF602")
190 REM MAKE RIGHT ARROW
200 CALL CHAR(137,"000406FF0604")
210 REM MAKE DOWN ARROW
220 CALL CHAR(138,"10101010107C381")
230 REM MAKE UP ARROW
240 CALL CHAR(139,"10387C101010101")
250 PRINT " LEARNING THE T.I
. KEYBOARD":::::"PRESS ANY K
EY WHILE WATCHING"::"THE SCR
EEN, NOT THE KEYBOARD"
260 PRINT ::"  PRESS FUNCTIO
N 4 TO QUIT":::::"KEY PRESSE
D WAS: ":::::
270 REM WAIT FOR KEY PRESS
280 CALL KEY(4,K,S)
290 IF S=0 THEN 280
300 IF K=13 THEN 310 ELSE 320
310 K=141
320 REM PUT IT ON SCREEN
330 CALL HCHAR(19,20,K)
340 GOTO 280
350 END
```

EXTENDED BASIC VERSION

```
100 ! KEYBOARD ECHO
110 ! IN TI EXTENDED BASIC
120 ! BY R.W. AUGUST
130 CALL CHAR(136,"002060FF6
02")! MAKE LEFT ARROW
140 CALL CHAR(137,"000406FF0
604")! MAKE RIGHT ARROW
150 CALL CHAR(138,"101010101
07C381")! MAKE DOWN ARROW
160 CALL CHAR(139,"10387C101
010101")! MAKE UP ARROW
170 DISPLAY AT(3,2)ERASE ALL
:"LEARNING THE T.I. KEYBOARD
": : : : :"PRESS ANY KEY WHI
LE WATCHING": :"THE SCREEN,
NOT THE KEYBOARD"
180 DISPLAY AT(13,3):"PRESS
FUNCTION 4 TO QUIT": : : : :
"KEY PRESSED WAS: ";K$
190 CALL KEY(4,K,S):: IF S=0
 THEN 190 :: K$=CHR$(K)
200 IF K=13 THEN K$="ENTER"
210 GOTO 180 :: END
```

In the Basic version of the program we used the value of K, from the CALL KEY statement, in the HCHAR to put the key press on the screen. In the Extended Basic version we changed the value of K to a string K$. We did this with the statement K$= CHR$(K). This changed the value 65 to the ASCII A. The reverse of this would be K=ASC(A). This takes the first character of a string and changes it to a numeric value. Another statement which works similar is the K$=STR$(K). This changes a numeric value to a string value. If K was equal to 65, K$ would still be equal to 65, but as a string and not a numeric value. Here is a short program that may help you understand what these three functions do.

```
100 CALL CLEAR
110 K=65
120 PRINT "K =";K
130 K$=CHR$(K)
140 PRINT :"K$=CHR$(K) = ";K$
150 A=ASC(K$)
160 PRINT :"A=ASC(K$) =";A
170 B$=STR$(K)
180 PRINT :"B$=STR$(K) = ";B$
190 END
```

Run the program and you will see that the first statement returns an A. The second statement returns a 65. And the third statement returns a 65. If we added two lines.

```
185 C=K+K
186 PRINT :"C=K+K =";C
```

you would get a total of 65 plus 65 or 130. If you tried to add B$ and B$ you would get an error because B$ is a string even though it has a value of 65.

These three statements are used quite a bit in programming and you should play with them until you know what they do and how to use them.

HAPPY PROGRAMMING! ■

SWEDLOW TI BITS * 26-27 *

by Jim Swedlow, USA

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

EDITOR ASSEMBLER (EA) MODULE

Do you own the EA module? Probably not. Most of us have XB but not EA. With the price down to $10, it is a good buy. Many fine programs run with the EA module including FUNNEL-WEB, ARCHIVER and DM1000. All of these programs will run under XB but they load much faster with EA.

Next time you see EA on sale, pick up a copy. Take the book and put it on a shelf somewhere. Put the disks in your master box. You won't need either unless you start programming in assembly.

You will wonder how you got along without EA.

LOADING UTIL1 FILES IN EA

When you insert the EA module and "PRESS ANY KEY", you have two choices:

1 FOR TI BASIC
2 FOR EDITOR ASSEMBLER

Press <2> and you get FIVE choices. Only two are of use for loading programs:

3 Load and Run
5 Run Program File

"Load and Run" is for object code (Display Fixed 80) files while "Run Program File" is for EA Program files.

If you press 5, EA will ask you for a file name. If you just press <ENTER> without entering anything, EA will look for and try and load a file named <UTIL1> in Drive 1 (close to the XB autoload of LOAD but you have to press two more keys). That is why FUNNELWEB has a UTIL1 and a LOAD program - one for XB and one for EA.

YOUR DISK WON'T WORK

Every now and then you get errors when you try to read from a disk. This can mean any of many things. Your drive might be dirty. You might have the wrong disk. The disk could be blown. The file could be lost. The list of horrors is long.

Here is a simple solution that you might try first.

Remove the disk from the drive. Holding it by a corner, put your fingers in the center hole and turn the disk in its sleeve a few turns. Then move the media (the disk) a-round a little. Touch only the corner and the center.

Sometimes this will restore your disk.

BAD DISKS

I purchased a box of disks at a dis-count house (low prices and no ser-vice!). One of the disks had bad sectors when I formatted it. I wondered what would happen, so I sent the disk back to the manu-facturer asking for a replacement.

I was out about $1 (between postage and mailer) but it was worth the try. Over a month later another disk came in the mail along with a form letter explaining about quality control and such.

The only problem was that the new disk also had bad sectors. That one went into the trash and I won't buy that brand again! Who was it? I really shouldn't say, but they are known for their film and there aren't any K's in their name.

FUNNELWEB VERSION 4.13

The McGoverns released version 4.13 in December, 1988. The major changes are a fix to DM1000, a mod in QD, 80 column support and some

minor bug fixes in UTIL1 and LOAD.

DM1000 (through early releases of Version 4.12) would not work on a one drive system and lost the copy/ delete sector count when moving from one screen of files to another. These have been fixed.

Eighty column support works with the DIJIT and similar cards. A separate fix is available that includes an 80 column ROS (RAMdisk Operating System).

Quick Directory (QD) and Show Directory now support CTRL E and CTRL X for moving between displayed pages of files. The earlier CTRL/SHIFT paging system was removed due to compatibility problems with the Geneve.

EIGHTY COLUMN CARDS

In his note on Version 4.13, Tony McGovern said:
"The main new TI event around here has been the arrival of an evaluation model of the DIJIT AVPC card a few weeks ago. I'm really impressed with this item and I suspect most of my program writing in future will be directed towards exploiting this device. I've already done a preliminary 80 column version of the FWb Editor (it should run on the Geneve also)."

GRAPH

The other night my 13 year old just had to have graph paper for her home work. Never mind that she knew that she would need some paper for some time; she had forgotten to tell us and the need was urgent.

I remembered that I had an XB program that printed graph paper. To my surprise, I found it fairly quickly. I ran it and it worked - almost. It printed graph paper but the boxes were wider than they were long. It sufficed for her assignment but I had to fix the program.

Here it is:

```
100 ! GRAPH
110 E$=CHR$(27)
120 A$=RPT$(CHR$(128),228)
130 B$=RPT$(CHR$(255)&SEG$(A$,1,6),
    8)
140 B$=RPT$(B$&CHR$(255),4)
150 A$=E$&"K"&CHR$(228)&CHR$(0)&A$
160 B$=E$&"K"&CHR$(228)&CHR$(0)&B$
170 OPEN #1:"PIO.CR"
180 FOR I=1 TO 11
190 PRINT #1:E$;"@";E$;"3";CHR$(16)
200 FOR J=1 TO 8
210 PRINT #1:B$;B$;CHR$(10)
220 NEXT J
230 PRINT #1:A$;A$;E$;"3";CHR$(2)
240 NEXT I
250 PRINT #1:RPT$(CHR$(13)&CHR$(10)
    ,9)
260 PRINT #1:E$;"@"
270 CLOSE #1
```

This program will work with MOST Gemini and Epson compatible printers. There are two printer commands that can cause you problems.

The first one is <E$;"@"> (remember that E$ is defined in line 110 as CHR$(27) or Escape) which appears in lines 190 and 260. This is a reset command that tells your printer to restore its default settings. Earlier Epson compatibles (including the TI Impact Printer) do not recognize this command. If you get garbage with some "@" characters, yours doesn't either. These embedded reset commands cause your printer to completely loose any idea of where the top of form is so you will have to manually reset it.

The printer command that caused my problem is <E$;"3";CHR$(16)> in line 190. For the Gemini 10X and most Epson MX compatible printers, this sets line height to 16/144 (or .1111) inch. For the Star NX10, NX1000, and most Epson FX compatibles, it sets the line height to 16/216 (or .0741) inch. Hence the squat squares. I changed the line to read <E$;"3";CHR$(24)>. This set line height at 24/216 (or .1111) inch and everything worked correctly.

The difference for the <E$;"3"; CHR$(2)> in line 230 is so small (.0046 inch) that it makes no difference. If you really wanted to, you could change it to <E$;"3"; CHR$(3)>.

OLD TRICKS FOR NEW USERS

by Joseph Cohen, Lima Ohio User Group, USA

Thou many of us tend to ignore most of the cartridge software for our computer, with the exception of TI Extended Basic and, possibly, Multiplan, Logo II, Editor/Assembler, and TE-2 (for speech), many of the cartridges are very enjoyable. In order to give you an excuse for searching your closets and basements looking for those hidden modules, I'd like to point out that many of them have undocumented features ranging from useful to interesting to amusing. Here are a few examples.

Many are probably familiar with "The Secret of Personal Record Keeping: Implementing DISPLAY AT and ACCEPT AT without Extended BASIC", published way back in 99'er magazine and reprinted in The Best of 99er, p. 76. Briefly, TI BASIC with the PRK module contains the commands CALL D() and CALL A() (similar to DISPLAY AT and ACCEPT AT). Presumably this also works with the Statistics module, but I do not have this one and could not verify it. I have been told that this is a result of the hybrid nature of certain modules, containing both GPL and BASIC coding.

Next, the TI Disk Manager cartridge offers a proprietary protection feature that does not allow the Disk Manager to copy a protected diskette. To use it, press the FCTN-X key ten times while on any menu screen. You will hear a beep (if your monitor has sound!) and >< will appear at the center top of the screen. Any diskettes initialized at this point will be proprietary protected. Each time you address them using the Disk Manager cartridge (e.g., to catalog such a diskette), a low-tone beep will sound (not present for unprotected diskettes), informing you that the diskette is protected. The protection information is stored in sector 0 on the diskette. This type of protection is ineffective against the sector disk copiers and has been discussed in the past. I wonder of anybody knows more about it. The DM-1000 offers protection and unprotection of diskettes; is it the same kind as the TI Disk Mangager cartridge?

Now to a few game cartridges. Moonmine, Alpiner, Munchman, Munchmobile, and Hooper have a test mode, where you can select the starting level. So if you wanted to see what it is like to play at those levels you could never reach, here is a good reason to plug those cartirdges into your 99/4A! The test mode is obrained by pressing SHIFT/8 3 8 at the game title screen (SHIFT/8 only, for Hopper). and on Burgertime, pressing SHIFT/8 gives a message: "code modifications by John M. Phillips". Have you always played Parsec as a one player game? Here is something different, for a two player team. If the fire buttons on both joysticks are pressed simultaneously, Spaceship Parsec will not overheat. Horizontal lines will appear on the screen, but they do not disturb the game and would allow, in fact, one to see the Bynites when they turn invisible. ■

The problem with CHR$(27) "3" CHR$(n) raises its ugly head in a number of programs. Most folks wrote for the Gemini 10X/Epson MX family. This change, which came with the NX/FX lines of printers, has not received wide attention.

If your printout lines are too close together, look for this code and increase "n" by a factor of 1.5. You can, that is, if it is an XB program.

Enjoy. ■

ASGARD SOFTWARE

Ny adress till Asgard är nu:

Asgard Software
1423 Flagship Dr.
WOODBRIDGE, VA 22192
USA ■

CRU POWER-UP AND SEARCH

by Tony McGovern, Funnelweb Farm, Australia (Dec/20/92)

When pondering on continuing HRD-3000 problems I realized that I had never seen a thorough discussion of how RAMdisks and the like fit in with the TI system in so far as CRU base address settings and machine capture on power-up are concerned. Writers of HRD ROSs obviously have faced the problem, but it is worth chatting about in public. During power-up the monitor program in a standard TI-99 system amongst its other housekeeping chores, sequences through the peripheral CRU base addresses and at each one executes the power-up routine if the card has one, and finally executes from GPL any cartridge power-ups in the GROM library. TE-2 snatches some more VDP memory below the disk buffer area, possibly for text to speech work area. The nature of the routines varies from device to device. For instance the TI disk controller spins the drive motors briefly, while Myarc FDCs do not. This is the point at which a device may capture control of the machine from the monitor - all it has to do is usurp the monitor and not hand back control. TI made two well known but not common devices which did capture the machine this way - the p-code card and the the Plato module. Notice that Plato, being a module, was the last little piggy of all at the power-up trough, while p-code at CRU base >1F00 was the last card to be found in the PE-box. Also there was only one possible peripheral master in the system, the p-code which did not care about the cartridge anyway. This meant that every other peripheral box card in the system could do its own initial housekeeping first.

Now with the introduction of Horizon and similar devices with re-programmable DSRs and switchable CRU bases, it is possible to have the user configure a card to capture the machine from any CRU slot, and also hand over to the cartridge at any time later. This implies that for correct operation of the system, the card which captures the machine must take the place of the monitor and execute the power-up routines of all cards normally later in the search, and strictly, resume the monitor power-up sequence at the cartridge rather than just starting the cartridge. Fortunately it doesn't seem to matter for XB which is the only module GROM normally entered from programs loaded at several stages of remove from a HRD power-up capture. The TI system specs allow for power-up to use GPL workspace registers R0-R10. R12 contains the CRU base address and R13-R15 are already set up. The various documents do not all agree on just what PAD RAM is available. The 1983 System Software comprehensive spec says PAD+>4 to PAD+>BF may be used, but does not mention any of the specific bytes to be preserved, when clearly the VDP pointer at PAD+>70 is important information for other DSRs at the very least. Why does this matter? Well, if a card is to execute power-up routines in other cards it must run from code placed outside its own DSR ROM and this must not be killed by other power-ups. Why outside the DSR ROM? Only one DSR can be active at a time - you switch yourself off before switching the other one on. The p-code did not need to do this as all power-ups had already been executed. The method adopted in the Miami ROSs was to assume memory expansion present and tip some code out at >A000 (better for FW compatibility at >A050 so the system workfile name is not trashed). I assume in the absence of published descriptions or source code that the OPA 8,14 ROS does much the same. The assumption is that no other power-up uses the same area. TI never allowed for card capture except by p-code, and so made no provisions for it in general.

Not all the design problems are simply and universally solvable unless the system is restricted to having only one auto-boot card active at a time. I always set up my systems so that the auto-boot HRD (192Kb devices) is at CRU >1600 or

FUNNELWEB v5 TEXT EDITOR

described by Charles Good, Lima Ohio User Group, USA (Feb 1993)

Tony McGovern has released a "completely rewritten from source code" Funnelweb version 5 80 column editor dated Dec 15, 1992. A similar 40 column version will follow soon. I have a beta test edition of this 40 column editor. These v5 editors are designed to run from the Funnelweb v4.4 environment. So far, the only "version 5" parts of Funnelweb are the text editors. They are fully multi-lingual and compare favorably with Asgard's new FIRST DRAFT word processor. New features, added or revised since the v4.4 editor are summarized below.

HELP SCREENS AND FILES IN MEMORY

FIRST text in memory - edit buffer.

SECOND text in mem. - help screens: When the 80 column editor first boots it loads into memory up to four help screens. These can be viewed from the command line by pressing H(elp). Each screen is 26 lines by 80 columns and they pop up on screen immediately because they are already in VDP memory. The Program Editor loads one set of help screens relating to assembly language coding. The Word Processing editor loads another set of screens more appropriate for help with text editing and formatting. You can move back and forth from one help screen to another by pressing the Q and A keys. FCTN/9 returns you back to the edit buffer. Sets of useful help screens are provided, and the user can also create personalized help screens. A utility is provided to convert the first 26 lines of any DV80 file into an 80 column help screen. {The 40 column editor will have 28 line by 40 column help screens. An unlimited number of these screens can be loaded into memory one at a time from disk by pressing H(elp) from the 40 column command line.}

THIRD text in mem. - screen viewing: As in the v4.4 text editor, one can do a S(how) D(irectory), move the cursor next to a DV80 file name, press a key, and display a screen of that file. Subsequent presses of the same key window down through the entire file. There is no limit to the size of the file that can be viewed one screen at a time in this manner. This file isn't really stored in memory, just displayed on screen.

FOURTH text in mem. - V(iew) buffer: From S(how) D(irectory) you can put the cursor next to the name of a DV80 file, press CTRL/V and load the whole file (or any part of it) into a 64K memory buffer for instant recall any time during the editing process. Once loaded into the V(iew) buffer the file can be scrolled one line at a time or windowed up and down very rapidly. Pressing <enter> from within S(how) D(irectory) or V from the editor command line pops this text into view. This V(iew) buffer can hold very large text files. It is in fact the same memory area as Funnelweb's Disk Review "V" text buffer. You can load some text into the 80 column editor's V(iew) buffer, exit the editor to a central menu, and from there go to Disk Review. After performing some disk management functions from Disk Review, you can go back to the 80 column text editor and the V(iew) buffer text will still be there! You can also load ANY KIND OF FILE into Disk Review's V(iew) buffer. Then if you exit Disk Review and go to the 80 column v5 text editor this text will be waiting for you in the editor's view buffer. Just press "V" from the editor's command line to see the text you loaded into Disk Review! {Because 40 column systems have only a limited amount of VDP memory, this V(iew) buffer feature is not avail-

---

>1700, safely above the AVPC or Mechatronics at >1400, and have the main system RAMdisk as DSK5 at CRU >1000 for speed as this is the first DSR accessed by normal (and Funnelweb) DSRLinks. ■

able from the 40 column v5 editor.}

FIFTH text in mem. - ST(ore) buffer: From the editor command line you can press ST(ore) and move the contents of the edit buffer into temporary storage in VDP RAM. You can then load another file into the edit buffer, edit the second file and save it back to disk, then press RE from the command line to RE(call) the ST(ore)d text back into the edit buffer. The ST(ore) buffer acts as a temporary ramdisk, but is much faster. Text files are saved and loaded to horizon ramdisks one record (line of text) at a time. This is fast but definately not instantaneous. Large files take 10s of seconds to load and save with a horizon. The ST(ore) buffer response time is immediate! It is too bad you can't exchange text between the edit and store buffers. Tony says this trick would eat up lots of memory and that's why such a feature has not been included. {ST and RC to and from VDP RAM is not available from the 40 column v5 editor. There isn't enough memory.}

FILE SAVING AND PRINTING OPTIONS

These are available in both the 40 and 80 column editors and are accessed via the P(rint)F(ile) command. You can configure the editor with printer codes. Then every time you insert a "P" in front of the printer name (such as PF <enter>, P PIO) the editor will send these preconfigured codes to the printer before any text. I have my v5 editor set up to send the "print all the following in emphasized print" command. If I also use a "Q" with PF the editor will send a printer reset code to the printer after all text has been printed (PF <enter>, P Q PIO).

You can append the contents of the text buffer to the end of an existing disk file by specifying the disk file as the printer device preceeed by an "A" (PF <enter>, A DSKx. FILENAME). DV80 files of unlimited size can be created this way.

You can also use PF to create DF128 text files readable directly by MS-DOS and Unix software.

NEW POWERUP OPTIONS

Normally the v5 editor boots in either Word Processing or Program Editing mode depending upon which of Funnelweb's central menus is used to select the editor. However, if you hold the space bar down as the editor loads you get a list of choices. The editor can be pre-configured to always give you this list of choices without pressing the space bar or to automatically boot as any one of the choices unless you press the space bar.

1. Word Processing
2. Program Editor

Then you get these choices:

1. Default 7-bit
2. National 7-bit
4. All Chararcters
3. TI Euro-Writer

If you want the resulting disk file of your document to be readable by someone else on another computer using anything except Funnelweb v5 (such as an earlier version of Funnelweb, or TI Writer) then select items 1 or 2 from this menu. Items 3 and 4 from the above menu do some fancy stuff (more about this later), but produce disk files that can only be read and displayed on screen properly with Funnelweb v5.

After you chose one of these options, you are given the following choice of languages, comparable to what is suggested by the TI Writer module:

1. USA---My 40 column beta test editor lists this as "default".
2. Britain---Choosing this loads in a separate character set that redefines SHIFT/3 as the British pound sterling symbol.
3. France
4. Deutschland
5. Italia
6. Sverige
7. Nederland
8. Espana

Choosing a non English language loads in foreign language character sets that redefine little used keyboard characters such as FCTN/A,

FCTN/F, FCTN/W, etc as appropriate foreign characters. Most of these foreign characters are vouls with accent symbols over them such as umlaut, grave, acute, or circumflex. These character sets and their ASCII values correspond to some of the international character sets 1-9 found on most modern printers. This means that if you send a control code to set your printer for the appropriate foreign character set then the foreign characters you see on screen will be printed properly. From the editor the epson compatible printer key sequence with no spaces between keypresses is CTRL/U FCTN/R CTRL/U R CTRL/U SHIFT/A thru I CTRL/U where A-I specify the particular character set (1-9) desired. For Genimi 10X and SG10 printers substitute 7 for R in the above key sequence.

Non-English languages also load appropriate foreign text into the command line and change the command abbreviations to reflect the foreign language. For example, in French "Imprimer Fichier" means Print File, and you use the command IF, not PF, to print stuff. The Swedish version has "Lagra Filer" for Save File. The command LF in the Swedish text editor will save (not load) a file. This can be disasterous for English speakers who don't know Swedish.

Not all the foreign commands and command line text are finished. English, German, and Swedish are complete. French and Dutch are almost complete. Spanish hasn't been started. Sample source code and a utility that creates foreign commands and command line text are included for those interested in expanding Funnelweb v5's multilingual capabilities.

ALL CHARACTERS MODE

Our 99/4As normally can directly type ASCII 0-127 with ASCII characters below 32 accessed from CTRL/U "special character mode". But our 8 bit computer is capable of generating codes 0-255. When high ASCII codes >127 are sent to a printer during text printing the printer will print graphic symbols. A common standard for these high ASCII graphics is the IBM character set #2 found on most printers. High ASCII codes sent to a printer with IBM graphics #2 enabled print line shapes. Check your printer's manual to see what these graphics look like.
[SPECIAL NOTE FOR STAR SG10 PRINTER OWNERS: There is an undocumented software method of switching from STAR mode to the IBM character set #2. You don't need to use a dip switch. The code with no spaces between keypresses is CTRL/U FCTN/R CTRL/U w CTRL/U SHIFT/A CTRL/U. To switch from IBM set #2 back to STAR mode use this code: CTRL/U FCTN/R CTRL/U w CTRL/U SHIFT/2 CTRL/U. The w in these codes is lower case.]

Selecting All Chars mode with the Funnelweb v5 editor allows you to directly type on screen and print to the printer ASCII 0-254 of the IBM character set #2. This includes all the normal upper and lower case letters numbers and keyboard symbols, plus the graphic symbols coded by high ASCII numbers. To type the graphic symbols type CTRL/, (control and comma simultaneously) and then each keypress will produce a graphic symbol. To return to the keyboard normal letters type CTRL/, again. Normal letters and graphic symbols remain on screen as you use CTRL/, to toggle the keyboard back and forth beteen graphics and normal. Graphics and text print normally using PF (PrintFile). You don't need a formatter to print these graphic symbols.

EURO-WRITER

In Europe, TI released a multilingual version of TI Writer (TIW v2) with some special features. By selecting EURO-WRITER from the powerup menus, the Funnelweb v5 editor provides all the features of the TI Writer v2 editor; specifically an intuitive way of adding accent marks to vouls.

When in Funnelweb's EuroWriter mode you have access to the foreign character set of the language you select from the powerup menus, and these character sets include some,

but not all, accented vouls. But there is another intuitive way to create accented vouls that lets you put ANY ACCENT over ANY VOUL. Type a voul, either upper or lower case. Then backspace to put the cursor back over the voul, type any of four FCTN/key or CTRL/key combinations, and an umlaut grave acute or circumflex mark will appear on screen over the voul! The only problem is that these voul/backspace/accent screen displays are coded with high ASCII numbers above 127 and don't normally print properly. You need to print text files with these accented vouls using the European formatter (also multingual), the formatter that TI included with TI Writer v2. You need to use special transliteration files that redefine ASCII codes greater than 127 as accented vouls. This formatter with its auxiliary language and transliteration files is not part of the Funnelweb v5 editer package, but the files can be obtained by anyone from the Lima user group. Unfortunately, the European formatter REQUIRES use of the TI Writer module. It hasn't yet been modified to run easily out of the Funnelweb environment using something other than the TIW module to boot Funnelweb. Also, transliterations to print some of the accented vouls are less than ideal. Accented vouls you see clearly on screen with Funnelweb's Eurowriter mode may look strange when printed.

SOME OF THE OTHER NEW FEATURES

--You can move text up and down from within the command line. This is very handy for M(ove lines)), D(elete lines), and C(opy lines) operations. You don't have to remember line numbers. Go to the command line and type M, D, or C. Then use the arrow or up/down screen keys to display the first line number and last line number so that you will enter the proper numbers to M, D, or C.

--From the various fixed modes with an open box cursor (Program editor or WP with word wrap off) you can break lines at the cursor, insert text, then rejoin with the next text line. This means you can insert text into the middle of a paragraph from a fixed mode without losing existing text off the end of the right margin, something no other version of TI Writer will allow.

--Typing a number in a blank command line followed by <enter> will put that line at the top of the screen and return to edit mode (S before line number not necessary). <Enter> from a blank command line returns to edit mode (E prior to <enter> not necessary).

--You can freeze the display beginning with the line below the cursor while continuing to scroll, window, and edit from the cursor line to the top of the screen. This means you can simultaneously display two parts of the edit buffer with full editing capabilities for one of these displayed parts.

--You can put a bookmark (mark the text) at any line number from either command mode or edit mode. Later you can put the cursor on this text with FCTN/= even if the text has been edited since marking.

--You can display the contents of any hard drive path from the command line similar to doing a SD. Enter "HD" from the command line, then type a path name and press enter. The resulting display of that directory's file names resembles the SD display, and you can mark DV80 files for loading into the editor. This should be great for hard drive users who have trouble cataloging their hard drives with existing software.

--From SD or HD two different files can be marked, the regular and "temporary" file. These can be loaded into the editor with LF (regular) and LT (loads the "temporary" file).

--A user definable wild card character can be used the string searches with FS and RS.

--When you load, print, or save files an incrementing number in the upper right of the screen shows the current line being loaded into or out of memory. ■

TIPS FROM TIGERCUB #65

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. I am out of printed documentation so it will be supplied on 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.

It is a bit of a nuisance to have to hit Enter after inputting a single character such as Y or N for "yes" or "no". CALL KEY accepts a single character without Enter, but has no blinking cursor to tell you that it is waiting. I should have had this one in my Nuts & Bolts years ago - the CALL KEY WITH CURSOR subprogram! R is the row, C is the TAB position, V$ is the validation string, such as "YyNn", and the character selected is returned in K$.

```
30000 SUB CALLKEY(R,C,V$,K$)
30001 CALL HCHAR(R,C+2,30)::
 FOR T=1 TO 3 :: CALL KEY(0,
K,S):: IF S<>0 THEN 30004
30002 NEXT T :: CALL HCHAR(R
,C+2,20):: FOR T=1 TO 3 :: C
ALL KEY(0,K,S):: IF S<>0 THE
N 30004
30003 NEXT T :: GOTO 30001
30004 IF POS(V$,CHR$(K),1)=0
 THEN 30001 ELSE K$=CHR$(K)
30005 SUBEND
```

And for a demonstration of the use of that subprogram, here is a little game that no one will ever play to the end -

```
100 DISPLAY AT(3,6)ERASE ALL
:"THE ULTIMATE TEST":"":" An
swer the question with a num
ber according to whether the
 number or color shown,"
110 DISPLAY AT(8,1):"or the
note sounded, was 1stor 2nd
or 3rd, etc."
120 DISPLAY AT(23,6):"PRESS
ANY KEY" :: DISPLAY AT(23,6)
:"press any key" :: CALL KEY
(0,K,SS):: IF SS=0 THEN 120
ELSE CALL CLEAR
130 DATA 2,BLACK,3,GREEN,5,B
LUE,9,RED,12,YELLOW,14,PURPL
E
140 FOR J=1 TO 6 :: READ C(J
),C$(J):: CT$=CT$&CHR$(J)::
W$=W$&CHR$(J+48):: NEXT J ::
 T=2 :: DL=500 :: V$="12"
150 RANDOMIZE :: T$,NN$=CT$
:: FOR J=1 TO T :: X=INT(RND
*LEN(T$)+1):: X$=SEG$(T$,X,1
):: T$=SEG$(T$,1,X-1)&SEG$(T
$,X+1,255):: Y(J)=ASC(X$)
160 X=INT(RND*LEN(NN$)+1)::
X$=SEG$(NN$,X,1):: NN$=SEG$(
NN$,1,X-1)&SEG$(NN$,X+1,255)
:: S(J)=ASC(X$):: NEXT J ::
FOR J=1 TO T
170 Z(J)=INT(89*RND+10):: FO
R K=1 TO J-1 :: IF Z(J)=Z(K)
THEN 170
180 NEXT K :: NEXT J :: CALL
 CLEAR :: CALL COLOR(3,16,1,
4,16,1)
190 FOR J=1 TO T :: CALL SCR
EEN(C(Y(J))):: CALL SOUND(-9
99,110*S(J),0):: DISPLAY AT(
12,12):Z(J):: FOR D=1 TO DL
:: NEXT D :: NEXT J
200 CALL CLEAR :: CALL SCREE
N(16):: CALL COLOR(3,2,1,4,2
,1):: X=INT(3*RND+1):: W=INT
(T*RND+1):: ON X GOTO 210,23
0,210
210 IF X=1 THEN Q$=C$(Y(W))E
LSE IF X=3 THEN Q$=STR$(Z(W)
)
220 DISPLAY AT(12,1):"WHICH
WAS ";Q$ :: GOTO 240
230 CALL SOUND(1,30000,30)::
 DISPLAY AT(12,1):"WHICH WAS
?" :: FOR D=1 TO 200 :: NEXT
 D :: CALL SOUND(500,110*S(W
),0)
240 CALL CALLKEY(12,20,V$,K$
):: Q=ASC(K$)-48
250 IF Q=W THEN DISPLAY AT(1
5,12):"RIGHT!" ELSE DISPLAY
AT(15,12):"WRONG!"
260 IF Q=W THEN DL=DL-50 ELS
E DL=DL+50
270 IF DL<100 THEN DL=500 ::
 T=T+1 :: V$=SEG$(W$,1,T)
280 GOTO 150
290 SUB CALLKEY(R,C,V$,K$)
300 CALL HCHAR(R,C+2,30):: F
OR T=1 TO 3 :: CALL KEY(0,K,
S):: IF S<>0 THEN 330
310 NEXT T :: CALL HCHAR(R,C
+2,20):: FOR T=1 TO 3 :: CAL
L KEY(0,K,S):: IF S<>0 THEN
330
320 NEXT T :: GOTO 300
330 IF POS(V$,CHR$(K),1)=0 T
HEN 300 ELSE K$=CHR$(K)
340 SUBEND
```

I have warned repeatedly over the years, in these Tips and in Micropendium and elsewhere, that printing program listings through the Funlweb Formatter usually results in garbled listings that cannot be keyed in correctly - but I still see the garbled listings published. Here is a fix to the Funlweb FO file that will partially solve the problem -
Boot DSKU. Select 1. File Utilities. Select 5. Find String. Enter filename FO and the drive number. Enter H for hex. Enter the string 2A23214026 . Enter replace string 7C2321605C . When the string is found, enter R for replace, then CTRL W, hit Enter twice to accept the defaults. Thereafter, use FCTN Z instead of & to underline, FCTN C instead of @ to double-strike, and FCTN A instead of * to call a value added file. I don't know why Texas Instruments didn't do that in the first place, and I wonder why the McGoverns didn't make that fix.

Now, can anyone tell me how to replace the ^, which tends to disappear, and the period, which will make the whole line disappear if it happens to be at the begin-

ning of the line?

If you are one of the few who are still interested in recreational computing - the use of the computer to solve puzzles and math problems just for the fun of it - you might be interested in Recreational and Educational Computing, published 8 times a year at 909 Violet Terrace Clarks Summit PA 18411. The annual subscription is $27. Program listings are in dialects of Basic other than TI but usually not hard to convert.

That is where I found this ridiculously short, simple and fast card shuffling routine.

```
100 DIM C(52)
110 FOR X=1 TO 52 :: C(X)=X
:: NEXT X
120 FOR X=52 TO 1 STEP -1 ::
 I=INT(RND*X+1)
130 T=C(I):: C(I)=C(X):: C(X
)=T :: NEXT X
```

In the same place, I read a routine to extract a root to 16-digit accuracy instead of the 8 digits available on a PC from the basic formula ROOT=NUMBER^(1/POWER). We don't need it - our obsolete 16k 16-bit computer can give us 14-digit accuracy from the basic formula!

The same publication gave me the idea for this little game -

```
100 DISPLAY AT(3,6)ERASE ALL
:"THE GAME OF N":"":"You and
 the computer will  take tu
rns adding to a num- ber to
reach a goal."
110 DISPLAY AT(8,1):"If you
reach the goal, you win. Yo
u get to go first andyou sho
uld be able to win  almost
every time."
120 RANDOMIZE :: N=INT(RND*1
5)+15 :: R=INT(4*RND+3):: S=
R+1 :: D=N-INT(N/S)*S :: T=0
130 DISPLAY AT(13,1):"The go
al is";N:"":"Maximum input i
s";R :: DISPLAY AT(19,1):RPT
$(" ",28*6)
140 DISPLAY AT(17,1):"Your n
umber?" :: ACCEPT AT(17,14)S
IZE(1)VALIDATE(DIGIT):A :: I
F A<1 OR A>R THEN DISPLAY AT
(15,1):"" :: GOTO 130
150 T=T+A :: DISPLAY AT(21,1
):"Total is";T :: IF T=N THE
N DISPLAY AT(23,1):"YOU WIN!
" :: GOSUB 190 :: GOTO 120
160 IF N-T<S THEN P=N-T :: T
=T+P :: DISPLAY AT(19,1):"Co
mputer adds";P :: DISPLAY AT
(21,1):"Total is";T :: DISPL
AY AT(23,1):"COMPUTER WINS!"
 :: GOSUB 190 :: GOTO 120
170 IF T=0 THEN P=D ELSE IF
(N-T)/S=INT((N-T)/S)THEN P=I
NT(R*RND+1)ELSE Y=N-T :: P=Y
-INT(Y/S)*S
180 T=T+P :: DISPLAY AT(19,1
):"Computer adds";P :: DISPL
AY AT(21,1):"Total is";T ::
GOTO 140
190 DISPLAY AT(24,8):"PRESS
ANY KEY" :: DISPLAY AT(24,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 190 EL
SE T=0 :: RETURN
```

REC also printed a puzzle which seemed so simple that I could not see why. It goes like this -

A game show host shows you three curtains. Behind one is a new car, behind the other two are goats. You choose one. The host, who can peek behind the curtain, opens one of those you did not pick, and shows a goat. Then he offers to let you change your choice. Should you switch, stand pat, or does it make no difference?

You now have a 50-50 bet, so it makes no difference, right? But some very distinguished mathematicians were saying you should switch, so I wrote this computer simulation to prove them wrong. Key it in, run it, and be surprised. Do figures lie? Do computers lie? Is there something wrong with my simulation?

```
100 CALL CLEAR
110 DATA CAR BEHIND,A PICKS,
HOST SHOWS,A WINS,B WINS,C W
INS
120 FOR J=1 TO 3 :: READ M$
:: DISPLAY AT(J,1):M$ :: NEX
T J :: FOR J=12 TO 14 :: REA
D M$ :: DISPLAY AT(J,1):M$ :
: NEXT J
130 FOR J=1 TO 1000 :: RANDO
MIZE :: X=INT(3*RND+1):: DIS
PLAY AT(1,13):X !RANDOMLY PL
ACE CAR
140 A=INT(3*RND+1):: DISPLAY
 AT(2,13):A !PLAYER CHOOSES
150 D=INT(3*RND+1):: IF D=X
OR D=A THEN 150 :: DISPLAY A
T(3,13):D :: ! HOST PICKS CU
RTAIN WITH GOAT
160 IF A=X THEN AA=AA+1 :: D
ISPLAY AT(12,7):AA ! A DOES
NOT SWITCH
170 B=INT(3*RND+1):: IF B=A
OR B=D THEN 170
180 IF B=X THEN BB=BB+1 :: D
ISPLAY AT(13,7):BB ! B SWITC
HES
190 C=INT(3*RND+1):: IF C=D
THEN 190
200 IF C=X THEN CC=CC+1 :: D
ISPLAY AT(14,6):CC ! C CHOOS
ES RANDOMLY
210 NEXT J
```

Here is an improved version of a program that was in a Tips long ago, to strip out the extra blanks from a Filled and Adjusted Funlweb Formatter file -

```
100 DISPLAY AT(3,6)ERASE ALL
:"TIGERCUB UNFILLER":"":" To
 remove extra spaces from":"
a TI-Writer text which has":
"been Filled and Adjusted by
"
110 DISPLAY AT(8,1):"the For
matter, prior to":"reformatt
ing."
120 DISPLAY AT(15,1):"Input
file? DSK" :: ACCEPT AT(15,1
6):IF$ :: OPEN #1:"DSK"&IF$,
INPUT
130 DISPLAY AT(17,1):"Output
 file? DSK" :: ACCEPT AT(17,
17):OF$ :: OPEN #2:"DSK"&OF$
140 LINPUT #1:M$ :: P=1
150 X=POS(M$," ",P):: IF X=P
 THEN P=P+1 :: GOTO 150
160 X=POS(M$,"  ",P):: IF X=
0 THEN PRINT #2:M$ :: GOTO 1
80
170 M$=SEG$(M$,1,X)&SEG$(M$,
X+2,255):: GOTO 160
```

```
180 IF EOF(1)<>1 THEN 140 ::
 CLOSE #1 :: CLOSE #2
```

While a program is running, the computer periodically pauses for a fraction of a second to do a "garbage collection", getting rid of information it no longer needs, to make room in memory. If this pause occurs at a critical moment in program execution, it can cause problems. Thanks to the Sydney User Group in Australia, here is a CALL LOAD which will force a garbage collection just before that critical point -

```
CALL LOAD(-31885,144,"",-318
58,81,169,152,0)
```

Here is a neat one from Bruce Harrison. Key it in, (you can skip the lines that start with an asterisk) and assemble it, then use ALSAVE to imbed it in any program that opens a disk file. Put CALL LINK("DEVICE",DEV$) at the beginning of the program and change any line reading OPEN #1:"DSK1.FILENAME" - or whatever - to read - OPEN #1:DEV$&".FILENAME" (don't forget the period before the filename!). Now you can load the program from any drive and it will open the file on that same drive!

```
* STRING ASSIGN DEVICE NAME
* PLACES DEVICE NAME IN AN
*  XBASIC STRING
* HARRISON SOFTWARE
* 8 OCTOBER 1990
* FOR USE WITH ALSAVE AND XB
* TAKES ONLY 42 BYTES MEMORY
STRASG EQU  >2010
WS     EQU  >20BA
       DEF  DEVICE
DEVICE
* USE OUR WORKSPACE
       LWPI WS
* GET THE CRU BASE IN R12
       MOV  @>83D0,R12
* GET ROM ADDRESS FOR DEVICE
*  IN R2
       MOV  @>83D2,R2
* ENABLE THE ROM
       LDCR @ONES,0
* ADDING 4 PUTS US AT THE
*  LENGTH BYTE
       AI   R2,4
* FIRST PARAMETER
       LI   R1,1
* NOT AN ARRAY VARIABLE
       CLR  R0
* ASSIGN DEVICE NAME TO A
*  STRING
       BLWP @STRASG
* CLEAR CRU, DISABLE ROM
       LDCR R0,0
* LOAD GPL WORKSPACE
       LWPI >83E0
* RETURN TO GPL INTERPRETER
       B    @>006A
* WORD TO TURN ON ROM IN CRU
ONES   DATA >0101
       END
```

Getting short on memory, so more next time.

Jim Peterson■

TYPING TEACHER

by Alan McCright, USA

```
80 REM TYPING TEACHER
90 REM COMPUTE 83-04
100 DIM CHAR(23,30)
110 RANDOMIZE
120 D=20
130 F1=300
140 F2=4000
150 V1=10
160 V2=2
170 CALL CLEAR
180 FOR J=9 TO 12
190 CALL COLOR(J,2,14)
200 NEXT J
210 FOR J=2 TO 8
220 CALL COLOR(J,2,15)
230 NEXT J
240 IF R=82 THEN 270
250 RESTORE
260 CALL CLEAR
270 PRINT "      typing teac
her": : : : : : : : : : : :
: : : : : : :
320 CHARCNT=0
330 TIME=0
340 REM ENTER CHAR POS DATA

350 FOR ROW=11 TO 23 STEP 3
360 FOR COL=6 TO 30 STEP 2
370 READ CHAR(ROW,COL)
380 IF CHAR(ROW,COL)=0 THEN
450
390 IF CHAR(ROW,COL)=-1 THEN
 460
400 IF CHAR(ROW,COL)=32 THEN
 430
410 CALL HCHAR(ROW,COL,CHAR(
ROW,COL))
420 GOTO 440
430 PRINT " ";
440 NEXT COL
450 NEXT ROW
460 PRINT :"  PRESS any key
TO START";
480 CALL KEY(3,S,STA)
490 IF STA=0 THEN 480
500 CALL HCHAR(24,5,32,22)
510 REM RND NR
520 N=INT((RND*47)+44)
530 IF (N>=60)*(N<=64)+(N=45
)+(N=58)+(N=OLDCHAR)THEN 520
540 OLDCHAR=N
550 CALL VCHAR(7,16,N)
560 REM PROCESS RESPONSE
570 TIME=TIME+1
580 IF TIME>900 THEN 670
590 CALL KEY(0,CR,STA)
600 IF STA=0 THEN 570
610 CALL SOUND(D,F1,V1)
620 CHARCNT=CHARCNT+1
630 REM ADD ONE TO TOTAL
640 TIME=TIME+12
650 GOTO 760
670 PRINT TAB(4);"characters
/minutes= ";CHARCNT: :"
HIT r TO RESTART";
710 CALL KEY(3,R,STA)
720 IF STA=0 THEN 710
730 IF R=82 THEN 250
750 END
760 IF CR<>N THEN 860
770 FOR ROW=11 TO 23 STEP 3
780 FOR COL=6 TO 30 STEP 2
790 IF CHAR(ROW,COL)=N THEN
820
800 NEXT COL
810 NEXT ROW
820 CALL HCHAR(ROW-1,COL,N)
830 CALL HCHAR(ROW-1,COL,32)
840 CALL HCHAR(ROW-1,COL,N)
850 GOTO 520
860 CHARCNT=CHARCNT-1
870 CALL SOUND(D,F2,V2)
880 GOTO 520
890 REM ASCII FOR KEYBOARD
900 DATA 49,50,51,52,53,54,5
5,56,57,48,61,0
910 DATA 81,87,69,82,84,89,8
5,73,79,80,47,0
920 DATA 65,83,68,70,71,72,7
4,75,76,59,0
930 DATA 32,90,88,67,86,66,7
8,77,44,46,-1,@
```

■

INVENTORY - EXTENDED BASIC

Bob August, BUG News, USA

This program is a records type program to record your household belongings. It's nothing fancy but it works. You can enter up to 14 characters for the item, 4 characters for the year purchased and 7 digits for the replacement cost. If you are using a cassette recorder for storage then change the two open state- ments to OPEN #1:"CS1",INPUT and OPEN #1:"CS1",OUTPUT.

```
100 ! HOUSEHOLD INVENTORY
110 ! IN TI EXTENDED BASIC
120 ! BY R.W. AUGUST
130 DIM ITEM$(100),YR$(100),
COST(100):: N=0 :: T=0
140 DISPLAY AT(4,2)ERASE ALL
:"<< HOUSEHOLD  INVENTORY >>
": : : :"Press:": :"   1) To
 Display items": :"   2) To
Enter new items"
150 DISPLAY AT(14,4):"3) To
Edit items": :"   4) To see
totals": :"   5) To Quit": :
 :"   Your choice [ 1 - 5 ]"
160 CALL KEY(0,K,S):: IF S=0
 OR K<49 OR K>53 THEN 160 ::
 CALL CLEAR :: ON K-48 GOSUB
 1000,2000,3000,4000,5000
170 GOTO 140
180 ! LOAD DATA SECTION
190 DISPLAY AT(10,6)ERASE AL
L:"<< LOADING DATA >>" :: OP
EN #1:"DSK1.INV/DATA",INPUT
:: INPUT #1:N :: T=0
200 FOR I=1 TO N :: INPUT #1
:ITEM$(I),YR$(I),COST(I):: T
=T+COST(I):: NEXT I
210 CLOSE #1 :: RETURN
220 ! SAVE DATA SECTION
230 DISPLAY AT(10,6)ERASE AL
L:"<< SAVING DATA >>" :: OPE
N #1:"DSK1.INV/DATA",OUTPUT
:: PRINT #1:N
240 FOR I=1 TO N :: PRINT #1
:ITEM$(I):YR$(I):COST(I):: N
EXT I
250 CLOSE #1 :: RETURN
260 ! TITLE SECTION
270 DISPLAY AT(1,2)ERASE ALL
:"<< HOUSEHOLD  INVENTORY >>
": :"HOUSEHOLD      YEAR REP
LACE":"ITEM:          PER: C
OST:"
280 DISPLAY AT(5,1):"-------
------- ---- --------" :: RE
TURN
290 DISPLAY AT(24,2):"PRESS
ANY KEY TO CONTINUE"
300 CALL KEY(0,K,S):: IF S=0
 THEN 300 :: RETURN
310 DISPLAY AT(12,6):"NO DAT
A IN MEMORY"
320 DISPLAY AT(14,1):"DO YOU
 WISH TO LOAD OLD DATA": :"[
Yes or No] Y" :: ACCEPT AT(1
6,13)SIZE(-1)VALIDATE("YyNn"
):Y$
330 IF Y$="Y" OR Y$="y" THEN
 GOTO 180 ELSE RETURN
1000 ! DISPLAY ITEMS SECTION
1010 IF N=0 THEN GOSUB 310 :
: IF Y$="N" THEN RETURN
1020 GOSUB 260 :: A=0 :: B=0
1030 FOR I=1 TO N :: A=A+1 :
: B=B+1 :: DISPLAY AT(A+5,1)
:ITEM$(B);TAB(16);YR$(B);TAB
(21);"$";STR$(COST(B)):: IF
A=17 THEN 1050
1040 NEXT I :: GOSUB 290 ::
RETURN
1050 GOSUB 290 :: A=0 :: CAL
L HCHAR(6,1,32,605):: GOTO 1
040
2000 ! ENTER ITEMS SECTION
2010 IF N>0 THEN 2030 ELSE G
OSUB 310 :: DISPLAY AT(8,1)E
RASE ALL:"WARNING!  IF YOU D
O NOT LOAD": :"DATA, ALL DAT
A NOW SAVED"
2020 DISPLAY AT(12,1):"WILL
BE DISTROYED!" :: GOSUB 320
2030 GOSUB 260 :: DISPLAY AT
(6,1):"ENTER END TO QUIT"
2040 FOR I=1 TO 16 :: N=N+1
:: ACCEPT AT(I+6,1)SIZE(14):
ITEM$(N):: IF ITEM$(N)="END"
 THEN N=N-1 :: GOSUB 220 ::
RETURN
2050 ACCEPT AT(I+6,16)SIZE(4
):YR$(N):: ACCEPT AT(I+6,22)
SIZE(7):COST(N):: T=T+COST(N
):: NEXT I
2060 GOSUB 290 :: CALL HCHAR
(7,1,32,573):: GOTO 2040
3000 ! EDIT SECTION
3010 IF N=0 THEN GOSUB 310 :
: IF Y$="N" THEN RETURN
3020 GOSUB 260
3030 DISPLAY AT(12,1):"ENTER
 NAME OF ITEM TO EDIT" :: AC
CEPT AT(14,1):HI$
3040 FOR I=1 TO N :: IF ITEM
$(I)=HI$ THEN 3060
3050 NEXT I :: DISPLAY AT(16
,1):"NO MATCH FOUND" :: GOSU
B 290 :: RETURN
3060 DISPLAY AT(6,1):ITEM$(I
);TAB(16);YR$(I);TAB(21);"$"
;STR$(COST(I)):: ACCEPT AT(6
,1)SIZE(-14):ITEM$(I):: ACCE
PT AT(6,16)SIZE(-4):YR$(I)
3070 ACCEPT AT(6,22)SIZE(-7)
:COST(I):: GOSUB 290 :: GOSU
B 220 :: RETURN
4000 ! Totals Section
4010 IF N=0 THEN DISPLAY AT(
12,6):"NO DATA IN MEMORY": :
" PRESS ANY KEY TO CONTINUE"
 :: GOSUB 300 :: RETURN
4020 DISPLAY AT(1,2)ERASE AL
L:"<< HOUSEHOLD  INVENTORY >
>"
4030 DISPLAY AT(10,1):"TOTAL
:": :"NUMBER OF ITEMS:";N: :
"COST OF ITEMS: $";STR$(T)::
 GOSUB 290 :: RETURN
5000 ! EXIT SECTION
5010 CALL CLEAR :: END
```

In this months program I was going to use ON ERROR, but didn't becauce there isn't anything like it in basic. The ON ERROR statement will keep your program from bombing when you enter wrong data or try to open a non existing disk file. When you use it, be sure to go to an ON ERROR statement each time you use it. If you only put it at the begining of your program it wil work the first time and then not again. Heres an example using this months program:

```
130 DIM ITEM$(100),YR$(100),
COST(100):: N=0 :: T=0 :: ON
ERROR 6000
170 ON ERROR 6000 :: GOTO 14
0
6000 ! ERROR ROUTINE
6010 FOR E=1 TO 5
6020 CALL SOUND(250,110,0)
6030 DISPLAY AT(12,1)ERASE A
LL:"ERROR HAS OCCURED"
6040 FOR D=1 TO 100 :: NEXT
D :: NEXT E
6050 RETURN 170
```

We put the first ON ERROR in line 130 so it would pick up any errors before line 170. Line 170 resets the ON ERROR routine each time after an error. Lines 6000 to 6050

just give you something to put on the screen so you know an error has occured. In the program we could use this to find out if a file has been created and if it has been it would load. Otherwise it would give you an error and let you create a file. Try this in the program and see what happens: Add lines:
185 ON ERROR 215
215 RETURN 216
216 RETURN
Change line 2010 to:
2010 IF N=0 THEN GOSUB 180
Remove line 2020.

Now run your program and the first time you try to enter new items the program will look for the file DSK1.INV/DATA and as it is not there the ON ERROR routine will return the program to line 2030. Hope this helps you in using the ON ERROR routine. ■

GOBLIN - SPEL BASIC

Använd S och D för att ta figurerna. Undvik rutorna

```
60 REM GOBLIN
70 REM BY D.GOFF
80 REM COMPUTE 83-07 REV
100 RANDOMIZE
110 GOTO 170
120 FOR I=1 TO LEN(H$)
130 R=ASC(SEG$(H$,I,1))
140 CALL HCHAR(ROW,XCOL+I,R)
150 NEXT I
160 RETURN
170 A=96
180 B=97
190 C=104
200 D=105
210 Z=24
220 COL=16
230 W=0
240 G=0
250 S=J
260 CALL CLEAR
270 GOSUB 1270
280 CALL SCREEN(16)
290 PRINT "        G O B L I
 N"
310 PRINT :"          HS : "
: : : : : : : : : : : : : :
: : : : : :
350 PRINT "[=LEFT
   ]=RIGHT";
360 ROW=4
370 XCOL=17
380 H$=STR$(HS)
390 GOSUB 120
400 FOR I=1 TO 50
410 X=INT(RND*30)+2
420 Y=INT(RND*16)+6
430 CALL GCHAR(Y,X,L)
440 IF L=B THEN 410
450 CALL HCHAR(Y,X,B)
460 NEXT I
470 FOR I=1 TO 27
480 X=INT(RND*30)+2
490 Y=INT(RND*16)+6
500 CALL GCHAR(Y,X,L)
510 IF (L=B)+(L=C)+(L=D)THEN
 480
520 CALL GCHAR(Y+1,X-1,L)
530 CALL GCHAR(Y+1,X,M)
540 CALL GCHAR(Y+1,X+1,N)
550 IF (L<>B)+(M<>B)+(N<>B)T
HEN 590
560 CALL HCHAR(Y,X,D)
570 G=G+1
580 GOTO 600
590 CALL HCHAR(Y,X,C)
600 NEXT I
610 CALL SOUND(100,500,6)
620 CALL HCHAR(Z,COL,32)
630 IF L<>C THEN 650
640 CALL SOUND(10,880,4)
650 Z=Z-1
660 IF Z>4 THEN 680
670 Z=23
680 CALL KEY(3,L,ST)
690 IF (L<>83)*(L<>68)THEN 7
40
700 IF L<>83 THEN 730
710 COL=COL-SGN(COL-2)
720 GOTO 740
730 COL=COL+SGN(31-COL)
740 CALL GCHAR(Z,COL,L)
750 IF L=B THEN 1030
760 IF L=C THEN 820
770 CALL HCHAR(Z,COL,A)
780 FOR I=1 TO 25
790 NEXT I
800 IF W=27-G THEN 890
810 GOTO 620
820 W=W+1
830 S=S+25
840 H$=STR$(S)
850 ROW=4
860 XCOL=3
870 GOSUB 120
880 GOTO 770
890 J=S
900 CALL HCHAR(10,1,32,31)
910 GOSUB 120
920 H$="**** ALL RIGHT! ****
"
930 XCOL=6
940 ROW=10
950 GOSUB 120
960 FOR I=1 TO 15
970 X=INT(RND*100)+300
980 CALL SOUND(75,X,8)
990 NEXT I
1000 FOR I=1 TO 100
1010 NEXT I
1020 GOTO 210
1030 REM YOU CRASHED
1040 CALL HCHAR(Z,COL,98)
1050 FOR I=3 TO 30 STEP 3
1060 CALL SOUND(50,-7,I)
1070 NEXT I
1080 CALL CHAR(104,"3C42A581
A599423C")
1090 J=0
1100 IF S<=HS THEN 1120
1110 HS=S
1120 H$="PLAY AGAIN (Y/N)?"
1130 ROW=22
1140 XCOL=2
1150 GOSUB 120
1160 CALL KEY(3,L,ST)
1170 IF ST=0 THEN 1160
1180 H$=CHR$(L)
1190 IF H$="Y" THEN 1230
1200 CALL CLEAR
1210 PRINT "SEE YA!"
1220 END
1230 CALL CHAR(104,"3C3CA581
99A5423C")
1240 GOTO 210
1250 REM DEF CUSTOMS CHARS
1260 REM GOBLIN
1270 CALL CHAR(96,"7EDBDBFFA
55A5AA5")
1280 REM BARRIER
1290 CALL CHAR(97,"CCCC3333C
CCC3333")
1300 REM CHRUNCED GOBLIN
1310 CALL CHAR(98,"CCCC33337
EDBFFBD")
1320 REM FROWN
1330 CALL CHAR(104,"3C3CA581
99A5423C")
1340 REM SMILE
1350 CALL CHAR(105,"3C42A581
A599423C")
1360 CALL COLOR(10,7,1)
1370 CALL CHAR(91,"102040FF4
0201")
1380 CALL CHAR(93,"080402FF0
20408")
1390 FOR I=5 TO 8
1400 CALL COLOR(I,2,12)
1410 NEXT I
1420 RETURN
```
■