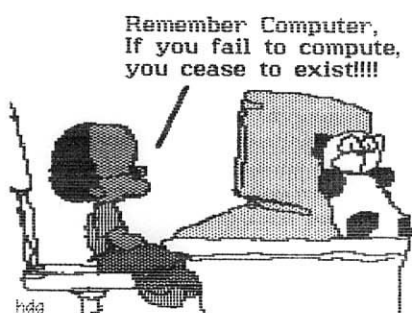


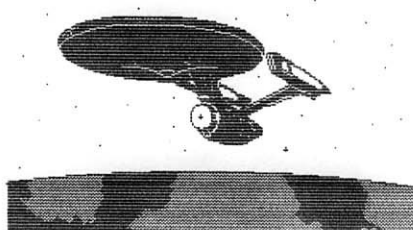
PROGRAM

BITEN

91-2



"WHERE NO MAN HAS GONE BEFORE"



Redaktören	2
Arsberättelse 1990	3
Solens upp- och nedgång	4-8
Disk Patch	8
99/4A från grunden - 1	9-11
Högerjusterade tal	11
BUBBLE - assembler	12-15
Ändra markör i XB	15
Beginner Assembler - 2	16-20
Perspektiv	20
Tigercub Tips #41	21-23
HTA - TI Writer	24-26
Music Preprocessor	26
Sortering i Basic	27-28
Norm Sellers	29
Asgard	30-31
User group Berlin	32

ISSN 0281-1146

REDAKTÖREN

Vid årsmötet som hölls den 16 mars valdes följande styrelse för 1991: Claes Schibler, ordförande
John Hanssen, kassör
Jan Alexandersson, Åke Olsson,
Börje Häll, Peter Olsson.
Vi återkommer med protokoll från årsmötet i nästa nummer av PB.

Micropendium jan 91 sid 7 har en felaktig insändare av Lutz Winkler. YAPP för 80-kolumnskort har verkligen 512x424 punkter med unik information för varje punkt. Det går åt 128 kbytes VDP-RAM. Äldre program som MY-Art har det dock inte med interlace inkopplat.

John Birdwell, 41, dog den 27 dec 1990 i cancer. Han var mest känd som författare till Disk Utilities (som nu finns i version 4.2 i programbanken). Han har även skrivit mycket av Disk Manager V för Myarc HFDC. Inkomster från försäljning av hans datorutrustning och alla fairware-donationer går till en fond som kommer att dela ut ett årligt pris till någon 99-anshänare. Du kan sända din donation eller beställa den kompletta manualen från:
The John Birdwell Memorial Fund
c/o Chicago TI Users Group
P.O. Box 578341
CHICAGO, IL 60657, USA

Manusstopp 15 maj PB 91-3
15 aug PB 91-4
Sänd in dina program och artiklar till redaktören (tel. 08-771 05 69):
Jan Alexandersson
Springarvägen 5, 3 tr
142 61 TRÅNGSUND ■

SÄLJES!

TI-99/4A med nätaggregat och modulator. Expansionsbox med 32k minne, diskkontrollkort, diskdrive, RS232 kort, joystick, kassettspelare, diskettbox med ca 50 disketter, 14" färgskärm, mängder med nytto- och spelprogram, kablar, original bruks, 5 kompl. årgångar av Programbiten. Språk: ex-basic, C99. med manualer. PRISIDÉ: 5000:-. Ev kan en bra printer säljas separat.
Thomas Gröning, S Fiskebäcksv. 188, 421 58 V FRÖLUNDA. Tel. dag 010-34 09 76, kväll: 031-29 85 16 ■

Redaktör: Jan Alexandersson
Utmanarredaktör: Anders Persson
TI-74 redaktör: Lars Herold Andersen
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 1991 är 120:-

Datainspektionens licensnummer:
82100488

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

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

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

Föreningens tillbehörsförsäljning: Följande tillbehör finns att köpa genom att motsvarande belopp insätts på postgiro 19 83 00-6 (porto ingår)

Användartips med Mini Memory	20:-
Nittinian T-tröja	40:-
99er mag. 12/82, 1-5, 7-9/83(st)	40:-
Nittinian årgång 1983	50:-
Programbiten årgång 84-89(styck)	50:-
1990	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.

FÖRENINGEN PROGRAMBITEN
ÅRSBERÄTTELSE 1991-02-27

STYRELSENS ÅRSREDOVISNING

Verksamheten 1990 har bedrivits i form av tidningsutgivning och programförmedling. Antalet medlemmar har minskat från 80 till 66.

STYRELSE OCH FUNKTIONÄRER

Föreningens styrelse:

Claes Schibler, ordf. och registerf.
John Hanssen, kassör
Åke Olsson, sekreterare
Jan Alexandersson, redaktör
Börje Häll, programbankir
Peter Olsson
Mikael Nordlin
Per Rundquist

BALANSRÄKNING

Ingående Balans 1990

Tillgångar

Postgiro	14923.80
Varulager	0.00
Skattefordringar	0.00
Summa	14923.80

Under året har 5 protokollförda styrelsemöten hållits.

FÖRENINGENS EKONOMI

Årets förlust uppgår till 2464.82 kr.

FÖRENINGENS VERKSAMHET

Sex nummer av tidningen Programbiten har framställts.

Flexskivor med textfiler och program har kommit från Jim Swedlow, Tigercub, Tony McGovern, Alexander Hulpke, Texaments(TML demo), Bill Sponchia och Larry Tippet.

Arvode till funktionärer har ej utgått.

Utgående Balans 1990

Tillgångar

Postgiro	12578.98
Varulager	0.00
Skattefordringar	0.00
Summa	12578.98

Skulder

Leverantörsskulder	0.00
Förutbetalda medlemsavg.	600.00
Eget kapital	14323.00
	14923.00

Skulder

Leverantörsskulder	0.00
Förutbetalda medlemsavg.	720.00
Eget kapital	11858.98
	12578.98

RESULTATRÄKNING

Intäkter

Medlemsavgifter	7880.00
Varuförsäljning	1035.00
Räntor	582.18
Överskjutande skatt	896.00
Årets förlust	2464.82
Summa	12858.00

Kostnader

Tryckkostnader	7236.00
Varuinköp	770.00
Varulagersförändring	0.00
Förbrukningmat.+PORTO	3662.00
Skatt	1190.00
	12858.00

Stockholm som ovan

Claes Schibler, ordf

John Hansen, kassör

SOLENS UPP- OCH NEDGÅNG

av Erland Ericsson

Detta program är baserat på en bok :
Peter Duffet-Smith
Med miniräknaren i rymden
Mannerheim & Mannerheim
Stockholm 1983

Boken förklarar på ett utmärkt sätt hur man gör alla tänkbara beräkningar ifråga om himlakroppar. Där kan man alltså få förklaringar till alla de konstigheter som finns i programmet.

En annan bok av intresse i sammanhanget är: Den svenska almanackan, som ges ut varje år och som fn kostar ca 60:-.

Programmet finns i två versioner: Den ena, Printsol, skriver ut på printern när det börjar ljusna, när solen går upp, när den går ner och när det blir mörkt. Dessutom hur lång skymningen är och solens deklination. Detta görs för varje vecka under året.

Det andra programmet, Visasol, visar på skärmen samma uppgifter fyra gånger under angiven månad. Med hjälp av deklinationen kan man räkna

ut hur högt solen står på himmelen:
 $90 + \text{deklinations} - \text{latitud} = \text{solens höjd över horisonten}$

Latitud och longitud tar man från en karta. Lägg märke till att longituden skall räknas från Greenwich och att den är positiv här i Sverige. Väster om Greenwich räknas den negativ. Liknande gäller för latituden: Den räknas positiv på norra halvklotet och negativ på södra.

I programmet finns inlagt att vi befinner oss i tidszon +1 (rad 500). Jag har lagt in att det är sommartid 22 mars - 22 september, vilket stämmer ganska bra.

Med skymning avses här den BORGERLIGA skymningen. Vill man hellre ha någon av de andra definitionerna, den NAUTISKA eller den ASTRONOMISKA, får man ändra på rad 10810 i programmet. Ändra siffran 96 till 102 respektive 108. Tiderna stämmer inte alltid på minuten och deklinationen slår väl fel på någon grad ibland. Det finns också ett par skönhetsfel i presentationen av resultaten.

```
100 ! PRINTSOL
110 ! 1991.01.12 EE
120 CALL CLEAR
130 PRINT "SOLENS UPP- OCH N
EDGANG"
140 INPUT "ORT,AR ":ORT$,AA
:: INPUT "LATITUD GG,MM ":G
RAD,MI :: INPUT "LONGITUD GG
,MM ":GRAL,MIL
150 OPEN #9:"RS232.PA=E.BA=1
200" :: PRINT #9:CHR$(27);"B
"
160 DATA Jan, Feb, Mars, Apr, Ma
j, Juni, Juli, Aug, Sept, Okt, Nov
, Dec
170 PRINT #9:"Solens upp- oc
h nedgång samt":"gryning och
skymning": :
180 PRINT #9:ORT$, "LAT ";GR
AD;": ";MI:AA, "LONG ";GRAL;":
";MIL:
190 PRINT #9:TAB(4);"DATUM";
TAB(12);"DAGEN";TAB(18);"SOL
-";TAB(24);"SOL-";TAB(30);"M
```

```
ÖRKT";TAB(36);"SKYM-";TAB(42
);"ANM";TAB(50);"DEKL"
200 PRINT #9:TAB(12);"GRYR";
TAB(18);"UPP";TAB(24);"NEDG"
;TAB(36);"NING"
210 FOR M=1 TO 12:READ MAN$
:: PRINT #9:MAN$;
220 PRINT MAN$
230 FOR W=1 TO 22 STEP 7 ::
PRINT W :: GOSUB 300
240 PRINT #9:TAB(6);W;TAB(12
);GRT$;TAB(18);UTR$;TAB(24);
UTS$;TAB(30);SKT$;TAB(36);TS
K$;TAB(42);S$;TAB(44);X$;TAB
(50);DEK$ :: NEXT W
250 NEXT M
260 PRINT #9:"S = Korr för s
ommartid": "> = Solen över el
ler under horisonten": "< = S
kymning och gryning går ihop
"
270 CLOSE #9 :: END
300 ! *****
310 MA=M :: DA=W
```

```

320 YZ=AA :: CALL JULIAN((AA
-1),(12),(31),JUL):: JA=JUL
:: CALL JULIAN((AA),MA,DA,JU
L):: JB=JUL
330 DAG=JB-JA :: CALL LAM(DA
G,LAM):: LAM1=LAM :: BET=0 :
: CALL ALFDEL(BET,LAM1,DEL,A
LF):: DEL1=DEL :: ALF1=ALF
340 LAM2=LAM+.985647 :: CALL
ALFDEL(BET,LAM2,DEL,ALF)::
DEL2=DEL :: ALF2=ALF
350 FI=GRAD+MI/60
360 CALL STID(DEL1,FI,ALF1,S
TR,STS,FL):: ST1R=STR :: ST1
S=STS
370 FK=FL
380 CALL STID(DEL2,FI,ALF2,S
TR,STS,FL):: ST2R=STR :: ST2
S=STS
390 FK=FL
400 TR=24.07*ST1R/(24.07+ST1
R-ST2R):: TS=24.07*ST1S/(24.
07+ST1S-ST2S)
410 DELM=(DEL1+DEL2)/2 :: CA
LL AZIM(DELM,FI,AR,AS,FL)
420 FK=FL
430 CALL KORR(FI,DELM,DX,DT,
FL):: AR=AR-DX :: AS=AS+DX :
: TR=TR-DT :: TS=TS+DT
440 LO=GRAL+MI/60
450 N=M+W/30 :: IF N>=3+22/3
0 AND N<=9+22/30 THEN 470
460 SOM=0 :: S$="" :: GOTO 4
80
470 SOM=1 :: S$="S"
480 GTR=TR-LO/15 :: GTS=TS-L
O/15 :: CALL B(JA,YZ,B7)
490 CALL UT(DAG,B7,GTR,UT)::
UTR=UT :: CALL UT(DAG,B7,GT
S,UT):: UTS=UT
500 UTR=UTR+1+SOM :: UTS=UTS
+1+SOM
510 CALL SKYMN(FI,DELM,TSK,F
L)
520 GRT=UTR-TSK :: SKT=UTS+T
SK :: CALL THM(GRT,THM$):: G
RT$=THM$ :: CALL THM(SKT,THM
$):: SKT$=THM$ :: CALL THM(T
SK,THM$):: TSK$=THM$
530 CALL THM(UTR,THM$):: UTR
$=THM$ :: CALL THM(UTS,THM$)
:: UT$=THM$ :: CALL THM(DELM,THM$):: DEK$=THM$
540 IF FK=1 THEN UTR$,UTS$="
***" :: X$=">"
550 IF FL=4 THEN GRT$,SKT$,T
SK$="***" :: X$=" <"
560 RETURN
10000 SUB JULIAN(AR,MA,DA,JU
L)!Julianska dagtal *****
10010 YB=AR :: IF MA>2 THEN
10030

```

```

10020 YB=YB-1 :: MA=MA+12
10030 AX=INT(YB/100):: BX=2-
AX+INT(AX/4):: CX=INT(365.25
*YB):: DX=INT(30.6001*(MA+1)
):: JUL=BX+CX+DX+DA+1720994.
5
10040 SUBEND
10050 !
10060 !
10070 SUB LAM(DAG,LAM)
10080 N2=360*DAG/365.2422
10090 IF N2>360 THEN N2=N2-3
60 ELSE 10110
10100 GOTO 10090
10110 IF N2<360 THEN N2=N2+3
60 ELSE 10130
10120 GOTO 10110
10130 M2=N2+278.833540-282.5
96403
10140 IF M2<0 THEN M2=M2+360
ELSE 10160
10150 GOTO 10140
10160 EC2=360*.016718*SIN(M2
*PI/180)/PI
10170 LAM=N2+EC2+278.833540
10180 IF LAM>360 THEN LAM=LA
M-360 ELSE 10200
10190 GOTO 10180
10200 SUBEND
10210 !
10220 !
10230 SUB ALFDEL(BET,LAM,DEL
,ALF)
10240 SD3=SIN(BET*PI/180)*CO
S(23.441884*PI/180)+COS(BET*
PI/180)*SIN(23.441884*PI/180
)*SIN(LAM*PI/180):: DEL=(180
/PI)*ATN(SD3/SQR(1-SD3*SD3))
10250 Y3=SIN(LAM*PI/180)*COS
(23.441884*PI/180)-TAN(BET*P
I/180)*SIN(23.441884*PI/180)
10260 X3=COS(LAM*PI/180):: A
3=Y3/X3 :: ALF=(180/PI)*ATN(
A3)
10270 IF (Y3>0)AND(X3>0)THEN
ALF=ALF
10280 IF (Y3>0)AND(X3<0)THEN
ALF=ALF+180
10290 IF (Y3<0)AND(X3<0)THEN
ALF=ALF+180
10300 IF (Y3<0)AND(X3>0)THEN
ALF=ALF+360
10310 ALF=ALF/15
10320 SUBEND
10330 !
10340 !
10350 SUB THM(DEC,THM$)!DEC-
->T:M
10360 H4=INT(DEC):: R4=DEC-H
4 :: M4=INT(R4*60):: THM$=ST
R$(H4)&":"&STR$(M4)
10370 SUBEND

```

```

10380 !
10390 !
10400 SUB AZIM(DEL,FI,AR,AS,
FL):: FL=0 !Azimuter *****
*
10410 CAR5=SIN(DEL*PI/180)/C
OS(FI*PI/180)
10420 IF CAR5>1+1E-8 OR CAR5
<-1+1E-8 THEN FL=1 :: SUBEXI
T
10430 AR=(180/PI)*(-ATN(CAR5
/SQR(1-CAR5*CAR5))+PI/2):: A
S=360-AR
10440 SUBEND
10450 !
10460 !
10470 SUB STID(DEL,FI,ALF,ST
R,STS,FL):: FL=0 !Stjarntid
*****
10480 CH5=-TAN(FI*PI/180)*TA
N(DEL*PI/180)
10490 IF CH5>1+1E-8 OR CH5<-
1+1E-8 THEN FL=2 :: SUBEXIT
10500 H5=(180/PI)*(-ATN(CH5/
SQR(1-CH5*CH5))+PI/2)/15
10510 STR=24+ALF-H5 :: IF ST
R>24 THEN STR=STR-24
10520 STS=ALF+H5 :: IF STS>2
4 THEN STS=STS-24
10530 SUBEND
10540 !
10550 !
10560 SUB KORR(FI,DEL,DX,DT,
FL) !*****
10570 !Summa korrektioner =
.835608333
10580 C6=SIN(FI*PI/180)/COS(
DEL*PI/180):: IF C6>1+1E-8 O
R C6<-1+1E-8 THEN FL=4 :: SU
BEXIT
10590 PSI=(180/PI)*(-ATN(C6/
SQR(1-C6*C6))+PI/2):: A6=TAN
(.835608333*PI/180)/TAN(PSI*
PI/180)
10600 DX=(180/PI)*ATN(A6/SQR
(1-A6*A6))
10610 S6=SIN(.835608333*PI/1
80)/SIN(PSI*PI/180):: Y6=(18
0/PI)*ATN(S6/SQR(1-S6*S6))::
DT=240*Y6/COS(DEL*PI/180)/3
600
10620 SUBEND
10630 !
10640 !
10650 SUB B(J,YZ,B7)
10660 T7=(J-2415020)/36525 :
: R7=6.6460656+2400.051262*T
7+.00002581*T7*T7
10670 B7=24-(R7-24*(YZ-1900)
)
10680 SUBEND
10690 !

```

```

10700 !
10710 SUB UT(DAG,B7,T,UT)
10720 A8=.0657098 :: C8=1.00
2738 :: D8=.99727
10730 T8=DAG*A8-B7 :: IF T8<
0 THEN T8=T8+24
10740 UT=T-T8 :: IF UT<0 THE
N UT=UT+24
10750 UT=UT*D8
10760 SUBEND
10770 !
10780 SUB SKYMN(FI,DEL,TSK,F
L):: FL=0
10790 CH9=-TAN(FI*PI/180)*TA
N(DEL*PI/180):: IF CH9>1+1E-
10 OR CH9<-1+1E-10 THEN FLAG
=1 :: SUBEXIT
10800 H9=(180/PI)*(-ATN(CH9/
SQR(1-CH9*CH9))+PI/2)
10810 CHS=(COS(96*PI/180)-SI
N(FI*PI/180)*SIN(DEL*PI/180)
)/COS(FI*PI/180)/COS(DEL*PI/
180)
10820 IF CHS<-1+1E-14 OR CHS
>1+1E-18 THEN FL=4 :: SUBEXI
T
10830 HS=(180/PI)*(-ATN(CHS/
SQR(1-CHS*CHS))+PI/2)
10840 TSK=(HS-H9)/15 :: TSK=
TSK*.9973
10850 SUBEND
10860 !
10870 !

100 ! VISASOL
110 ! 1991.01.12 EE
120 CALL CLEAR
130 PRINT "SOLENS UPP- OCH N
EDGANG"
140 INPUT "ORT,AR ":ORT$,AA
:: INPUT "LATITUD GG,MM ":G
RAD,MI :: INPUT "LONGITUD GG
,MM ":GRAL,MIL
150 INPUT "MANAD ? ":M
160 FOR W=1 TO 22 STEP 7 ::
GOSUB 300
170 PRINT W;TAB(5);GRT$;TAB(
10);UTR$;TAB(15);UTS$;TAB(21
);SKT$;TAB(5);TSK$;TAB(11);S
$;TAB(15);X$;TAB(18);"DEKL "
;DELM$ :: NEXT W
180 INPUT "FORTSATTA ? ":SVA
R$ :: IF SVAR$="J" THEN 150
190 INPUT "ANNAN ORT ? ":SV$
:: IF SV$="J" THEN 140
200 END
300 ! *****
310 MA=M :: DA=W
320 YZ=AA :: CALL JULIAN((AA
-1),(12),(31),JUL):: JA=JUL
:: CALL JULIAN((AA),MA,DA,JU

```

```

L):: JB=JUL
330 DAG=JB-JA :: CALL LAM(DA
G,LAM):: LAM1=LAM :: BET=0 :
: CALL ALFDEL(BET,LAM1,DEL,A
LF):: DEL1=DEL :: ALF1=ALF
340 LAM2=LAM+.985647 :: CALL
ALFDEL(BET,LAM2,DEL,ALF)::
DEL2=DEL :: ALF2=ALF
350 FI=GRAD+MI/60
360 CALL STID(DEL1,FI,ALF1,S
TR,STS,FL):: ST1R=STR :: ST1
S=STS
370 FK=FL
380 CALL STID(DEL2,FI,ALF2,S
TR,STS,FL):: ST2R=STR :: ST2
S=STS
390 FK=FL
400 TR=24.07*ST1R/(24.07+ST1
R-ST2R):: TS=24.07*ST1S/(24.
07+ST1S-ST2S)
410 DELM=(DEL1+DEL2)/2 :: CA
LL AZIM(DELM,FI,AR,AS,FL)
420 FK=FL
430 CALL KORR(FI,DELM,DX,DT,
FL):: AR=AR-DX :: AS=AS+DX :
: TR=TR-DT :: TS=TS+DT
440 LO=GRAL+MI/60
450 N=M+W/30 :: IF N>=3+22/3
0 AND N<=9+22/30 THEN 470
460 SOM=0 :: S$="" :: GOTO 4
80
470 SOM=1 :: S$="S"
480 GTR=TR-LO/15 :: GTS=TS-L
O/15 :: CALL B(JA,YZ,B7)
490 CALL UT(DAG,B7,GTR,UT)::
UTR=UT :: CALL UT(DAG,B7,GT
S,UT):: UTS=UT
500 UTR=UTR+1+SOM :: UTS=UTS
+1+SOM
510 CALL SKYMN(FI,DELM,TSK,F
L)
520 GRT=UTR-TSK :: SKT=UTS+T
SK :: CALL THM(GRT,THM$):: G
RT$=THM$ :: CALL THM(SKT,THM
$):: SKT$=THM$ :: CALL THM(T
SK,THM$):: TSK$=THM$
530 CALL THM(UTR,THM$):: UTR
$=THM$ :: CALL THM(UTS,THM$)
:: UTS$=THM$ :: CALL THM(DELM,THM$):: DELM$=THM$
540 IF FK=1 THEN UTR$,UTS$="
***" :: X$=">"
550 IF FL=4 THEN GRT$,SKT$,T
SK$="***" :: X$="<"
560 RETURN
10000 SUB JULIAN(AR,MA,DA,JU
L)!Julianska dagtal *****
10010 YB=AR :: IF MA>2 THEN
10030
10020 YB=YB-1 :: MA=MA+12
10030 AX=INT(YB/100):: BX=2-
AX+INT(AX/4):: CX=INT(365.25

```

```

*YB):: DX=INT(30.6001*(MA+1)
):: JUL=BX+CX+DX+DA+1720994.
5
10040 SUBEND
10050 !
10060 !
10070 SUB LAM(DAG,LAM)
10080 N2=360*DAG/365.2422
10090 IF N2>360 THEN N2=N2-3
60 ELSE 10110
10100 GOTO 10090
10110 IF N2<360 THEN N2=N2+3
60 ELSE 10130
10120 GOTO 10110
10130 M2=N2+278.833540-282.5
96403
10140 IF M2<0 THEN M2=M2+360
ELSE 10160
10150 GOTO 10140
10160 EC2=360*.016718*SIN(M2
*PI/180)/PI
10170 LAM=N2+EC2+278.833540
10180 IF LAM>360 THEN LAM=LA
M-360 ELSE 10200
10190 GOTO 10180
10200 SUBEND
10210 !
10220 !
10230 SUB ALFDEL(BET,LAM,DEL
,ALF)
10240 SD3=SIN(BET*PI/180)*CO
S(23.441884*PI/180)+COS(BET*
PI/180)*SIN(23.441884*PI/180
)*SIN(LAM*PI/180):: DEL=(180
/PI)*ATN(SD3/SQR(1-SD3*SD3))
10250 Y3=SIN(LAM*PI/180)*COS
(23.441884*PI/180)-TAN(BET*P
I/180)*SIN(23.441884*PI/180)
10260 X3=COS(LAM*PI/180):: A
3=Y3/X3 :: ALF=(180/PI)*ATN(
A3)
10270 IF (Y3>0)AND(X3>0)THEN
ALF=ALF
10280 IF (Y3>0)AND(X3<0)THEN
ALF=ALF+180
10290 IF (Y3<0)AND(X3<0)THEN
ALF=ALF+180
10300 IF (Y3<0)AND(X3>0)THEN
ALF=ALF+360
10310 ALF=ALF/15
10320 SUBEND
10330 !
10340 !
10350 SUB THM(DEC,THM$)!DEC-
->T:M
10360 H4=INT(DEC):: R4=DEC-H
4 :: M4=INT(R4*60):: THM$=ST
R$(H4)&":"&STR$(M4)
10370 SUBEND
10380 !
10390 !
10400 SUB AZIM(DEL,FI,AR,AS,

```

```

FL):: FL=0 !Azimuter *****
*
10410 CAR5=SIN(DEL*PI/180)/C
OS(FI*PI/180)
10420 IF CAR5>1+1E-8 OR CAR5
<-1+1E-8 THEN FL=1 :: SUBEXI
T
10430 AR=(180/PI)*(-ATN(CAR5
/SQR(1-CAR5*CAR5))+PI/2):: A
S=360-AR
10440 SUBEND
10450 !
10460 !
10470 SUB STID(DEL,FI,ALF,ST
R,STS,FL):: FL=0 !Stjarntid
*****
10480 CH5=-TAN(FI*PI/180)*TA
N(DEL*PI/180)
10490 IF CH5>1+1E-8 OR CH5<-
1+1E-8 THEN FL=2 :: SUBEXIT
10500 H5=(180/PI)*(-ATN(CH5/
SQR(1-CH5*CH5))+PI/2)/15
10510 STR=24+ALF-H5 :: IF ST
R>24 THEN STR=STR-24
10520 STS=ALF+H5 :: IF STS>2
4 THEN STS=STS-24
10530 SUBEND
10540 !
10550 !
10560 SUB KORR(FI,DEL,DX,DT,
FL) !*****
10570 !Summa korrektioner =
.835608333
10580 C6=SIN(FI*PI/180)/COS(
DEL*PI/180):: IF C6>1+1E-8 O
R C6<-1+1E-8 THEN FL=4 :: SU
BEXIT
10590 PSI=(180/PI)*(-ATN(C6/
SQR(1-C6*C6))+PI/2):: A6=TAN
(.835608333*PI/180)/TAN(PSI*
PI/180)
10600 DX=(180/PI)*ATN(A6/SQR
(1-A6*A6))
10610 S6=SIN(.835608333*PI/1
80)/SIN(PSI*PI/180):: Y6=(18
0/PI)*ATN(S6/SQR(1-S6*S6))::

```

```

DT=240*Y6/COS(DEL*PI/180)/3
600
10620 SUBEND
10630 !
10640 !
10650 SUB B(J,YZ,B7)
10660 T7=(J-2415020)/36525 :
: R7=6.6460656+2400.051262*T
7+.00002581*T7*T7
10670 B7=24-(R7-24*(YZ-1900)
)
10680 SUBEND
10690 !
10700 !
10710 SUB UT(DAG,B7,T,UT)
10720 A8=.0657098 :: C8=1.00
2738 :: D8=.99727
10730 T8=DAG*A8-B7 :: IF T8<
0 THEN T8=T8+24
10740 UT=T-T8 :: IF UT<0 THE
N UT=UT+24
10750 UT=UT*D8
10760 SUBEND
10770 !
10780 SUB SKYMN(FI,DEL,TSK,F
L):: FL=0
10790 CH9=-TAN(FI*PI/180)*TA
N(DEL*PI/180):: IF CH9>1+1E-
10 OR CH9<-1+1E-10 THEN FLAG
=1 :: SUBEXIT
10800 H9=(180/PI)*(-ATN(CH9/
SQR(1-CH9*CH9))+PI/2)
10810 CHS=(COS(96*PI/180)-SI
N(FI*PI/180)*SIN(DEL*PI/180)
)/COS(FI*PI/180)/COS(DEL*PI/
180)
10820 IF CHS<-1+1E-14 OR CHS
>1+1E-18 THEN FL=4 :: SUBEXI
T
10830 HS=(180/PI)*(-ATN(CHS/
SQR(1-CHS*CHS))+PI/2)
10840 TSK=(HS-H9)/15 :: TSK=
TSK*.9973
10850 SUBEND
10860 !
10870 !

```

DISK SECTOR EDITOR FW 4.30 DISK PATCH

(dropped from FW 4.31
in favour of Disk Review)

by Tony McGovern, Australia

This is based on DPATCH aka DISKO, issued by TI to user groups after orphaning the 99/4a. DP has had extensive modifications to make it easier to use. Key functions have been changed around a little, for better or for worse. <Ctrl-C>

parallels <Fctn-9> for some purposes. Both the original and DSKU (J. Birdwell) pattern are active. With Myarc FDControllers a read of sector #0 is forced so that 16/18 sectors/track can be distinguished.

If QD is already in memory when DP is loaded then a third choice for disk directory appears. This is just a bonus and its absence is NOT a bug. Exercise AID first if you want to make sure it appears. ■

TI-99/4A FRÅN GRUNDEN - 1

av Peter Walker, TI*MES, England

TANGENTBORD

Jag tänkte utforska olika tangentbord i detta avsnitt. Du känner säkert till CALL KEY kommandot som har följande syntax:

CALL KEY(tangentbord, returvariabel, statusvariabel)

Tangentbordet kan ha värdet 0 till 5 och det finns en beskrivning av hur en tangenttryckning översättes till returvariabel (returvariabeln blir -1 om ingen tangent har trycks ö.a.) i bruksanvisningen till TI-99/4A. När används de olika tangentborden? Jag hoppas att det följande ska hjälpa.

Tangentbord 1 och 2 används för delat tangentbord. Detta möjliggör att tangenter som trycks på den vänstra och högra sidan av tangentbordet registreras oberoende av varandra. Detta är nödvändigt för spel med två spelare.

Tangentbord 3 gör om TI-99/4A till det som användes av den tidigare TI-99/4 d.v.s. inga små bokstäver. Detta är ett bra tangentbord att använda när du vill göra saker som "PRESS Y/N". Med tangentbord 3 kommer Y och N att svara med värdena 89 och 78 oberoende av läget på ALPHA LOCK. Du behöver inte testa 121 och 110 (små bokstäver y och n). Inga CTRL-tangenter är aktiverade eftersom 99/4 saknade dessa.

Tangentbord 4 (känt som Pascal-tangentbordet) är avsett att användas för datakommunikationsprogram. När datorn samtalar med yttre datorer, förväntas ASCII-värdena 0 till 31 när kontrolltangenter trycks, t.ex. CTRL-Q är ASCII 17. Undvik tangentbord 4 för normala program eftersom det ändrar funktionen hos tangentbordet vid efterföljande INPUT-kommandon. Prova följande:

```
100 CALL KEY(4,K,V)
110 INPUT X
120 GOTO 100
```

På rad 110 kommer du att upptäcka att vänster och höger pil-tangent inte fungerar, du måste istället använda CTRL-H och CTRL-I. Inte heller FCTN-4 (BREAK) kommer att avbryta programmet, du måste istället använda CTRL-B. Detta kan vara besvärande så det är bäst att undvika tangentbord 4. Om du endast har grundmaskinen (konsolen) så tvivlar jag på att du använder datakommunikation!

Tangentbord 5 är det bästa allmän-giltiga tangentbordet när du vill ha både små och stora bokstäver. Skillnaden mellan detta och tangentbord 4 är att CTRL alstrar ASCII 128-159 medan de välkända FCTN-tangenterna AID till BACK är 1 till 15. INPUT-kommandot fungerar normalt.

Det finns ytterligare ett stort antal FCTN och CTRL tangenter som är aktiva vid tangentbord 4 och 5. Dessa är:

tangent	ASCII	tangent	ASCII
FCTN-V	127	CTRL-/	187
CTRL-0	176	FCTN-0	188
CTRL-1	177	FCTN-;	189
CTRL-2	178	FCTN-B	190
CTRL-3	179	FCTN-H	191
CTRL-4	180	FCTN-J	192
CTRL-5	181	FCTN-K	193
CTRL-6	182	FCTN-L	194
CTRL-7	183	FCTN-M	195
FCTN-,	184	FCTN-N	196
FCTN-.	185	FCTN-Q	197
FCTN-/	186	FCTN-Y	198

Tangentbord 4 har även CTRL-, som ger ASCII 0 medan tangentbord 5 ger 128 för detta.

Min äldre maskin svarar med 185 vid tryckning av FCTN-Q medan min nyare maskin ger 197. Observera att det finns ett fel i manualen eftersom tangentbord 4 och ENTER alltid ger 13 oberoende av CTRL och FCTN.

Till slut ska jag nämna att tangentbord 0 kommer att använda samma tangentbord (dock ej 1 eller 2 ö.a.) som senast användes. Detta är inte särskilt användbart om du inte väx-

lar tangentbord inne i ett program (många amerikanska program av t.ex. Regena har tangentbord 0 eftersom den ursprungliga 99/4 inte kan svälja siffran 3 för tangentbord ö.a.).

OMVANDLINGAR I BASIC

I detta avsnitt tänkte jag att vi skulle utforska de omvandlingsfunktioner som finns i Basic. I många program finns det behov av att omvandla en variabel från en form till en annan. Låt oss först titta på de enkla omvandlingsfunktionerna som finns i Basic.

STR\$

Detta omvandlar ett tal till motsvarande sträng (text). Jag har funnit detta användbart vid data-lagring av korta tal. Exempelvis kräver, utan hänsyn till längdbyten, tal i TI-99/4A 8 bytes utrymme medan strängmotsvarigheten endast behöver antalet ingående tecken. Talet 345 behöver 8 bytes utrymme medan textsträngen "345" endast behöver 3. STR\$ används även när du vill binda ihop ett antal ord och tal till en ny textsträng, talet måste först omvandlas till sträng:

```
A$="The result was "&STR$(N)&
" people from a total of "&STR$(M)
```

Observera mellanslaget efter 'was' och före 'people'. Till skillnad mot utskrift av tal har tal i strängform inga omgivande blanktecken.

VAL

Detta är motsatsen till STR\$. Det skapar ett tal av en textsträng med siffror. Ett exempel på dess användning kan vara följande:

```
100 INPUT "CAR REG NO?":A$
110 N$=SEG$(A$,2,3)
120 N=VAL(N$)
```

I detta exempel plockas siffrorna ut i strängform ur den blandade alfa-numeriska A\$ med bilens registreringsnummer. Den numeriska delen tas fram på rad 110 och omvandlas till

ett tal på rad 120.

ASC

Detta är ytterligare en användbar omvandlingsfunktion. Det omvandlar det första tecknet i en sträng till dess motsvarande ASCII-kod. En varning: ASC gillar inte att få en tom sträng att omvandla. Om detta kan inträffa bör du använda följande:

```
100 IF A$="" THEN 200
110 A=ASC(A$)
```

På detta sätt fångas tomma strängar in.

CHR\$

Detta är i själva verket motsatsen till ASC. Det omvandlar en ASCII-kod till motsvarande tecken i strängform. Här följer ett exempel med ASC och CHR\$ i användning i en rutin som omvandlar stora bokstäver i en INPUT-sats till små bokstäver.

```
100 INPUT "UPPERCASE CHARACTER":UC$
110 N=ASC(UC$)
120 LC$=CHR$(N+32)
```

TAL TILL NAMN

Nu till något mer invecklat. Du vill omvandla talen 1 till 13 till namnet på korten: "ACE", "TWO"... "TEN", "JACK", "QUEEN" och "KING". Detta är tillfället du kan använda en fältvariabel (array):

```
100 DIM CN$(13)
110 FOR C=1 TO 13
120 READ CN$(C)
130 NEXT C
140 DATA ACE,TWO,THREE,FOUR,FIVE,
    SIX,SEVEN,EIGHT,NINE,TEN
150 DATA JACK,QUEEN,KING
160 INPUT "NUMBER? ":C
170 PRINT CN$(C)
180 GOTO 160
```

Eftersom READ väntar sig strängar, behövs inte situationstecken "" i DATA-satsen.

Hur skulle vi göra med den omvända situationen? Vi har texten "ACE" eller "FOUR" och önskar översätta

detta till siffrorna 1 och 4. En möjlighet är att använda fältvariabeln som vi skapade ovan:

```
100 INPUT "CARD NAME ":N$
110 FOR C=1 TO 13
120 IF N$=CN$(C) THEN 160
130 NEXT C
140 PRINT "NO SUCH CARD"
150 GOTO 100
160 PRINT "CARD NO IS";C
```

Den enda nackdelen är snabbheten. Det tar en stund att jämföra varje namn med den presenterade texten. Finns det en snabbare metod? Svaret är ja. Vi kan använda POS-funktion som snabbt söker efter en sträng i en annan sträng. Vi lägger märke till att alla kortnamn är unika när det gäller de två första bokstäverna. Titta nu på detta:

```
100 B$="ACTWTHFOFISISEEINITEJAQUKI"
110 INPUT "CARD NAME ":N$
120 M$=SEG$(N$,1,2)
130 P=POS(B$,M$,1)
140 IF P=0 THEN 170
150 PRINT "CARD NO IS";(P+1)/2
160 ...
170 PRINT "NO SUCH CARD"
```

Denna användning av POS är mycket nyttig i ett stort antal tillämpningar. Om du skulle skriva ett program som omvandlar hexadecimala tal till decimala, skulle följande användning av POS ge det decimala värdet av varje tecken:

```
D=POS("123456789ABCDEF",HEX$,1)
```

SPEECH AV TAL

En speciell användning av att omvandla från nummer till namn finns med "Speech Synthesizer" (SS) när tal ska uttalas. Prova detta:

```
100 INPUT "NUMBER ":N
110 CALL SAY(STR$(N))
```

Detta fungerar bra för N=0-9. Större tal som t.ex. "34" kommer att uttalas "THREE", "FOUR" istället för "THIRTYFOUR". Följande rutin behövs om SS ska kunna uttala tal upp till 999.

```
100 DIM U$(19)
110 FOR J=0 TO 19 :: READ U$(J):: NEXT J
120 DATA "",ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE,TEN,ELEVEN,TWELVE,THIRTEEN,FOURTEEN,FIFTEEN
130 DATA SIX+TEEN,SEVEN+TEEN,EIGHT+TEEN,NINE+TEEN
140 FOR J=2 TO 9 :: READ TN$(J):: NEXT J
150 DATA TWENTY,THIRTY,FORTY,FIFTY,SIXTY,SEVENTY,EIGHTY,NINETY
160 CALL CLEAR
170 SAY$=""
180 INPUT "NUMBER ":N
190 IF N>999 THEN 180
200 IF N=0 THEN CALL SAY("0"):: GOTO 180
210 H=INT(N/100):: IF H>0 THEN SAY$=U$(H)+"HUNDRED"
220 N=N-H*100 :: IF N<>0 AND H<>0 THEN SAY$=SAY$+"AND"
230 TN=INT(N/10):: UN=N-TN*10
240 IF TN=0 THEN SAY$=SAY$+"&U$(UN):: GOTO 270
250 IF TN=1 THEN SAY$=SAY$+"&U$(UN+10):: GOTO 270
260 SAY$=SAY$+"&TN$(TN)+"&U$(UN)
270 CALL SAY(SAY$):: GOTO 170
0
```

HÖGERJUSTERADE TAL

```
90 REM av Anders Brodin
100 REM NITTINIAN 83-3.09
110 CALL CLEAR
120 T=10
130 HJT=666.777
140 GOSUB 1000
150 HJT=44.1
160 GOSUB 1000
170 HJT=1.5
180 GOSUB 1000
190 GOTO 190
1000 REM HÖGERJUSTERING
1010 IF POS(STR$(HJT),".",1) THEN 1040
1020 PRINT TAB(T-LEN(STR$(HJT)));STR$(HJT)
1030 RETURN
1040 PRINT TAB(T+1-POS(STR$(HJT),".",1));STR$(HJT)
1050 RETURN
```

BUBBLE - ASSEMBLERPROGRAM

av Jan Alexandersson

Du kan använda Funnelweb för att skriva in källkoden och sedan assemblera den. Gör så här:

1. Välj Central Menu nr 1 PROGRAM ED
2. Skriv in källkoden och använd TAB för att stega fram
3. Spara källkoden som BUBBLE/S
4. Välj Central Menu nr 2 ASSEMBLER
5. Tryck ENTER för källkodsfil
6. Tryck ENTER för objektkodsfil BUBBLE/O (FW skriver automatiskt O)
7. Tryck ENTER för ingen listfil
8. DELETE C så att endast OPTION R finns kvar och sedan ENTER
9. PROCEED för att assemblera
10. Välj Central Menu nr 3 LOADERS
11. Välj nr 4 LOAD/RUN av BUBBLE/O
12. ERASE av andra filnamnet
13. Låt markören peka på BUBBLE
14. PROCEED för att starta program
15. Bryt med QUIT

Istället för att göra en lång DIS/ VAR 80-fil kan du skriva in följande korta fil som BUBBLE/S:

```
DEF BUBBLE
DEF SFIRST,SLAST,SLOAD
SLOAD
SFIRST JMP BUBBLE
COPY "DSK2.MAIN"
COPY "DSK2.INIT"
COPY "DSK2.VDP-UTIL"
COPY "DSK2.GPLLNK"
SLAST END
```

Skriv sedan de fyra delarna MAIN, INIT, VDP-UTIL och GPLLNK till sina egna filer. Det är sedan ganska enkelt att återanvända dessa delar i andra program. BUBBLE-programmet är

skrivet så att det ska kunna laddas av alla moduler som klarar av att ladda DIS/FIX 80 uncompressed objektkodsfil. Du kan ladda det på följande olika sätt:

- Editor/Assembler Load and Run
- Mini Memory Load and Run
- Funnelweb Load and Run
- EA Basic CALL LOAD och CALL LINK
- MM Basic CALL LOAD och CALL LINK
- XB med CALL LOAD och CALL LINK

Följande Basic-program laddar från Extended Basic eller Basic med EA-modul eller MM-modul:

```
100 CALL INIT
110 CALL LOAD("DSK2.BUBBLE/O")
120 CALL LINK("BUBBLE")
```

Om du inte vill köra programmet från Basic eller XB kan du låta bli att ta med INIT, VDP-UTIL och GPLLNK. BUBBLE/S kan då se ut så här:

```
DEF BUBBLE
REF VMBW,VMBR,VSBW
COPY "DSK2.MAIN"
INIT RT
END
```

I detta fall används inbyggda rutiner i EA, MM och FW så att dessa bestäms till aktuell adress under laddningen. Om endast XB skall användas kan du skriva så här:

```
DEF BUBBLE
VMBW EQU >2024
VMBR EQU >202C
VSBW EQU >2020
VWTR EQU >2030
COPY "DSK2.MAIN"
INIT LI R0,>030E
BLWP @VWTR
LI R0,>0401
BLWP @VWTR
LI R0,>07F3
BLWP @VWTR
RT
END
```

```
*****
* från Editor/Assembler manual sid 342
* reviderat av Jan Alexandersson 1990-12-17
* kan brytas med QUIT
* kan laddas av godtycklig modul
* eller CALL LOAD från Basic (EA/MM) eller XB
*****
```

```
DEF BUBBLE, SFIRST, SLAST, SLOAD
```

```
SLOAD
```

```
SFIRST JMP BUBBLE          behövs om DIS/FIX 80 ska göras om till PROGRAM
```

```
***** MAIN START för huvudprogram
```

```
BBLE DATA >3C7E,>CFDF,>FFFF,>7E3C
```

```
COLOR DATA >F333
```

```
BBL BYTE >A0
```

```
SPACE BYTE >A8
```

```
EVEN          saknas i manualen, bör alltid finnas efter BYTE
```

```
LOC DATA >01DA,>020D,>0271,>02A5,>02D6,>02E1,>0000
```

```
MYREG BSS >20
```

```
* bestäm färgerna
```

```
BUBBLE LWPI MYREG
```

```
BL @INIT          initierar VDP motsvarande Editor/Assembler
```

```
LI R0,>394          färgtabell 20 och 21
```

```
LI R1,COLOR          ladda färgerna >F3 och >33
```

```
LI R2,2          två bytes att ladda
```

```
BLWP @VMBW          flytta till VDP-RAM
```

```
* bestäm teckendefinition
```

```
LI R0,>D00          plats för tecken >A0
```

```
LI R1,BBLE          bestämma bubblans tecken
```

```
LI R2,8          8 bytes ska flyttas
```

```
BLWP @VMBW
```

```
* rensa skärmen
```

```
CLR R0          börja med VDP-RAM >0000
```

```
MOVB @SPACE,R1          flytta blanktecknet
```

```
LOOP1 BLWP @VSBW          flytta ett tecken åt gången (LOOP1 fel i manual)
```

```
INC R0          pekar på nästa skärmposition
```

```
CI R0,>300          utanför skärmen
```

```
JNE LOOP1
```

```
* placera bubblorna på skärmen
```

```
MOVB @BBL,R1          ladda teckendefinition för bubbla
```

```
LI R2,LOC          ladda pekare till adress för bubbla
```

```
LOOP2 MOV *R2+,R0          ladda verklig adress
```

```
MOV R0,R0          kontrollera om laddning är klar
```

```
JEQ SCROLL          avslutat, börja skrolla skärmen
```

```
BLWP @VSBW          skriv bubbla på skärmen
```

```
JMP LOOP2
```

```
* skrolla skärmen
```

```
VDPBF1 BSS >20
```

```
VDPBF2 BSS >20
```

```
SCROLL CLR R0          ursprungsadress i VDP-RAM
```

```
LI R1,VDPBF1          buffertadress i CPU-RAM
```

```
LI R2,>20          antalet bytes som ska flyttas
```

```
BLWP @VMBW          flytta >20 från VDP-RAM
```

```

LI    R0,>20          VDP-adress >20
LI    R1,VDPBF2       CPU-buffertadress
LI    R2,>20          antalet bytes som ska flyttas

LOOP3  BLWP @VMBR      kopiera raden
AI     R0,->20         flytta till lägre VDP-minne (obs fel i manualen)
BLWP @VMBW            skriv tillbaka den undre raden
AI     R0,>40          läs nästa rad
CI     R0,>300         undersök om slutet av skärmen
JL     LOOP3          om inte så kopiera mer

LI     R0,>2E0         skriv sista raden
LI     R1,VDPBF1      CPU-buffert där första raden är
BLWP @VMBW            flytta CPU till VDP

LIMI 2               möjliggör interrupt så att QUIT kan bryta
LIMI 0

JMP  SCROLL          fortsätt skrolla
***** MAIN SLUT för huvudprogram *****

*****
* INITiering av VDP motsvarande EA
* av Jan Alexandersson 1986-08-17
* se PROGRAMBITEN 86-5
*****

FAC    EQU  >834A
STATUS EQU  >837C

STOR   DATA >0900
LITEN  DATA >0B00
CURSOR DATA >7070,>7070,>7070,>7070
        DATA >007E,>4242,>4242,>7E00
CRIGHT DATA >3C42,>99A1,>A199,>423C

INIT   LI    R0,>01A0
        BLWP @VWTR

        LI    R0,>0500
        MOVB R0,@>8374
        CLR  R0
        MOVB R0,@>837A

        LI    R1,>20D0
LOOPR  BLWP @VSBW
        INC  R0
        CI   R0,>0300
        JNE  LOOPR

        SWPB R1
        BLWP @VSBW

        LI    R0,>0380
        LI    R1,>1300
        LI    R2,>20
LOOPS  BLWP @VSBW
        INC  R0
        DEC  R2
        JNE  LOOPS

        LI    R0,>400
        CLR  R1
        LI    R2,>4F0
        LOOPT BLWP @VSBW
        INC  R0
        DEC  R2
        JNE  LOOPT

        MOV  @STOR,@FAC
        MOVB R1,@STATUS
        BLWP @GPLLNK
        DATA >18

        MOV  @LITEN,@FAC
        MOVB R1,@STATUS
        BLWP @GPLLNK
        DATA >4A

        LI    R0,>0C00
        LI    R2,>400
        LOOPU BLWP @VSBW
        INC  R0
        DEC  R2
        JNE  LOOPU

        LI    R0,>0850
        LI    R1,CRIGHT
        LI    R2,8
        BLWP @VMBW

        LI    R0,>08F0
        LI    R1,CURSOR
        LI    R2,16
        BLWP @VMBW

        LI    R0,>030E
        BLWP @VWTR
        LI    R0,>0401
        BLWP @VWTR
        LI    R0,>07F3

```

```

BLWP @VWTR
LI R0,>01E0
BLWP @VWTR

RT
***** INIT SLUT

*****
* VDP UTILITY disassemblerat *
* av J.Peter Hoddie 1987 *
* för GRAM Packer *
*****

VSBW DATA UTILWS,VSBWO
VMBW DATA UTILWS,VMBWO
VSBRO DATA UTILWS,VSBRO
VMBRO DATA UTILWS,VMBRO
VWTR DATA UTILWS,VWTR0

UTILWS BSS 32
AK BSS 2
AR EQU 2*2+1+UTILWS

VSBWO BL @AM
MOVB @>0002(R13),@>8C00
RTWP

VMBWO BL @AM
AN MOVB *R1+,@>8C00
DEC R2
JNE AN
RTWP

VSBRO BL @AO
MOVB @>8800,@>0002(R13)
RTWP

VMBRO BL @AO
AP MOVB @>8800,*R1+
DEC R2
JNE AP
RTWP

VWTR0 MOV *R13,R1
MOVB @>0001(R13),@>8C02
ORI R1,>8000
MOVB R1,@>8C02
RTWP

AM LI R1,>4000
JMP AQ
AO CLR R1
AQ MOV *R13,R2
MOVB @AR,@>8C02
SOC R1,R2
MOVB R2,@>8C02
MOV @>0002(R13),R1
MOV @>0004(R13),R2
B *R11

***** VDP UTILITY SLUT

```

```

*****
* GPLLNK *
* från Smart Programmer juli 1986 *
* av Craig Miller och Doug Warren *
*****

```

```

GPLWS EQU >83E0
GR4 EQU GPLWS+8
GR6 EQU GPLWS+12
STKPNT EQU >8373
LDGADD EQU >60
XTAB27 EQU >200E
GETSTK EQU >166C

GPLLNK DATA GLNKWS
DATA GLINK1

JMP RTNAD

BSS >20

RTNAD DATA XMLRTN
GXMLAD DATA >176C
DATA >50

GLNKWS EQU $->18
BSS >08

GLINK1 MOV *R11,@GR4
MOV *R14+,@GR6
MOV @XTAB27,R12
MOV R9,@XTAB27
LWPI GPLWS
BL *R4
MOV @GXMLAD,@>8302(R4)
INCT @STKPNT
B @LDGADD
XMLRTN MOV @GETSTK,R4
BL *R4
LWPI GLNKWS
MOV R12,@XTAB27
RTWP

```

```

***** GPLLNK SLUT

```

```

SLAST END

```

```

ÄNDRA MARKÖR I XB
(favorit i repris från PB 84-5.12)
5 REM CURSOR-UTSEENDE XB+EM
10 CALL CLEAR :: CALL INIT
20 CALL LOAD(8196,63,248)
30 CALL LOAD(16376,67,85,82,
83,79,82,48,8)
40 CALL LOAD(12288,0,0,0,0,0,
0,0,252)
50 CALL LOAD(12296,2,0,3,240,
2,1,48,0,2,2,0,8,4,32,32,36,
4,91)
60 CALL LINK("CURSOR")

```

BEGINNER ASSEMBLER - 2

USING SBUG TO DEBUG AL

by Mack McCormick, USA

This tutorial will describe how to use the TI SBUG program to debug your assembly language programs. Although specifically written for SBUG the principles apply to the other debuggers on the market. As I am sure you have already discovered, the familiar error messages and helps are not available from Assembler that you are accustomed to from BASIC. I know many of you have typed in my first tutorial or used one of the beginning assembly language books on the market and had difficulty getting the program to run properly. Take heart, we all had the same problems when we were learning and until you master the use of one of the debugging programs your you can pull your hair out and may eventually give up.

As I have already mentioned there are several very good debug programs available for the TI. Here they are briefly. DEBUG is the program supplied with your editor assembler cartridge. It has many helpful features particularly the ones that allow you to set CRU bits, move memory, and compare memory. My main criticism of DEBUG is its inability to single step through instructions. BUGOUT written by Gregg Wonderly is also very good. It offers multiple fields of information on the screen at the same time. Many advanced features like dumping to disk. Without a doubt the most advanced that I have used. My main criticism is that it maps the VDP to text mode which interferes with VDP mapping for the screen image table and color table if you need to inspect these locations in your program. The one I use the most is SBUG. This is the super debugger released by TI to users groups after the pull out. SBUG allows you to operate from graphics or bit map mode, single step, and output to printer or disk. Let's get down to how to use it.

First you should print out the instructions supplied with SBUG on the

disk. Since this is a display/variable 80 file you may print it out with TI-Writer or Editor/Assembler editor. This help file covers all of the instructions available. I intend to cover only the ones of greatest interest to us. It's important to know where your assembly program loads into memory. With some few exceptions which won't be discussed now, your program will load into high memory expansion at >A000. That's the first address in high memory. You can make your program load in other places but that is beyond the scope of this tutorial. SBUG is what we call relocatable object code. It loads on top (not over) of the last program entered. In other words, if your program is >500 bytes long it would load from >A000 to >A500. SBUG when loaded will load beginning at >A502.

I've included a brief program in this tutorial which puts characters on the screen. We'll use it to illustrate how to use SBUG. Go ahead at this time and type it in using the editor, save it, load the assembler except this time at the LIST FILE NAME prompt enter your printer description. Remember if you are using a PIO printer it should be entered with a period following PIO. At the OPTIONS prompt enter RLS. R to tell the assembler that you used R in front of your registers in the source code, L to give you a source listing to your printer, and S to give you a table of the symbols you used. This source list is necessary for using SBUG effectively.

Let's examine the source listing columns for a moment. Here's an example line from the source listing:

```
0049 006C 0201      LI  R1,>5500
      006E 5500
```

Here's what that tells us. 0049 means this statement is number 0049 in our source code. That means if we received the message invalid re-

gister in line 0049 when we were assembling the code we could look at line 0049 and see what was wrong with R1. The next field over is the most important one to us when using SBUG. This gives us the exact location in memory where this instruction resides. If this was the first program you loaded into memory then this instruction would reside at >A06C. Remember I told you that the program loads at >A000 by default? In other words >A000 plus >006C=>A06C. We'll come back to this field shortly. The next field >0201, is the machine language mnemonic for Load Immediate (LI). Look on page 5 of your blue editor/assembler card. See the opcode >0200? This tells us the instruction for this OPCODE is LI. The 1 in >0201 tells us to load immediate R1 with the value in the next word of memory. In this case Load Immediate R1 with >5500. See how easy that is? For a detailed description see section 15.4 in the E/A manual.

We're ready to get down to business. First load the object code for the program I have given you into the computer. You should **always** load your program before you load SBUG. Next before you press enter after you have loaded the demo program load SBUG. One note here. As long as you are not using the X/B cartridge to load with, it is much faster to load the compressed version of SBUG or SBUGC as it is listed on your disk. Both do the exact same job. Press enter and for the program name enter SBUG. The SBUG title screen should come up. The message on the screen will ask you if you are using a bit map screen? Enter N at this prompt. Next you will be asked for your list device. Enter your printer description here (you could enter DSK1.FILENAME instead if you wanted output to disk). The input prompt is a '.'. At the prompt enter L to turn off the list device. Our program runs from SBUG's control so we need to set up SBUG to run our program so we may interact with it. To do this we must tell SBUG where the first executable instruction (entry point) to our program is and what address we are using for a workspace. You may remember there are three hardware registers used by the

9900 CPU in the TI-99. These are the program counter, workspace register, and status register. The program counter always points to the next instruction to be executed in memory. The workspace register points to the current 32 bytes (16 words) of memory we are using for workspace. The status register contains the current status of the computer as a result of the last instruction executed. Given that, we now need to find the entry point to our program. There are two basic ways to do this. Perhaps the easiest is to look on the source listing at the label (in the example program START) we placed in the DEF statement at the beginning of the program and get the address directly from the source listing. The alternate way is to use the M command of SBUG to examine memory. The memory we need to examine is the Reference/Definition (REF/DEF) table approximately from >3F30 to >4000. This table lists the entry point to the program and the utilities we referenced. Here's how to examine this table.

M 3F30,3FFF <enter>

This command will scroll the contents of these memory locations to the screen. In this case we are looking for the program name which is START. When you see START press any key to stop the scroll. You should see a line of memory which looks like:

```
3F30=5354 4152 5420  START
3F36=A058 ..... 
```

Press FCTN X to abort the listing or any command. Since labels may be up to 6 characters (bytes) long, the seventh and eighth bytes contain the program entry point. In this case >A058. This should be the same address on the source listing. **Remember** SBUG always displays and only accepts HEX numbers.

Next we need the workspace address. We obtain that from the source listing. In this case it is >A05C. We get this by finding the label for our workspace, in this case WS. We look at the second column and there is the beginning of the workspace.

We're now ready to set up the hardware registers to run our program. We do this by entering a R at the prompt. SBUG responds with W=0000. W is the workspace address. In this case type A05C and press the <space bar>. That calls up the prompt P=0000 for program counter address which in this case is A058. If you press space again you could set the status register but this is unnecessary in this case so just press enter. We've now told SBUG where our program resides in memory and where the workspace is.

To make it easy to figure the offset of our instructions we have a bias command. Press X and the computer displays 0000 press A000 <enter>. This sets the value of X to >A000. Now let's get down to examining our program in memory. To do this we set breakpoints (just like BASIC) at the instruction where we want to stop. Ready to set one? At the prompt press B 68X <enter>. This automatically adds the A000 offset to the instruction at 0068. You could have just entered it as B A068 if you wanted to add the offset yourself. This tells SBUG to interrupt the program before it executes LI R0, >390. See where I got that? Look at the second column on your source listing and go down it until you find memory location 0068. Now if you want to see where your current breakpoints are press B <enter>. Now to execute the program up to that instruction press E <enter>. The screen will show three addresses across the screen. From left to right they are the workspace address, program counter address, and status register. Now we're ready to single step through some instructions. Enter S for single step at the prompt. This displays the following:

```
A068=0200  LI R00,>0390
```

A068 is the memory address for the instruction. Remember what 0200 is? LI R0. Then there is the plain text instruction. What value should be in R0? >0390 of course. To check this press W 0 <enter>. This shows the current contents of R0. If you ever have the need you may change this value by entering another value be-

fore pressing enter. If you press the space bar you see the next register R1 and so on. One important point. Never try to single step thru VDP, GROM, or KSCAN routines. There is a good chance you will lock up the computer. When you encounter one of these instructions just set a breakpoint (B) on the other side of it and execute (E) around it. Now press B 80X or B A080 <enter> at the prompt. This sets the breakpoint at the INC R0 instruction. Press E <enter>. What's the value in R0? It should be >383 from instruction >A082. Now press S. The INC R0 instruction appears. Again check the value in R0 (remember W0), it should be >384. See how you may interact directly with the computer? Press S again and you see the CI instruction. Press S again and you get:

```
JNE      $+>00FA
JMP TO A07C
```

This says to jump to A07C (LOOP2) if R3 is not equal to zero. Now set B A0CE. Press E <enter> to execute. The program stops at the MOV @SAVRTN,R11 instruction. Let's examine some VDP RAM. To do this we use the M command again but proceed the address with a V for VDP access. Press M V168,1AF. This shows the screen image table from decimal 360 to 431 where we put our text and ball. Examine the bytes carefully and you should see the border characters. Note: The screen is not exactly as we had it because SBUG shares the same screen with us and we're seeing some of SBUG's screen also. Now press M VBF8,C00 <enter>. This is the pattern descriptor table where the shape of our ball is stored. Compare it with PATRN from the source listing and then press M A022, A028 to see the pattern description as it resides in CPU RAM.

Well I've gone long again but there is so much to cover. You really need to get the hang of using SBUG, it allows you to interact with your program to see exactly what is going wrong when you have a problem. It will save you many hours of grief.

(Du kan även använda Edgar Dohmanns Superbug II version 2.1, som är en förbättring av SBUB, se PB 88-1)

```

*****
*
* This program accompanies the tutorial
* on using SBUG. It also shows you
* how to redefine characters, place
* them on the screen, and change the
* color of each character set
* Entry Point -> START
* Must be run from E/A or MM Load & Run
* R0-R2 Used for general VDP access
* R3 Used as a general purpose counter
* R11 contains the return address on
* entry.
* by Mack McCormick
*
*****

```

```

DEF START          DEFINE THE ENTRY POINT OF THE PROGRAM
REF VMBW, VSBW, VWTR SYSTEM UTILITIES WE WILL USE

```

```

* DATA STATEMENTS *

```

```

SAVRTN DATA 0          SET ASIDE A WORD OF MEMORY TO SAVE THE RETURN ADDRESS
BORDER DATA >8080,>2020,>2020,>2020 CHAR >80 WILL BE OUR BORDER
DATA >2020,>2020,>2020,>2020
DATA >2020,>2020,>2020,>2020
DATA >2020,>2020,>2020,>8080
PATRN DATA >3C7E,>FFFF,>FFFF,>7E3C

```

```

STATUS EQU >837C        LOCATION OF THE GPL STATUS BYTE
*                        THIS EQUATES THE LABEL STATUS TO THIS ADDRESS
WS      BSS >20          SET ASIDE 32 BYTES OF MEMORY FOR OUR WORKSPACE

```

```

MSG      TEXT 'THIS IS A TEST' 14 BYTES LONG
EVEN          FORCES THE PROGRAM COUNTER TO AN EVEN MEMORY ADDRESS.
* AS A GENERAL RULE ALWAYS USE EVEN AFTER THE LAST BYTE, BSS, OR TEXT OPCODE.

```

```

START MOV R11,@SAVRTN   MOV THE ADDR IN R11 TO THE WORD OF MEMORY AT SAVRTN
LWPI WS          TELL THE HARDWARE WORKSPACE REGISTER WHERE YOUR WS IS

```

```

*--PUT A BLUE BORDER AROUND THE SCREEN--*

```

```

LI R0,>0705          07 IS VDP REG 7 OR THE SCREEN BACKGROUND COLOR
BLWP @VWTR          05 IS THE BACKGROUND COLOR. (LT BLUE)
* THIS SETS THE TOP AND BOTTOM OF THE SCREEN TO LT BLUE.
* SEE SECTION 16.1 E/A MANUAL FOR MORE INFORMATION ON VWTR.

* NOW SET CHAR >80 (128 decimal) TO A BLUE ON BLUE SQUARE
LI R0,>390          THIS IS THE POS IN VDP RAM COLOR TABLE FOR CHARS >80->87
LI R1,>5500         BLUE FOREGROUND/BACKGROUND IN MSB OF R1
BLWP @VSBW          REMEMBER R0 ALWAYS IS THE ADDR IN VDP. R1 ALWAYS CPU.
* SEE SECTION 21.2.2 FOR MORE INFORMATION ON THE COLOR TABLE

```

```

* NOW LET'S MAKE ALL THE CHARACTER SETS BLACK ON WHITE

```

```

LI R0,>383          START WITH CHAR SET >18
LI R1,>1F00         1=BLACK FG, F=WHITE BKGND
LOOP2 BLWP @VSBW
INC R0             POINT TO NEXT VDP RAM COLOR TABLE ADDRESS
CI R0,>390          LAST COLOR TABLE ADDRESS (CHARS >78->7F)
JNE LOOP2

```

* NOW WRITE THE BORDER TO THE SCREEN AND CLEAR THE SCREEN AT THE SAME TIME

```

        LI    R3,24          24 ROWS TO WRITE
        CLR   R0             BEGINNING OF SCREEN IMAGE TABLE
        LI    R1,BORDER      ADDRESS OF ONE ROW OF BORDER DATA
        LI    R2,32          32 BYTES TO WRITE
LOOP    BLWP @VMBW           WRITE A ROW
        AI    R0,32          POINT TO BEGINNING OF NEXT ROW
        DEC   R3
        JNE   LOOP

```

* NOW WE'LL PUT THE MESSAGE ON THE SCREEN CENTERED

* REMEMBER THAT THE SCREEN IMAGE TABLE IS FROM 0 TO 767 IN VDP RAM

* TO DETERMINE THE ADDRESS FROM ROW AND COLUMN WE USE THE FOLLOWING FORMULA

* $ADDR = (ROW-1) * 32 + (COLUMN-1)$ IF WE WANT OUR MESSAGE AT ROW 12 COLUMN 9 THE VDP

* ADDRESS WOULD BE 360

```

        LI    R0,360         VDP ADDRESS IN SCREEN IMAGE TABLE
        LI    R1,MSG         ADDRESS OF THE DATA IN CPU RAM
        LI    R2,14          14 BYTES LONG
        BLWP @VMBW           WRITE IT TO THE SCREEN

```

* NOW SUPPOSE WE WANT TO MAKE CHAR >7F A BALL SHAPE AND PLACE IT UNDER THE TEXT

* THE PATTERN TABLE IN E/A IS LOCATED AT >0800. TO CALCULATE THE LOCATION OF A

* PARTICULAR CHARACTER MULTIPLY IT'S HEX VALUE BY 8 AND ADD THE RESULT TO >800.

```

        LI    R0,>0BF8       (>7F*8)+>800=>BF8
        LI    R1,PATRN       PATTERN TO DEFINE >81 TO FROM CPU RAM
        LI    R2,8            PATTERNS ALWAYS 8 BYTES
        BLWP @VMBW           >81 IS NOW THE SHAPE OF A BALL

        LI    R0,431         SCREEN IMAGE TABLE TWO ROWS BELOW TEXT CENTERED
        LI    R1,>7F00        WRITE >81 (BALL) TO THE SCREEN
        BLWP @VSBW           PUT IT UP

```

* EXPERIMENT WITH THIS PROGRAM UNTIL YOU ARE COMFORTABLE WITH VDP RAM ACCESS

```

        MOV   @SAVRTN,R11    RESTORE R11 TO THE ADDRESS YOU WISH TO RETURN
        LIMB 2               ENABLE INTERRUPTS SO QUIT KEY WILL WORK
        JMP   $              LOCK UP THE COMPUTER (SAME AS 100 GOTO 100)
        RT
        END

```

```

60 REM PERSPEKTIV
70 REM NITTINIAN 83-1.04
80 REM TIJDNINGEN, HOLLAND
90 REM AV JAN BARNIER 1982
100 CALL CLEAR
110 CALL SCREEN(5)
120 PRINT "          VAENTA EN ST
UND": : : : : : : : :
130 RANDOMIZE
140 A=26
150 FOR J=5 TO 17
160 CALL CHAR(J*8,"")
170 NEXT J
180 CALL CLEAR
190 FOR J=1 TO 12
200 A=A-2
210 FOR Z=J TO 25-J
220 CALL HCHAR(Z,4+J,40+(J-1)*8,A)
230 NEXT Z
240 NEXT J
250 C=INT(15*RND)+2
260 N=30*C+110
270 CALL SOUND(500,N,1,2*N,3)
280 CALL COLOR(2,1,C)
290 FOR J=2 TO 16
300 CALL COLOR(J,1,C)
310 NEXT J
320 GOTO 250

```

TIPS FROM TIGERCUB #41

Copyright 1986

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213
USA

I started this newsletter over 3 years ago, as a means of promoting my software business. It has never been a success for that purpose, but I have kept it going because of the many interesting newsletters that I have received in exchange, and the many friends that I have made around the world.

I know, from the editors' comments in many of your newsletters, that many of you are finding it difficult to finance a newsletter for your shrinking membership, and even more difficult to find the time, and the material to print. For a one-man user's group pretending to be a business which is getting very little business, it has become impossible. User group members have never been good customers for anyone's software, for reasons which you all know, and those who are remaining active in the TI world are wanting more sophisticated software than I have to offer.

Some of you have offered to subscribe to my Tips, but I just don't have the time to get involved in anything like that. I have had some other projects on the back burner for too long, and it's time I got to work on them - they can hardly turn out to be less profitable than trying to sell software!

I am NOT going out of business, and I am NOT releasing my programs to the public domain. I will continue to sell them, and will continue some classified advertising.

My heartfelt thanks to the many user group editors and officers who have tried in many ways to encourage and help me. Many thanks to those who have purchased my programs.

I will greatly miss your newsletters. I do hope to keep in contact with some of you. Perhaps now I can find time to browse in the TI sections of CompuServe or GENIE, and perhaps I will meet you there.

The answer to the challenge in the last Tips? For a clue, try -

DISPLAY AT(24,1):0 in Basic.

Still don't get it? In Basic, DISPLAY is the same as PRINT, but AT is not recognized, so the computer thinks you are telling it to print the variable AT(1,1) - which, being undefined, is 0 - and advance to the next line (the :) and print 0.

I have always wanted a pocket calculator with several memories and a window to display the contents of each one. So, since there is plenty of room for windows on a TV screen, I wrote one.

It does not require any use of the Enter key, but each CALL KEY input must be validated and processed, so don't type too fast. It will accept such inputs as M1=7= or M1=7+1= or M2=1-M1= to put a value in a memory, or 6+7= or 6+M2= to calculate and display, or 6+7M1 or M1-.M2M3 to calculate and put into memory, and will even do multiple calculations such as 1+2-3/4*5%6, subtotalling after the first two.

```
100 CALL CLEAR :: CALL SCREE
N(5):: DEF S$(X)=SEG$(A$,X,1)
)&" = " :: CALL PEEK(8198,A)
:: IF A<>170 THEN CALL INIT
110 CALL LOAD(-31806,16):: O
```

```

N WARNING NEXT :: GOTO 140
120 SET,M$( ),K,S,A$,S$( ),R,C
,N,N1,N2,N1F,N2F,M1F,M,MF,DF
,FF,VF,EF,FL,N$,F2,T,M2,MEM(
),ST,NX,ZF
130 CALL COLOR :: CALL CHAR
:: CALL KEY :: CALL SOUND !@
P-
140 FOR SET=0 TO 4 :: CALL C
OLOR(SET,16,1):: NEXT SET ::
FOR SET=5 TO 8 :: CALL COLO
R(SET,5,16):: NEXT SET :: CA
LL CHAR(64,"0")
150 FOR SET=9 TO 12 :: CALL
COLOR(SET,16,1):: NEXT SET
160 DISPLAY AT(1,10):"TIGERC
UB": "MULTIMEMORY@CALCULAT
OR": "MEMORY #1": "MEMORY
#2": "MEMORY #3": "MEMORY
#4": "MEMORY #5"
170 M$(1)="0123456789.+-%=
CXM" :: M$(2)="0123456789.AS
MDPECXM" :: DISPLAY AT(20,1)
:"use ?": "(1) symbols": "(2)
alpha characters"
180 CALL KEY(0,K,S):: IF S=0
OR K<49 OR K>50 THEN 180 ::
A$=M$(K-48)
190 DISPLAY AT(20,1):S$(12);
"add";TAB(16);S$(16);"percen
t" :: DISPLAY AT(21,1):S$(13
);"subtract";TAB(16);S$(17);
"equals"
200 DISPLAY AT(22,1):S$(14);
"multiply";TAB(16);S$(18);"c
ancel" :: DISPLAY AT(23,1):S
$(15);"divide by";TAB(16);S$
(19);"clear all"
210 DISPLAY AT(24,1):"M1 to
M5 = memories #1 to #5"
220 R=15 :: C=1 :: N,N1,N2,N
1F,N2F,M1F,M,MF,DF,FF,VF,EF,
FL,ZF=0 :: N$="" :: DISPLAY
AT(18,1):""
230 CALL KEY(3,K,S):: IF S<1
THEN 230 :: CALL SOUND(50,5
00,5):: DISPLAY AT(R,C):CHR$
(K):: C=C+1
240 ON POS(A$,CHR$(K),1)+1 G
OTO 260,270,270,270,270,270,
270,270,270,270,280,290,
250,290,290,290,340,410,420,
430
250 IF VF=1 OR MF=1 THEN 290
:: ZF=1 :: N$="-" :: GOTO 2
30
260 DISPLAY AT(R,C-1):"?" ::
C=C-1 :: GOTO 230
270 IF MF=1 THEN 260 :: FL=0
:: VF=1 :: IF DF=0 AND ZF=0
THEN N=N*10+K-48 :: GOTO 23
0 ELSE N$=N$&CHR$(K):: GOTO

```

```

230
280 IF DF=1 THEN 260 :: DF=1
:: MF,FL=0 :: IF ZF=1 THEN
N$=N$&". " :: GOTO 230 ELSE N
$=STR$(N)&". " :: GOTO 230
290 IF C=2 OR FL=1 THEN 260
:: FL=1 :: IF FF=0 THEN 320
300 F2=POS(A$,CHR$(K),1)-11
:: IF VF=1 THEN GOSUB 480
310 GOSUB 520 :: N1=T :: DIS
PLAY AT(18,1):"SUBTOTAL";T :
: N2F,N2=0 :: FF=F2 :: GOTO
230
320 IF VF=0 THEN 330 :: VF,M
F=0 :: GOSUB 480
330 MF=0 :: FF=POS(A$,CHR$(K
),1)-11 :: GOTO 230
340 IF C=2 OR (FF=0 AND M1F=0
)OR(C=4 AND M1F=0)OR FL=1 TH
EN 260
350 IF C=4 THEN EF=1 :: M2=M
:: N1F,MF=0 :: GOTO 230
360 IF VF=1 THEN GOSUB 480
370 IF EF=0 THEN 400
380 IF N2F=0 THEN MEM(M2)=N1
:: DISPLAY AT(M2*2+2,11):N1
:: GOTO 220
390 GOSUB 520 :: MEM(M2)=T :
: DISPLAY AT(M2*2+2,11):T ::
GOTO 220
400 GOSUB 520 :: DISPLAY AT(
15,C):T :: GOTO 220
410 DISPLAY AT(R,1):""::""::""
:: GOTO 220
420 MEM(1),MEM(2),MEM(3),MEM
(4),MEM(5)=0 :: FOR R=4 TO 1
2 STEP 2 :: DISPLAY AT(R,10)
:: GOTO 410
430 IF EF=1 AND MF=1 THEN 26
0
440 CALL KEY(3,K,ST):: IF ST
<1 OR K<49 OR K>53 THEN 430
ELSE CALL SOUND(50,500,5)::
M=K-48 :: DISPLAY AT(R,C):CH
R$(K):: C=C+1 :: MF=1 :: FL
=0 :: IF VF=1 THEN GOSUB 480
450 IF N1F=0 THEN M1F,N1F=1
:: N1=MEM(M):: IF ZF=1 OR DF
=1 THEN N1=VAL(N$&STR$(N1)):
: DF,ZF=0 :: GOTO 230 ELSE 2
30
460 IF N2F=0 THEN N2F=1 :: N
2=MEM(M):: IF ZF=1 OR DF=1 T
HEN N2=VAL(N$&STR$(N2)): DF
,ZF=0 :: GOTO 230 ELSE 230
470 GOSUB 520 :: MEM(M)=T ::
DISPLAY AT(M*2+2,11):T :: G
OTO 220
480 IF DF=0 AND ZF=0 THEN NX
=N ELSE NX=VAL(N$):: DF,ZF=0
490 IF N1F=0 THEN N1=NX :: N
1F=1 :: GOTO 510

```

```

500 N2=NX :: N2F=1
510 VF,N=0 :: N$="" :: RETURN
520 IF FF=1 THEN T=N1+N2 ELSE
IF FF=2 THEN T=N1-N2 ELSE
IF FF=3 THEN T=N1*N2 ELSE IF
FF=4 THEN T=N1/N2 ELSE T=N1
*N2/100
530 RETURN

```

I have always been annoyed by the difficulty of hyphenating with TI-Writer, when I want to avoid the gaping holes that wraparound and Fill and Adjust can cause. Manually filling and adjusting with carets is slow, and leaving a space after the hyphen is unreliable, so I wrote this program.

```

100 DISPLAY AT(2,10)ERASE ALL:
"TIGERCUB": " HYPHENATED FILL
AND ADJUST"
110 DISPLAY AT(6,1): " Prepare
text with TI-Writer": "Edit
or. Leave left TAB at 0,": "set
right TAB at the actual":
"value of the line length de-
"
120 DISPLAY AT(10,1): "sired
(i.e., for a 28-char": "line,
set it at 28)."
130 DISPLAY AT(12,1): " Indent
as desired. Center": "headings
as desired but be": "sure to
follow them with a": "line feed
(Enter). Hyphenate"
140 DISPLAY AT(16,1): "as desired
and follow the": "hyphen immedi-
ately with a": "line feed (Enter)."
150 ON ERROR 160 :: GOTO 170
160 ON ERROR 160 :: RETURN 170
170 DISPLAY AT(20,1): "INPUT
FILE? DSK" :: ACCEPT AT(20,1)
BEEP:F$ :: OPEN #1:"DSK"&F$,
INPUT
180 DISPLAY AT(22,1): "OUTPUT
FILE? DSK" :: ACCEPT AT(22,1)
BEEP:N$ :: OPEN #2:"DSK"&N$,
OUTPUT
190 DISPLAY AT(24,1): "LINE LENGTH?"
:: ACCEPT AT(24,1)VALIDATE(DIGIT):L
200 LF$=CHR$(13):: H$="-"&CHR$(13)
210 ON ERROR 210 :: GOTO 220
220 ON ERROR 210 :: RETURN 3

```

```

10
230 LINPUT #1:M$ :: IF M$="
" OR M$=LF$ OR M$="" OR ASC(
M$)>127 OR (LEN(M$)=L AND POS
(M$,LF$,1)=0) OR POS(M$," ",1)
)=0 THEN 310
240 IF POS(M$,LF$,1)<>0 AND
POS(M$,H$,1)=0 THEN 310
250 IF POS(M$,H$,1)<>0 THEN
M$=SEG$(M$,1,LEN(M$)-1)
260 IF LEN(M$)=L THEN 310
270 P=1
280 X=POS(M$," ",P):: IF X=P
THEN P=P+1 :: GOTO 280 ELSE
Y,P=X :: IF POS(M$," ",P)=0
OR P=L THEN 310
290 M$=SEG$(M$,1,X)&" "&SEG$(
M$,X+1,255):: IF LEN(M$)>L
THEN 310 ELSE P=X+2
300 X=POS(M$," ",P):: IF X=0
THEN P=Y :: GOTO 300 ELSE G
OTO 290
310 PRINT #2:M$ :: IF EOF(1)
<>1 THEN 230 ELSE CLOSE #1 :
: CLOSE #2

```

Here is one for the pre-schoolers -

```

100 CALL CLEAR :: CALL SCREE
N(14):: CALL COLOR(1,11,11,1
2,5,5):: DISPLAY AT(3,10): "S
EE-N-SAY": : "PRESS ANY KEY
" !by Jim Peterson based on
a routine by Michael Lyons
110 DIM E$(16),PAT$(16):: CA
LL CHAR(123,RPT$("F",16))
120 DATA " " " " " " {
" " {" " " { " " { {" " " { {
" " { {" " " { " " { {" " " { {
" " { {" " " { " " { {" " " { {
" " { {" " " {
130 FOR J=0 TO 15 :: READ PA
T$(J):: NEXT J
140 CALL KEY(0,K,S):: IF S=0
THEN 140
150 CALL CHARPAT(K,CP$):: FO
R X=1 TO 16 :: Y=ASC(SEG$(CP
$,X,1)):: E$(X)=PAT$(Y+(Y>57
)*7-48):: NEXT X :: IF K>96
AND K<123 THEN K=K-32
160 CALL CLEAR :: CALL SAY(C
HR$(K)):: FOR X=2 TO 16 STEP
2 :: DISPLAY AT(8+(X/2),12)
:E$(X-1);E$(X):: NEXT X
170 CALL SAY(CHR$(K)):: GOTO
140

```

And so, one more time

MEMORY FULL

Jim Peterson

HINTS, TIPS & ANSWERS (HTA) - TI-WRITER

from Bill Sponchia, Canada

1. Outdenting- This is the reverse of "indenting". It will allow first line of a paragraph to be start farther to the left then the remaining lines in the paragraph.

eg - ".LM12;IN7;RM71" causes the first line to start at column 7 and subsequent lines to begin at column 12.

2. When using REPLACE STRING you must be in the same MODE (Word wrap or Block) that the original document was produced in.

3. This is for use in Funnelweb Text Editor mode (note: it may work for regular TI-Writer but I'm not sure.) Want to convert the character case (Upper to Lower; Lower to Upper)?

i) Upper to Lower - depress CTRL and "." (period)

ii) Lower to Upper - depress CTRL and ";" (semi-colon)

By keeping the keys depressed the autorepeat function will take affect and every character the cursor passes over will be converted.

4. To save part of a document insert in front of the filename the first line number you wished saved then either a comma or a space and then the last line number you wished saved.

eg - 5,30 DSK1.MINUTES or 5 30 DSK1.MINUTES - will save the lines 5 to 30, inclusive, onto disk drive #1 under the name MINUTES.

5. To "get rid" of the line numbers on the left of the screen press FCTN 0 (zero). To get them back press FCTN 0 again.

6. If you erase a line in error, press CTRL 1 (known as "OOPS!") and your line will be restored. Note: This will work ONLY if no other keys were pressed between the erasing and "OOPS".

7. When using the "Search" command remember that the search is only from the point that the cursor is

located. Therefore to search the total document the cursor must be on line 1 before you go to the Command mode.

8. To backspace beyond the left margin press CTRL Y. This will temporarily disable the left margin. There is no right margin release.

9. When using the Header or Footer command with the page number it is possible to have NO value printed (such as for the introduction, etc) by using the .PA format command with a value of zero. The page numbering will begin on the following page. A .PA at the end of each page will delay the numbering further.

10. TI-Writer can save a file in other than the normal D/V80 format by using the PF command and either putting a "F" in front of the filename (ie F DSK1.MYFILE) or by putting a "C" in front of the filename. "F" causes a file to be created in a Display/Fixed 80 format. "C" strips any control characters from the file as it is sent.

11. TI-Writer can be used as a database. Each line must be a record and set up exactly the same. For example if the data was names, addresses and phone numbers then all names must start in the same column; all addresses must start in the same column and all phone numbers must start in the same column. There can be no lines which are blank or which have other type of information on them (ie - titles). Then using the program SORT UTILITY (by D R Romer & J Clulow) you can sort this file. Once sorted, which is done very quickly, titles can be added if you are printing it out.

12. There are CTRL keys equivalents to most FCTN keys, plus a few others. For example to tab to the right you can go FCTN 7 or CTRL W; to tab back (to the left) you can go

CTRL T (there is no equivalent FCTN key).

13. If you must go to the bottom of your text (and it is very long), instead of paging down simply go to the Command Line and press "S" for "show line" and at the prompt "enter line number" just type "E" and press ENTER. "E" is a valid line number for the last line (end) of a document. This feature is active in all the commands requiring you to enter a line number.

14. If you wish to prevent the form feed at the end of printing when using the Formatter then make the last line of your text ".PL 1". This will suppress the form feed, but note: do not forget to reset PL if you have another document to print.

15. You can string the formatter commands on the same line separated with a semicolon.
eg - LM 10;RM 70;IN +5;FI;AD

16. If you are having problems with formatter commands, make sure they are UPPERCASE letters.

17. To avoid a BUFFER FULL notice you just save the files as it gets larger, then use the SD command to see the file size. The BUFFER is full at 92 sectors.

18. When using the .CE command you MUST also use the .LM and .RM commands because .CE centres the text between the SET margins, not necessarily the middle of your paper.

19. The Formatter also ensures that you have two spaces after each period. To control this when you are typing such things as "Mrs. E Smith" or "1023 N. Queen Street" then use the "^" sign in place of the space after the period.
eg - Mrs.^E Smith; 1023 N.^Queen Street 20. If you must have a dot in column one of your text, transliterate it. ".TL 124:46" will allow FCTN A to print out a period.

21. To create a file without line feeds yet Formatted, do the following:

i) use the FORMATTER to print the text to disk
ii) go back to EDITOR and do a Print File (PF) replacing PIO with C DSKn.filename.

22. If you wish to include a program listing in your document instead of retyping it into TI-Writer just LIST the program to the diskette using the following command LIST "DSKn.filename".

This will save the program in DISPLAY/80 format which allows it to be read by TI-Writer. You can now load this file into TI-Writer and place the carriage return character at the end of each program line.

23. If you wish to place a Carriage Return at the end of a line of text (an line without the return will usually occur when you have inserted blank lines in the text and then put text on them) then place the cursor at the end of the text and press CTRL 8. This will place a Carriage Return where you want it and insert a blank line below. If this line is not wanted you can delete it with FCTN 3.

24. You can get a print out of your file WITH LINE NUMBERS when printing out of the Editor mode by placing an "L" and a space before the printer name in the command instruction.
eg - L PIO

This will eliminate the last 6 characters at the end of each line (#75 to #80) therefore keep your line lengths to a maximum of 74.

25. Did you know that you can type anything you want after a carriage return ON THE SAME LINE and it won't print out; but it will SAVE. This is great for text notes for screen reading.

26. When typing up a document which uses certain long words or phrases frequently then a time saver (and also added insurance against typing errors) is to type the words in shortform or initials (ie - TI-ARTIST could become TIA; Ottawa TI-99/4A Users Group could become OTIUG). When you are finished with the document use the Replace String function (RS) to change the words back to the full spelling (eg -

/TIA/TI-ARTIST/).

Care must be taken in three forms when using this:

- i) the search only begins from the spot that the cursor is on so to do the whole document insure that the cursor is on line 1 before starting.
- ii) the search will locate all occurrences of the string. Therefore if the string searched for is "at" it will find word "at" and also the "at" in "cat" and "that", etc. So before telling the machines to "Change" all occurrences automatically be sure this program can't arise. If you are not confident of this it is best to walk through and change each separately as it is found.
- iii) as a reformatting will be done wherever a change is made it would be wise to review the document after to be sure that it is still formatted correctly.

27. When using the FIND STRING command you can specify which column range to search.

eg - 5 15 /text/ will look for the string "text" in the columns 5 through to 15 inclusive.

28. If your printer does not have a slashed zero and you want to print it out that way use the following Transliteration: .TL 48:48,8,47 This will cause the normal zero (48) to be printed; then backspace (8); and then a printing of a slash (47).

29. Two files can be loaded into the Editor (assuming the total size is not too large for memory) by loading in the first file then doing a LF and entering E DSKn.YYY (where n= disk # and YYY=second file). This will load the second file after the end of the first file.

30. If you don't like the windowing when using the 80 column format then set the margins for 0 and 40 and turn off the line numbers (FCTN 0). When you are finished reset the left and right margins to what you desire and reformat each paragraph.

31. You can merge sections of a second file into the current document by the following entry using the LF command:

25 7 12 DSK1.YYY - This will load lines 7 to 12 (inclusive) from file YYY to the current document after line 25.

32. If you are using FUNNELWEB 4.1 or greater after you have the directory on the screen (using the SD command) you will be able to see how many lines are in a file by marking the file and then requesting (V)iew. The line count will be shown at the bottom of the screen. ■

MUSIC PREPROCESSOR

av Niklas Johansson

EMUSIC PREPROCESSOR är skrivet av Norm Sellers. Detta är ett mycket avancerat musikprogram, där man skriver Assembler musik i Extended Basic rader med hjälp av nya musikkommandon.

Ett exempel på hur man skriver in noter:

! N n1 n2 n3 n4

I detta program kan man använda 4 stycken toner och det betyder att den sista noten inte behöver vara en bruston.

Med på de två disketterna får man 13 demo låtar av (enligt mitt tycke!) mycket hög kvalitet. I demo låtarna visas hur man kombinerar de olika kommandona som finns, man kan även använda Ex-Basicens grafik och sprites i låtarna.

Till dessa program får man också det fristående programmet 'SYSTEX', där man kan överföra Ex-Basic maskinspråksprogram till vanliga XB-filer.

Jag anser att dessa två program är mycket köpvärda, eftersom de är oerhört bra!

BETYG: EMUSIC. +++++

Kul att jobba med. Lätt läst manual.

BETYG: SYSTEX. +++++

Tunn manual.

Kul att ha vissa filer översatta.

Manualerna finns med på disketterna och kan skrivas ut med hjälp av TI-writer. Programmet finns på två skivor i programbanken. ■

SORTERING I BASIC

från 99'er och Microdatorn

```

100 REM SORTERING BASIC
110 REM 99'ER 83-07&MIKRODAT
120 DIM A(500)
130 CALL CLEAR
140 INPUT "ANTAL NUMMER ":N
150 INPUT "ANTAL SIFFROR ":S
160 RANDOMIZE
170 PRINT : : " ** SORTERINGS
METOD **: "99'ER", "MIKRODATO
RN": "1 SELECTION", "6 BUBBL
E": "2 BUBBLE", "7 INSTICK"
180 PRINT "3 SHELL", "8 MOD.I
NSTICK": "4 HEAP", "9 URVAL": "
5 QUICK", ": SHELL": :
190 CALL KEY(3,KEY,STA)
200 IF (KEY<49)+(KEY>58) THEN
190
210 FOR I=1 TO N
220 A(I)=INT(RND*10^S)+1
230 PRINT A(I);
240 NEXT I
250 PRINT : :
260 CALL SOUND(50,440,0)
270 ON KEY-48 GOSUB 330,430,
540,700,980,1400,1540,1680,1
980,2150
280 CALL SOUND(50,440,0)
290 FOR I=1 TO N
300 PRINT A(I);
310 NEXT I
320 END
330 REM **SELECTION SORT**
340 FOR I=1 TO N-1
350 FOR J=I+1 TO N
360 IF A(I)<=A(J) THEN 400
370 CHANGE=A(I)
380 A(I)=A(J)
390 A(J)=CHANGE
400 NEXT J
410 NEXT I
420 RETURN
430 REM **BUBBLE SORT**
440 FLAG=0
450 FOR I=1 TO N-1
460 IF A(I)<=A(I+1) THEN 510
470 CHANGE=A(I)
480 A(I)=A(I+1)
490 A(I+1)=CHANGE
500 FLAG=1
510 NEXT I
520 IF FLAG=1 THEN 440
530 RETURN
540 REM **SHELL SORT**
550 GAP=N*1.5
560 GAP=INT(GAP/2)
570 IF GAP=0 THEN 690

```

```

580 FOR I=1 TO N-GAP
590 J=I
600 K=J+GAP
610 IF A(J)<=A(K) THEN 670
620 CHANGE=A(J)
630 A(J)=A(K)
640 A(K)=CHANGE
650 J=J-GAP
660 IF J>0 THEN 600
670 NEXT I
680 GOTO 560
690 RETURN
700 REM **HEAP SORT**
710 K=N
720 LL=INT(N/2)+1
730 IF LL=1 THEN 770
740 LL=LL-1
750 S=A(LL)
760 GOTO 830
770 S=A(K)
780 A(K)=A(1)
790 K=K-1
800 IF K>=1 THEN 830
810 A(I)=S
820 GOTO 970
830 J=LL
840 I=J
850 J=J+J
860 IF J<=K THEN 890
870 A(I)=S
880 GOTO 730
890 IF J>=K THEN 920
900 IF A(J)>=A(J+1) THEN 920
910 J=J+1
920 IF S<A(J) THEN 950
930 A(I)=S
940 GOTO 730
950 A(I)=A(J)
960 GOTO 840
970 RETURN
980 REM **QUICK SORT**
990 P=1
1000 L(P)=1
1010 R(P)=N
1020 IF P<=0 THEN 1390
1030 LB=L(P)
1040 RB=R(P)
1050 P=P-1
1060 IF RB<=LB THEN 1020
1070 I=LB
1080 J=RB
1090 T=A(I)
1100 IF J<1 THEN 1140
1110 IF T>=A(J) THEN 1140
1120 J=J-1
1130 GOTO 1100

```

```

1140 IF J>I THEN 1170
1150 A(I)=T
1160 GOTO 1290
1170 A(I)=A(J)
1180 I=I+1
1190 IF I>N THEN 1230
1200 IF A(I)>=T THEN 1230
1210 I=I+1
1220 GOTO 1190
1230 IF J<=I THEN 1270
1240 A(J)=A(I)
1250 J=J-1
1260 GOTO 1110
1270 A(J)=T
1280 I=J
1290 P=P+1
1300 IF I-LB>=RB-I THEN 1350
1310 L(P)=I+1
1320 R(P)=RB
1330 RB=I-1
1340 GOTO 1060
1350 L(P)=LB
1360 R(P)=I-1
1370 LB=I+1
1380 GOTO 1060
1390 RETURN
1400 REM BUBBEL MIKRODATORN
1410 J=1
1420 GOTO 1520
1430 K=N
1440 GOTO 1500
1450 IF A(K-1)<=A(K) THEN 1490
1460 T=A(K)
1470 A(K)=A(K-1)
1480 A(K-1)=T
1490 K=K-1
1500 IF K>J THEN 1450
1510 J=J+1
1520 IF J<N THEN 1430
1530 RETURN
1540 REM INSTICKSSORTERING
1550 J=1
1560 GOTO 1660
1570 K=J
1580 GOTO 1630
1590 T=A(K)
1600 A(K)=A(K+1)
1610 A(K+1)=T
1620 K=K-1
1630 IF K<1 THEN 1650
1640 IF A(K)>A(K+1) THEN 1590
1650 J=J+1
1660 IF J<N THEN 1570
1670 RETURN
1680 REM MODIFIERAD INSTICK
1690 K=1
1700 GOTO 1800
1710 X=A(K+1)
1720 IF X>=A(1) THEN 1760

```

```

1730 RR=1
1740 GOSUB 1920
1750 GOTO 1790
1760 IF X>=A(K) THEN 1790
1770 GOSUB 1820
1780 GOSUB 1920
1790 K=K+1
1800 IF K<N THEN 1710
1810 RETURN
1820 LL=1
1830 RR=K
1840 GOTO 1900
1850 M=INT((RR+LL)/2)
1860 IF X<A(M) THEN 1890
1870 LL=M
1880 GOTO 1900
1890 RR=M
1900 IF RR-LL>1 THEN 1850
1910 RETURN
1920 J=K
1930 A(J+1)=A(J)
1940 J=J-1
1950 IF J>=RR THEN 1930
1960 A(RR)=X
1970 RETURN
1980 REM URVALSSORTERING
1990 I=1
2000 GOTO 2130
2010 K=I
2020 T=A(I)
2030 J=I+1
2040 IF A(J)>=T THEN 2070
2050 T=A(J)
2060 K=J
2070 J=J+1
2080 IF J<=N THEN 2040
2090 T=A(K)
2100 A(K)=A(I)
2110 A(I)=T
2120 I=I+1
2130 IF I<N THEN 2010
2140 RETURN
2150 REM SHELL MIKRODATORN
2160 D=1023
2170 I=1
2180 GOTO 2280
2190 J=I
2200 GOTO 2250
2210 T=A(J)
2220 A(J)=A(J+D)
2230 A(J+D)=T
2240 J=J-D
2250 IF J<1 THEN 2270
2260 IF A(J)>A(J+D) THEN 2210
2270 I=I+1
2280 IF I+D<=N THEN 2190
2290 D=INT(D/2)
2300 IF D>=1 THEN 2170
2310 RETURN

```

The TI-99/4A now has GEN/DIR, the GENeric disk DiRectory.

A long needed MUST for anyone using disk files and programs.

GEN/DIR provides all generic information possible about files in a disk directory.

The Disk Directory part of GEN/DIR provides the following:

GEN/DIR will date/time stamp files that have no date/time already. This is compatible with GENEVE date/time stamps.

GEN/DIR allows you to key and save file descriptions for each file, and even for each entry in object files.

GEN/DIR provides directories in 3 forms: it writes a DV80 disk file, and a printer listing of the directory, and it gives a screen display of the directory.

GEN/DIR's generic information answers the following questions about disk files in the directory listing/file/display:

- * Are PROGRAM files BASIC or ASSEMBLY or DATA?
- * Are BASIC programs EXTENDED BASIC or TI BASIC and do they contain imbedded assembly code?
- * What memory locations are used in an assembly program file?
- * Are DV80 documentation files to be printed with the FORMATTER?
- * For object files, what are the entry names needed to execute the files; are they compressed or not; are they absolute or relocatable loaded; what memory is used; are there AUTOSTARTS; are there REFERENCES (these cannot be used under XB)?

Also GEN/DIR indicates what sectors are used for each file and shows how fragmented files are.

The Disk Cleanup part of GEN/DIR provides the following:

GEN/DIR indicates how many sectors are LOST (not used by any files, but marked in use), and quickly frees up all LOST sectors.

GEN/DIR indicates what sectors are BAD by making a "file of bad sectors" and marks these bad sectors in use.

GEN/DIR deletes files with bad sectors without freeing the bad sectors so the file may be re-saved to the same disk easily.

The program comes as two independent parts with a common menu screen.

The DISK DIRECTORY portion including the screen color/printer name options may be obtained for a \$20.00 donation.

The DISK CLEANUP portion to eliminate "lost" sectors, mark bad sectors, and delete files with bad sectors may be obtained for a \$15.00 donation.

The complete program may be obtained for a \$30.00 donation.

To obtain the program/s please indicate:

Part 1, directory for \$20.00 ----
Part 2, disk cleanup for \$15.00 ----
FULL Program \$30.00 ----

Please print clearly your Name and Mailing address below:

Please make checks or money orders payable to: C. N. Sellers

Mail requests to: Norm Sellers
15 Dorset Drive
Broomall, PA 19008
USA

I am distributing the program myself to insure that nobody gets stuck with "spent" money and nothing received" as often happens now in our fine computer's latter stages.

In response to many requests for additional music samples, I am making the music that I have coded available for a suggested donation of \$5.00 for each disk, or \$9.00 for a double sided floppy of music toward my efforts and expenses. The music itself is free. Of course, my original offer to "trade music disks that you have coded" still stands.

1. SONGSEZ-1: The Impossible Dream. The Anniversary Waltz.
Autumn Leaves. Beautiful Ohio.
Theme from Love Story. Wonderful One.
You Ain't Heard Nothing Yet.
Let's Sing Another Old-time Song.

AND AN EXTRA YOU CAN'T AFFORD TO BE WITHOUT:

SAVEXT, a BASIC program.

DESCRIPTION:

A BASIC Program to recover an Extended Basic program that is in memory when the system crashes. This is done by searching high memory for the program's line number table. When found, the program displays the CALL LOAD that may be entered to fully recover the program so it may be saved.

REQUIREMENTS:

You do need the Editor/Assembler and Extended Basic cartridges and 32K extended memory. NOTE: This program will not work if you have a special system that uses high memory in initializing Extended Basic or Assembler Editor menus such as automatic disk manager functions upon cold start.

Utilities

Extended Basic Programmers

The Animator

A complete frame-by-frame animation package for the TI-99/4A! *The Animator* is an extensive program that allows you to create multiple animation sequences one frame at a time, and then script them together in a complicated animation sequence. You can import TI-Artist instances as frames, and even save animation sequences so that they can be used in your Extended BASIC programs! *The Animator* is a complete animation editor for Extended BASIC programmers, as well as the animation buff. Includes example files and thorough documentation. By Brad Snyder. Requires Extended BASIC, 32K, Disk.

U01 - Disk - \$14.95 *NEW PRICE* Save 25%!

EZ-Keys Plus

EZ-Keys Plus is the ultimate Extended BASIC programmers environment! *EZ-Keys Plus* allows you to define keys to be "macros" - which can be short strings of text, long ones, and even large blocks of key commands and text! Unlike other macro programs, *EZ-Keys Plus* lets you put function and control key commands in your macros, and have them execute just like little programs! Macros can be up to 668 characters in size and even call each other! *EZ-Keys Plus* will also allow you to customize your XB environment - your screen colors, your cursor and even your character set! Set the program to automatically save your program as you write it. A built-in disk cataloger and screen-dump function make it even more useful and convenient! Even automatic check-sum is available. All "A's" from MICROpendium. An in-depth manual makes it easy to use. Requires TI Extended BASIC, 32K and Disk. By Harry Wilhelm.

U02 - Disk - \$14.95

Quick-Run

This is the ultimate speed-up utility for Extended BASIC! *Quick-Run* takes a "snap-shot" of a running program, and saves it to disk. Later, the "snap-shot" can be run and the program will begin exactly where it was when the "snap-shot" was taken! Eliminate long pre-scan times as well as long setup times forever. *Quick-Run* will actually eliminate the time needed for a running program to get around to doing what it is you need to do. *Quick-Run* is also a tool-box of routines for Extended BASIC programmers that will let you chain programs and have a shared data pool between them - no longer will you have to save variables between parts of a program! Requires Extended BASIC, 32K and Disk. By Travis Watford

U03 - Disk - \$9.95

ASGARD

Cartridge Enhancers

Beyond Video Chess

The ultimate companion to the *Video Chess* module from Texas Instruments Inc! *Beyond Video Chess* allows you to save and load your chess games to disk, print the game-board or listings of your moves to the printer, and even control the pieces with joysticks. Two people can even use the cartridge as a chess-board! Requires the *Video Chess* module, 32K, disk, and either Extended BASIC, Editor/Assembler or Mini-Memory, and either a widget, load-interrupt switch or Corcomp/Myarc disk controller. By Harry Wilhelm - author of *EZ-Keys Plus*.

U04 - Disk - \$9.95

TOD Editor

The only editor for making your own Tunnels of Doom games! This classic by John Behnke is fully documented, and menu-driven so it's easy to use. It has dozens of powerful options - use *TOD Editor* to change everything from the monsters, to the graphics, to the story and more! Requires 32K, Disk, Extended BASIC.

U05 - Disk - \$14.95

Graphics Support

Pix Pro

The "Super-Converter" to meet all your graphics needs! There are dozens of different graphics formats in use that together constitute a veritable Tower of Babel - many times pictures are available in one format but can't be found in another. Sometimes you need to use 2, even 3 different conversion programs to get a picture from one format to another - if at all. Now, for the first time ever, this simple-to-use utility gives all your programs access to the entire library of pictures available for the 99/4A and 9640! With *Pix Pro* you can convert pictures stored in *GRAPHX*, *TI-Artist* Picture, *Instance*, *RLE*, *Picasso*, *Page Pro*, *MacPaint* and other formats into most any of the above. Store a *MacPaint* or *Picasso* picture in *TI-Artist* Picture format, convert a *GRAPHX* screen to *Instance* format, go from *RLE* to *Picasso*, and do most any other conversion you can dream of, all easily and at Assembly speed! *Pix Pro* will even let you save a full-page *MacPaint* as a full-page *Page Pro*! Full color support is included! This is the essential tool for all desktop publishers! Requires 32K, Disk, Extended BASIC or Editor/Assembler. By Jim Reiss.

U06 - Disk - \$14.95

Disk Utilities

RAM*Boot

*RAM*Boot* is a setup utility and RAM-disk manager for Myarc RAM-disk owners that eliminates the tiresome setup process - it automatically partitions your RAM-disk, copies files to it, and runs any program you specify while you get coffee! *RAM*Boot* also installs itself on your RAM-disk so later you can have it save your work back to floppy for you. *RAM*Boot* is unprotected so you can have multiple set-up disks. Requires Extended BASIC, Disk and a Myarc 128K/256K/512K card. By Travis Watford.

U07 - Disk - \$9.95

HardMaster

HardMaster is the most capable disk sector editor for the Myarc HFDC! A sophisticated program with a powerful command-line interface, *HardMaster* is oriented towards helping you find your data as well as your files. It features a powerful editor that will let you edit a sector in hexadecimal or ASCII, a Quad-Editor that will let you edit 4 sectors at once, the ability to dump a range of sectors in both ASCII and hex to a printer, a directory function which lists the sectors a files header and data occupy, a powerful search function that will search by both a string or a hex number, a TREE function that gives a listing of all your directories and their associated files with sector information for each, the ability to dump the bitmap or sector allocation table to a printer, and much more! Even edit 1.44Mb floppies! If you have a hard-disk, this program will pay for itself the first time you repair a file, or recover a lost directory. Compatible with the Geneve and the TI-99/4A. Requires 32K, Editor/Assembler and a Myarc HFDC. By Colin Christensen.

U08 - Disk - \$14.95



Link

There are dozens of terminal emulator programs available for the 99/4A and the Geneve. Many of them offer support for multiple emulations or file transfer protocols. Some promise tremendous speed or efficiency. Some offer niceties such as disk managers or text editors. Only one terminal emulator works on a basic 99/4A console, and it won't even allow you to download programs or text files. *Link*, by Jim Reiss, changes all of that. This TE program in a cartridge supports baud rates up to 4800 baud, has a VT-52 emulation built-in, and supports full ASCII and XMODEM transfers to disk and cassette. It even supports a 32K memory expansion if it's available. Simple to use yet full-featured, *Link* is perfect if you want to set up a second system as a low-cost terminal, or if you only have a console based system and want to telecommunicate. This package requires a TI-99/4A with an RS232 port and a modem. Includes a free Compuserve subscription, a \$20 value.



U09 - Cartridge - \$29.95.

PrEditor

PrEditor is a "Programmer's Editor" for the TI-99/4A and Myarc Geneve 9640. Designed for people who write code in c99, Assembly, Fortran or other languages, *PrEditor* will let you have two files in memory at once and cut and paste between them, it uses block-oriented and not line-oriented functions (select a text block on disk, then move/copy/delete it), will let you view a file on disk, has instantaneous search & replace built in, supports variable windowing on 40 column displays, supports 80-columns on a Geneve or properly equipped 99/4A, and so on. *PrEditor* can even be permanently configured as to screen colors, memory usage, and even the key assignments. Uses standard DV-80 text files. Requires 32K, Disk, and either Editor/Assembler, TI-Writer or Extended BASIC. By Tom Bentley.

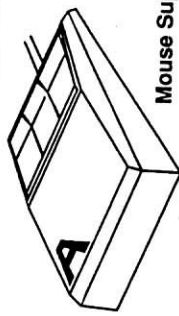
U10 - Disk - \$14.95 *NEW PRICE* Save 25%!



Batch It! 2.0

Batch It!, by Tom Bentley and Charles Earl, is a "batch" processing language that allows you to automate mundane, repetitive tasks performed by your computer. Really a miniature language complete with a compiler, *Batch It!* allows you to write programs to monitor what another program, and its users are doing, and act accordingly. You can use it to add additional functions to a program, combine multiple commands together (to create macros), etc. The new revised version is user extensible (programmers can add their own assembly language routines), features support for the Asgard Mouse (will allow you to add mouse support to a host of text-only programs), and includes a completely re-written manual with an extensive tutorial loaded with examples. Compatible with most 'program' type Assembly, c99 and Fortran 99 programs. Requires either a TI-99/4A with a disk system, 32K and the Editor/Assembler or Mini-Memory modules, or a Geneve 9640.

U11 - Disk - \$17.95 *NEW PRICE* Save 10%



Mouse Support

Classic Utilities

Pre-Scan It!

Pre-Scan It! takes any Extended BASIC program and inserts pre-scan commands - making it start more quickly. *Pre-Scan It!* also makes your Extended BASIC programs more efficient by using special tricks in the XB module. *Pre-Scan It!* even works with other versions of Extended BASIC! Requires 32K, Disk and TI Extended BASIC. By J. Peter Hoddie.

U12 - Disk - \$4.95 *NEW PRICE* Save 50%!

Page Pro 99

Page Pro 99 is a page-making program for the TI-99/4A and much more. With the utilities, fonts, pictures and templates described below it is a complete graphics system useful for banners, greeting cards, calendars, certificates, newsletters, flyers, letterheads, forms, etc. For more information see the *Page Pro Catalog 90-91*.

Page Pro 99 is the remarkable page making program for the TI-99/4A or Myarc Geneve 9640 that allows you to create an entire page, with graphics, lines and fonts, all at once on the screen exactly as it will appear on the page! *Page Pro 99* is fast, powerful, and even fun to use. Paste up to 28 pictures of any size and shape on the page at a keypress, type in any direction, draw in multiple linestyles, and even type in different fonts. Well tested and thoroughly documented. By Ed Johnson. Requires 32K, Extended BASIC or Editor/Assembler, a disk system, and an Epson or compatible printer. Required by all products below.

P01 - Disk - \$24.95

Page Pro Enhancements

Page Pro Banner Maker

Page Pro Banner Maker, by Chris Bobbitt, is the most professional banner maker available! Combine large, beautiful *Page Pro* pictures along with well formed text for banners that look smooth even close-up.

P02 - Disk - \$12.95

Page Pro Borders #1 & 2

A series of packages of pre-made borders for use with *Page Pro 99*, the *Page Pro Borders* packages are ideal for use in all of your graphics projects. By Mel Bragg and Shirley Slicer.

P03a or P03b - Disk - \$7.95/volume

Page Pro Sideways Printer

This utility, by Chris Bobbitt, allows you to print any *Page Pro* picture sideways on an Epson or compatible printer. Perfect for creating certificates, signs (up to 3 pages wide), and even sideways calendars quickly and easily. You can even create two-page newsletters ready to be sent to the printers!

P04 - Disk - \$12.95

Page Pro Fonts #1-3

Three volumes, each containing 25 original large and small fonts are now available in this series of fonts designed for use with *Page Pro 99*. By Paul Scheidmante.

P05a-c - Disk - \$7.95/volume

Page Pro Headline Maker

This utility allows you to create large, beautiful custom tiling for advertisements, newsletters, flyers or posters. Simply type a phrase and select a font and in a few seconds you'll have a large picture in *Page Pro 99* format containing the text you typed in the font you chose. By Paul Scheidmante.

P06 - Disk - \$9.95

Page Pro Headline Fonts

Each package in this two disk series contains 18 to 24 new fonts for use with the *Page Pro Headline Maker*. Requires *Page Pro Headline Maker*.

P07a or P07b - Disk - \$7.95/volume

Page Pro Pics

This popular series contains over 700 pictures for use with *Page Pro* divided by subject matter for ease of use. The following packages are currently available:

- | | | |
|--------------------|-----------------|--------------------|
| #1 - Animals 1 | #2 - Holidays 1 | #3 - Computer |
| #4 - Home/Misc. | #5 - People 1 | #6 - Publishing 1 |
| #7 - Animals 2 | #8 - People 2 | #9 - Christmas |
| #10 - Publishing 2 | #11 - Hebrew | #12 - Americana |
| #13 - Education | #14 - Nautical | #15 - Publishing 3 |

P08a-p - Disk - \$6.95 each or any 3 for \$17.95

Page Pro Poster Maker

Now you can create full-size posters and signs out of your *Page Pro 99* pages! This simple to use utility by Ed Johnson will blow up your page two times, four times and even eight times to create 17"x22" signs, 34"x44" posters and 68"x88" behemoths. Perfect for the student, teacher, business, or organization! *Asgard Mouse* optional.

P09 - Disk - \$12.95

Page Pro Templates

This outstanding collection of template disks contains over 80 different *Page Pro 99* pages along with all associated art-work and can be easily customized and printed for professional results the first time. The following packages by Virginia Davis and Mel Bragg are available:

- | | |
|-----------------------|-------------------------|
| #1 - Birthday Cards 1 | #2 - Birthday Cards 2 |
| #3 - Get Well Cards 1 | #4 - Get Well Cards 2 |
| #5 - Assorted Cards 1 | #6 - Assorted Cards 2 |
| #7 - Assorted Cards 3 | #8 - Invitations 1 |
| #9 - Envelopes 1 | #10 - Assorted Cards 4 |
| #11 - Envelopes 2 | #12 - Christmas Cards 1 |

P10a-1 - Disk - \$6.95 each or any 3 for \$17.95

Page Pro Utilities

Page Pro Utilities is a complete package of utilities for manipulating *Page Pro 99* pictures. It includes: *Page Pro Enlarger* - enlarge, reduce, stretch and squeeze your pictures to fit whatever space you have to fill; *Page Pro Stripper* - a utility for removing excess white space around your pictures; *Page Pro Flipper* - rotate your pictures 180 degrees, and mirror them left to right and vice versa; and finally, the *Page Pro Line Font Editor* - the best tool available for creating your own line fonts.

P11 - Disk - \$14.95

Screen Preview

A replacement for your *TI-Writer* (or compatible) Formatter, as the name suggests this program formats your document to the screen in a "greeked" format (the text is displayed as dots on a representation of the page). You can then check centering, margins, justification and so forth before printing it, or if you find errors make changes directly in your formatted text! Easy to use and well documented, this program will save you mounds of paper, and hours of frustration. Written by Joe Deleto. Requires a TI-99/4A or Myarc Geneve 9640, 32K, a disk system, a printer, and *TI-Writer* or a compatible.

P17 - Disk - \$12.95.

Artist Font Maker

This utility allows you to literally convert any picture into a character of a *TI-Artist* or *TI-Artist Plus* font. Now for the first time you can create a font from scratch within *TI-Artist* or in any other drawing program, and then load it into this program, and "dump" each letter to a font file of your creation. The beginning graphics user will appreciate the ease of use and the powerful features that make creating a new font simple. All users will appreciate the tremendous latitude the program allows in font generation. Requires 32K, disk and either the Editor/Assembler or Extended BASIC modules. By Jim Reiss and Asgard Software.

G13 - Disk - \$9.95

Sports Pics

An excellent collection of sports oriented pictures by Paul Scheidmante for use with *TI-Artist*, *TI-Artist Plus* or *Page Pro 99*. This collection is useful for the sports enthusiast, the teacher, the coach and the parent. Excellent for use in signs and logos. Requires either *TI-Artist* or *TI-Artist Plus* or *Page Pro 99*, 32K, and a disk system. Suggested retail

G14 - Disk - \$7.95

U.S. / Canada
please add
\$2.50 / order S&H
add 7%

Europe / Australia
Elsewhere add
\$7.50 / order S&H
for Airmail

Send
Orders
To: **Asgard Software**
P.O. Box 10306
Rockville, MD 20849

Castle Darkholm

An excellent two-part adventure for the Adventure module by Randy Cook, *Castle Darkholm* combines elements of gothic horror novels with the thrill of detective stories along with the raper action of dungeon adventures. You are entrusted by the Council of Elders with the task of removing the Evil Baron Manfred Ritter Von Darkholm before he slaughters all in the kingdom.

E11a - Cass. - \$9.95 or E11b - Disk - \$9.95

Rattlesnake Bend

A classic Western novel meets the Adventure module in this excellent new work by Mickey Schmitt. There's no law in the Wild West except an honest man and his gun, and your job is to comb Santa Diablo for the bandit Santiago Escondido, and bring him back to justice. No an easy game by any means, *Rattlesnake Bend* is nonetheless a satisfying shot of Red-Eye for the adventure enthusiast.

E12a - Cass. - \$7.95 or E12b - Disk - \$7.95

Waterworks

A 99 screen arcade game from David Bishop, *Waterworks* combines fast action with amazing digitized sounds to create a veritable feast for the arcade game fanatic. You've just been given the job of putting in the plumbing in the new 99 story building in your city. What an honor! While in principal it should be simple - you must connect pipe from each floor feed pipe to the faucets - it never is that easy. For one thing you are on a budget. For another, someone should have called the pest-control people first. A game for up to 10 players - dozens of small nuances make this a game that will be played over and over. Requires a TI-99/4A with joystick, 32K, disk and Extended BASIC or Editor/Assembler, or a Myarc Geneve 9640.

E07 - Disk - \$12.95.

Legends

This animated graphics adventure has been called the best of its kind for the TI-99/4A by every reviewer! Rated all "A's" by MICROpendium magazine, you must explore a vast island with dungeons, forests, swamps, mountains and inns. Along the way you fight monsters, cast spells, drink potions, talk to the inhabitants and generally have a good time! *Legends* is the most incredible adventure ever made for the 99/4A. By Donn Gramos and Ed Johnson. Requires Extended BASIC, 32K, Disk

E13 - Disk - \$22.95

Der

TI-99/4A User Club Berlin

lädt ein:

Internationales User-Treffen

13.9.-15.9.1991

in Berlin

Wiederum werden Neuigkeiten rund um den TI-99/4A im Mittelpunkt des Interesses stehen. Doch damit nicht genug. Neben dem Erfahrungsaustausch zwischen den Usern und einer umfangreichen Teile-Börse wird nicht nur für das leibliche Wohl und die Übernachtungsmöglichkeiten der Teilnehmer gesorgt werden. Zusätzlich wird es auch die Gelegenheit geben, Berlin von seiner kulturellen Seite kennenzulernen.

Damit der TI-99/4A User Club Berlin diesen umfangreichen Service optimal bewältigen kann, bitten wir um umgehende Anmeldung auf dem untenstehenden Abschnitt. Hinsichtlich der Hotelreservierung ist es unbedingt erforderlich, daß die Anmeldung sofort erfolgt. Vorab-Unterlagen zum Treffen gehen den Teilnehmern Anfang des Jahres 1991 zu.

Anmeldung

zum Internationalen TI-99/4A-User-Treffen 13.9.-15.9.1991 in Berlin

- ☐ Ich möchte mich zum User-Treffen anmelden und bitte um Übersendung der Unterlagen. Ich treffe am ____ .9.1991 in Berlin ein.
- ☐ Ich bitte um Unterbringung in einem Hotel. Die Kosten dürfen pro Person mindestens DM 40,- und höchstens DM ____,- betragen.
Es kommen weitere ____ Personen mit.
Die Unterbringung soll erfolgen im
☐ Mehrbettzimmer ☐ Doppelzimmer ☐ Einzelzimmer.
- ☐ Ich möchte neben meiner mitgebrachten TI-Anlage im Tagungsraum schlafen und bringe meine Liege und/oder Schlafsack mit (Kosten pro Nacht: DM 5-10).
- ☐ Ich möchte über kulturelle Angebote informiert werden.

Name und Adresse: _____
(in Druckbuchstaben)

Datum

Unterschrift

Diese Anmeldung bitte senden an:
Henry Hilsberg, Uhlandstr. 70, (W) 1000 Berlin 31