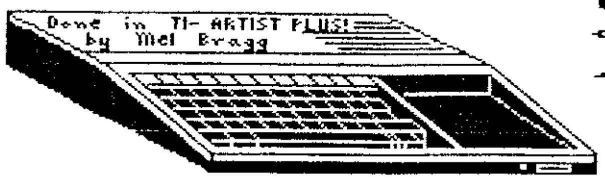
TI OGDENS USER GROUP

A publication of the TI 99/4 and 9640
Ogden users group Inc.

TI 99-4A

OGDENS USERS GROUP



Done in TI-ARTIST PLUS!
by Mel Bragg

THE CLUB THAT REFUSES TO SAY GOODBYE

NOVEMBER 1989

Converted to
PAGE PRO 99
by Mel

NO! CURLY you don't get any TURKEY DINNER.
You did'nt go to USERS GROUP MEETING !

TI-BASE - From INSCEBOT
TUTORIAL 11.1.1 By Martin Smoley
NorthCoast 99'ers - July 10, 1989

I am reserving the copyright on this material, but I will allow the copying of this material by anyone under the following conditions. (1) It must be copied in its entirety with no changes. (2) If it is retyped, credit must be given to myself and the NorthCoast 99ers, as above. (3) The last major condition is that there may not be any profit directly involved in the copying or transfer of this material. In other words, Clubs can use it in their newsletters and you can give a copy to your friend as long as its free.

```
*         Reverse-Print File  RPF1/C
CLOSE ALL
LOCAL SEETOP N 3
LOCAL LINE C 75
  SET RECNUM OFF
  SET SPACES=2
 PRINT (E)
USE LSTFRST
 TOP
 REPLACE SEETOP WITH ANYTHING
 BOTTOM
 PRINT
  SET HEADING OFF
 MOVE -1
 WHILE (ANYTHING<>SEETOP)
 REPLACE LINE WITH ANYTHING ¦ "            ";
 ¦ LAND_DEAL ¦ "  " ¦ COST ¦ "  ";
 ¦ SALEPRICE ¦ "  " ¦ PROF'LOSS
  PRINT LINE
  MOVE -1
 ENDWHILE
 REPLACE LINE WITH ANYTHING ¦ "            ";
 ¦ LAND_DEAL ¦ "  " ¦ COST ¦ "  ";
 ¦ SALEPRICE ¦ "  " ¦ PROF'LOSS
  PRINT LINE
  SET RECNUM ON
  SET HEADING ON
 CLOSE ALL
RETURN
```

```
DB   LSTFRST         Sort off
REC  ANYTHING  LAND_DEAL   COST          SALEPRICE      PROF'LOSS
0000   1       Parcel#28    34456.00      39671.00       5215.00
0001   2       Parcel#84    74246.00      91342.00      17096.00
0002   6       Parcel#21    29876.00      28235.00      -1641.00
0003  44       Parcel#18   123965.00     189913.00      65948.00
0004  99       Parcel#237   49156.00      63945.00      14789.00
0005  33       Parcel#84    44232.00      89491.00      45259.00
```

```
DB   LSTFRST         Sort on PROF'LOSS
REC  ANYTHING  LAND_DEAL   COST          SALEPRICE      PROF'LOSS
0002   6       Parcel#21    29876.00      28235.00      -1641.00
0000   1       Parcel#28    34456.00      39671.00       5215.00
0004  99       Parcel#237   49156.00      63945.00      14789.00
0001   2       Parcel#84    74246.00      91342.00      17096.00
0005  33       Parcel#84    44232.00      89491.00      45259.00
0003  44       Parcel#18   123965.00     189913.00      65948.00
```

```
Printout of RPF1/C or RPF2/C
ANYTHING  LAND_DEAL   COST            SALEPRICE       PROF'LOSS

  44      Parcel#18     123965.00      189913.00       65948.00
  33      Parcel#84      44232.00       89491.00       45259.00
   2      Parcel#84      74246.00       91342.00       17096.00
  99      Parcel#237     49156.00       63945.00       14789.00
   1      Parcel#28      34456.00       39671.00        5215.00
   6      Parcel#21      29876.00       28235.00       -1641.00
```

```
* Reverse-Print File II   RPF2/C
CLOSE ALL
LOCAL SEETOP N 3
  REPLACE SEETOP WITH -1
LOCAL SAVETOP N 3
LOCAL LINE C 75
  SET RECNUM OFF
  SET SPACES=2
 PRINT (E)
USE LSTFRST
 TOP
 REPLACE SAVETOP WITH ANYTHING
 REPLACE ANYTHING WITH SEETOP
 BOTTOM
 PRINT
  SET HEADING OFF
 MOVE -1
 WHILE (ANYTHING<>SEETOP)
 REPLACE LINE WITH ANYTHING ¦ "            ";
 ¦ LAND_DEAL ¦ "  " ¦ COST ¦ "  ";
 ¦ SALEPRICE ¦ "  " ¦ PROF'LOSS
  PRINT LINE
  MOVE -1
 ENDWHILE
 REPLACE ANYTHING WITH SAVETOP
 REPLACE LINE WITH ANYTHING ¦ "            ";
 ¦ LAND_DEAL ¦ "  " ¦ COST ¦ "  ";
 ¦ SALEPRICE ¦ "  " ¦ PROF'LOSS
  PRINT LINE
  SET RECNUM ON
  SET HEADING ON
 CLOSE ALL
RETURN
```

I received a question asking how to print down a page in descending order when TIB sorts everything in assending order? This is a quick demo of reverse file printing. The size and shape of the DB doesn't matter, and the sorted item can be numbers or names, the CF will still work. RPF1 goes to the TOP of the file and saves a unique item which it will look for as it moves back up the file. Then it goes to the BOTTOM of the file, prints a record, moves up one record, prints that record, etc. until it finds the record it has saved from the top of the file, where it stops. This theory works fine if you have a field with unique (one of a kind) items. If not, RPF2 gives you an idea on how to handle that problem. It goes to the TOP of the file and saves whatever it finds there. It replaces the item with an item I know is unique and then procedes in the same manner as RPF1. When it finds the top of the file it replaces the item it switched earlier with the original item, prints that record and stops. If this is a little confusing remember, I am always looking for more questions. So write me and ask.

Continued Next Month.

INSTANCE CONVERTER

by WESLEY R. RICHARDSON
NORTHCOAST 99ERS, CLEVELAND, OH, JUNE, 1989

INSTANCE-X is an Extended BASIC program which converts TI-ARTIST instances to a file suitable for TI-BASE version 2.01 to use as a database. Martin Smoley demonstrated the method of using TI-BASE for printing graphics in his article in the May, 1989 issue of the CLEVELAND AREA TI-99/4A USER GROUPS NEWSLETTER. He also had the idea of converting standard graphic data from other programs into TI-BASE. With the help of Deanna Sheridan, who suggested ARTIST instances as an extensive source for graphics, TI-BASE can now import and print graphics.

FILENAME	TYPE		DESCRIPTION
INSTANCE-X	PROGRAM		Extended BASIC program
INSTANCE/S	DIS/VAR	80	Assembly source code
INSTANCE/O	DIS/FIX	80	Assembly object code
INSTANCE/C	DIS/VAR	80	TI-BASE command file
TI-WORLD_I	DIS/VAR	80	TI-ARTIST instance
TI-WORLD_H	DIS/FIX	255	Hex code instance data
TI-WORLD/D	INT/FIX	254	TI-BASE instance data
TI-WORLD/S	INT/FIX	255	TI-BASE structure file

The source code listing INSTANCE/S is for the assembly routines which are called from the Extended BASIC program. To assemble this program, type in the INSTANCE/S file and save it. Then assemble the file giving INSTANCE/S as the source code file and INSTANCE/O as the object code file. Use only the R option because BASIC cannot access object files which use the C compress option.

When running the INSTANCE-X program, it will look for INSTANCE/O on drive 1. After loading, select option 1) to print an instance directly to the printer. Give the input file such as DSK1.TI-WORLD_I and the printer name PIO.CR.LF or the equivalent. Selecting option 2) will convert an instance to hex code for use by TI-BASE. Give an input file name like DSK1.TI-WORLD_I and an output file like DSK1.TI-WORLD_H. Option 3) is for a brief information summary, and option 4) is to end.

Load TI-BASE 2.01 and enter the date in the form MM/DD/YY, for example 05/26/89. For the following instructions, the notation (ENTER) will indicate to press the Enter key. Once in TI-BASE, type SET DATDISK DSK2. (ENTER). If you are using only one drive, the files listed above will need to be on your TI-BASE disk, and substitute DSK1 for DSK2 in these instructions. With your hex instance file, TI-WORLD_H in drive 2, type CONVERT TI-WORLD_H TI-WORLD_GO (ENTER).
Type HEXCODE (ENTER) under "descriptor", X (ENTER) under "type" and 254 (FCTN 8) under "width". The disk drive should run for a while, and then you will get the "." prompt. Type CATALOG DSK2. (ENTER) to confirm that files TI-WORLD/D and TI-WORLD/S were created. Type USE TI-WORLD (ENTER). Then type RECOVER (ENTER). Finally, type CLOSE (ENTER). If you have made it this far, you now have the instance in a database form which can be manipulated.

To print the instance, we must create a command file to tell TI-BASE what to print. Type MODIFY COMMAND INSTANCE (ENTER). When the screen clears, type in the file shown as INSTANCE/C in this article. Press (FCTN 8) when you are done to save the file and return to the command mode. With your printer turned on, type DO INSTANCE (ENTER) and the instance will print on your printer.

Instances up to 14 character positions wide, and any length up to 32 character rows long may be printed with this method. There is one problem, however. If the instance which you are printing has a long row of blank spaces, TI-BASE may not send these to the printer, and the printer will seem to print garbage, beep, eject paper, and so forth. If this happens with the particular instance you are printing, then you must add a few pixel positions (dots) which are printed to act as place holders.

NOTE: DO THE FOLLOWING STEPS ONLY IF YOU HAD PROBLEMS AT THE PRINTING FROM TI-BASE PROCEDURE.
If that is the problem for your instance, then do the following, type USE TI-WORLD (ENTER). Type MODIFY STRUCTURE (ENTER). With the cursor on HEXCODE, press (ENTER) and change the "type" from X to C, then press (FCTN 8). Type SET HEADING ON (ENTER). Type SET RECNUM ON (ENTER). Then type PRINT ALL (ENTER). Whe the printer stops, type MODIFY STRUCTURE (ENTER). And press (ENTER) to move the cursor from the word HEXCODE to the "type" C. Press X and (FCTN 8).

If you examine the printout of the hex codes you will find a 20 followed by 00's near the end of each record. The 00's after the 20 are ignored, however long sequences of 00's prior or to the left of that 20 can be removed by TI-BASE when it is sending to the printer. Type EDIT (ENTER) and then use (FCTN 5) to go to higher record number or (FCTN 6) to go to lower record numbers. When you have located the record with a long string of 00's to the left of the final 20, use the (FCTN S) or (FCTN D) to put the cursor on a 0 in the middle of a string of 00's. Change a 0 to a 1 and then press (FCTN 8) to keep the change and move the the next record. Do this for each record with a long string of 00's. After each record is changed, press (FCTN 8) and when you are done with all of the records, press (FCTN 9) to return to command mode. Type CLOSE (ENTER) to close the database.

Type DO INSTANCE (ENTER) to print the instance to your printer. When you are finished with TI-BASE, type QUIT (ENTER) to return to the TI title screen.

The more that I have used TI-BASE, the more impressed that I am with the power and flexibility of this program.

If you wish to get a copy of the files listed previously, send either a disk (SSSD or DSSD) with some of your favorite programs or copies of your club's last three newsletters to me at 27440 Pergl Road, Solon, OH 44139. I cannot send you any portions of TI-BASE.

----------------------------------890526WR-----

filename: INSTANCE/C

```
* INSTANCE/C
* TI-BASE INSTANCE PRINTER
* WESLEY R. RICHARDSON  MAY, 1989
* NORTHCOAST 99ERS, CLEVELAND, OH
* CONVERT FILE USING FORMAT:
* FIELD  DESCRIPTOR  TYPE  WIDTH
*    1   HEXCODE       X    254
SET TALK OFF
SET RECNUM OFF
SET HEADING OFF
USE TI-WORLD
WHILE .NOT. (EOF)
PRINT HEXCODE (LF) (CR)
MOVE
ENDWHILE
PRINT (LF) (LF) (CR)
CLOSE
RETURN
```

----

filename: INSTANCE-X

```
100 REM INSTANCE-X V 1.Ø TI-ARTIST INST
    ANCES CONVERTER
110 REM WESLEY R. RICHARDSON MAY, 1989
120 REM TI-99/4A EXTENDED BASIC
130 REM NORTHCOAST 99ERS, CLEVELAND, OH
140 DIM A(8),A$(8),C$(32,32)
150 GOTO 17Ø :: CALL CLEAR :: CALL INIT
     :: CALL KEY :: CALL LINK :: CALL L
    OAD :: CALL SCREEN :: CALL SOUND
160 B$ :: D$ :: E$ :: I :: J :: K :: P$
     :: W :: W$ :: X :: X$ :: Y :: Y$
170 !@P-
180 D$="DSK1.INPUT_I" :: P$="PIO.CR.LF"
190 CALL CLEAR :: CALL INIT
200 DISPLAY AT(6,4):"LOADING DSK1.INSTA
    NCE/O" :: CALL LOAD("DSK1.INSTANCE/
    O")
210 REM MAIN MENU
220 ON ERROR 8ØØ
230 CALL CLEAR :: DISPLAY AT(4,4):"INST
    ANCE-X CONVERTER FOR" :: DISPLAY AT
    (6,4):"TI-ARTIST INSTANCES TO"
240 DISPLAY AT(8,4):"PRINTER OR HEX FOR
    MAT" :: DISPLAY AT(1Ø,4):"by WESLEY
     R. RICHARDSON"
250 DISPLAY AT(14,6):"1=PRINT INSTANCE"
     :: DISPLAY AT(16,6):"2=CONVERT TO
    HEX CODE"
260 DISPLAY AT(18,6):"3=INSTRUCTIONS" :
    : DISPLAY AT(2Ø,6):"4=END"
270 CALL KEY(Ø,K,J):: IF J=Ø THEN 27Ø :
    : K=K-48 :: IF (K<1)+(K>4)THEN 27Ø
280 ON K GOTO 65Ø,42Ø,29Ø,84Ø
290 REM INSTRUCTIONS
300 CALL CLEAR :: DISPLAY AT(2,3):"INST
    ANCE PRINTING IS TO" :: DISPLAY AT(
    4,3):"EPSON TYPE PRINTERS."
310 DISPLAY AT(6,3):"HEX CODE OUTPUT IS
     16" :: DISPLAY AT(8,3):"CHARACTERS
     PER RECORD IN"
320 DISPLAY AT(1Ø,3):"THE FORM 91FDC32B
    786Ø5AE4" :: DISPLAY AT(12,3):"TYPE
     OF STRINGS."
330 DISPLAY AT(16,3):"FURTHER INSTRUCTI
    ONS ARE" :: DISPLAY AT(18,3):"IN FI
    LE INSTANCE-D."
340 DISPLAY AT(22,7):"PRESS ANY KEY"
350 CALL KEY(Ø,K,J):: IF J=Ø THEN 35Ø E
    LSE 21Ø
360 REM INPUT FILE
370 CALL CLEAR :: DISPLAY AT(14,6):"DSK
    Ø TO EXIT"
380 DISPLAY AT(6,4):"INSTANCE INPUT FIL
    E?" :: DISPLAY AT(8,6):D$ :: ACCEPT
     AT(8,6)SIZE(-15):W$
390 IF SEG$(W$,4,1)="Ø" THEN 21Ø ELSE D
    $=W$
400 E$=SEG$(D$,1,LEN(D$)-1)&"H"
410 RETURN
420 REM CONVERT TO HEX
430 GOSUB 36Ø
440 DISPLAY AT(1Ø,4):"OUTPUT FILE?" ::
    DISPLAY AT(12,6):E$ :: ACCEPT AT(12
    ,6)SIZE(-15):W$
```

...INSTANCE 3

```
450 IF SEG$(W$,4,1)="Ø" THEN 21Ø ELSE I
     F W$=D$ THEN 44Ø ELSE E$=W$
46Ø GOSUB 61Ø
47Ø IF X>14 THEN CLOSE #1 :: DISPLAY AT
    (16,4):"INSTANCE IS MORE THAN" :: D
    ISPLAY AT(18,4):"14 CHARACTERS WIDE
    " :: GOTO 34Ø
48Ø FOR J=1 TO 8 :: A(J)=Ø :: NEXT J
49Ø A(7)=X :: CALL LINK("SIZE",A(),B$):
    : C$(Ø,Ø)=B$
5ØØ FOR J=1 TO Y
51Ø DISPLAY AT(16,5):X;Y;J
52Ø FOR I=1 TO X :: GOSUB 63Ø
53Ø CALL LINK("CODE",A(),B$)! CONVERT 8
     NUMBERS IN A() TO HEX STRING IN B$
54Ø C$(I,J)=B$ :: NEXT I :: NEXT J :: C
    LOSE #1
55Ø DISPLAY AT(14,4):"WRITING TO DISK"
    :: OPEN #2:E$,DISPLAY ,FIXED 255
56Ø FOR J=1 TO Y :: DISPLAY AT(16,5):X;
    Y;J
57Ø B$="1B41Ø81B4B"&SEG$(C$(Ø,Ø),13,4):
    : FOR I=1 TO X :: B$=B$&C$(I,J):: N
    EXT I
58Ø W=LEN(B$):: B$=B$&"2Ø"&RPT$("Ø",252
    -W):: PRINT #2:B$
59Ø NEXT J :: CLOSE #2 :: GOTO 21Ø
6ØØ REM SUBROUTINES
61Ø DISPLAY AT(14,4):"READING DISK" ::
    OPEN #1:D$,INPUT
62Ø INPUT #1:X$,Y$ :: X=VAL(X$):: Y=VAL
    (Y$):: RETURN
63Ø INPUT #1:A$(1),A$(2),A$(3),A$(4),A$
    (5),A$(6),A$(7),A$(8)
64Ø FOR K=1 TO 8 :: A(K)=VAL(A$(K)):: N
    EXT K :: RETURN
65Ø REM INSTANCE PRINT
66Ø GOSUB 36Ø :: DISPLAY AT(14,6):"XXX
    TO EXIT"
67Ø DISPLAY AT(1Ø,4):"PRINTER NAME?" ::
     DISPLAY AT(12,6):"PIO.CR.LF"
68Ø ACCEPT AT(12,6)SIZE(-28):W$ :: IF S
    EG$(W$,1,3)="XXX" THEN 21Ø ELSE P$=
    W$
69Ø GOSUB 61Ø
7ØØ OPEN #2:P$ :: PRINT #2:D$;CHR$(1Ø);
    CHR$(1Ø);CHR$(27);CHR$(65);CHR$(8)
71Ø FOR J=1 TO Y
72Ø DISPLAY AT(16,5):X;Y;J
73Ø PRINT #2:CHR$(27);CHR$(75);CHR$(8*X
    );CHR$(Ø)
74Ø FOR I=1 TO X :: GOSUB 63Ø
75Ø CALL LINK("NUMB",A())
76Ø FOR K=1 TO 8 :: PRINT #2:CHR$(A(K))
    :: NEXT K
77Ø NEXT I :: PRINT #2:CHR$(13);CHR$(1Ø
    ):: NEXT J
78Ø PRINT #2:CHR$(27);CHR$(65);CHR$(12)
    ;CHR$(1Ø);CHR$(1Ø);CHR$(1Ø)
79Ø CLOSE #1 :: CLOSE #2 :: GOTO 21Ø
8ØØ REM ERROR
81Ø CALL SCREEN(9):: FOR I=1 TO 2ØØ ::
    NEXT I :: CALL SOUND(5ØØ,11Ø,Ø):: C
    ALL SCREEN(8)
82Ø RETURN 21Ø
83Ø !@P+
84Ø REM END
85Ø PRINT "STOP"
86Ø END
---------------------------------------
filename: INSTANCE/S

************************************
*                                  *
* INSTANCE-X EXTENDED BASIC  FILE  *
* INSTANCE/S ASSEMBLY SOURCE FILE  *
* INSTANCE/O ASSEMBLY OBJECT FILE  *
*                                  *
* WESLEY R. RICHARDSON             *
* MAY, 1989                        *
* NORTHCOAST 99ER'S - CLEVELAND, OH *
*                                  *
************************************
*
       DEF   CODE,NUMB,SIZE
*
NUMASG EQU   >2ØØ8       WRITE NUMBER
NUMREF EQU   >2ØØC       NUMBER GET
STRASG EQU   >2Ø1Ø       STRING ASSIGN
FAC    EQU   >834A       FAC ADDRESS
STATUS EQU   >837C       STATUS REGISTER
GPLWS  EQU   >83EØ       GPW WORKSPACE
*
MYREG  BSS   32          MY REGISTERS
BUF1   BSS   18          BUFFER 1
BUF2   BSS   18          BUFFER 2
SAVE   DATA >ØØØØ        RETURN ADDRESS
*
* CODE - A() CONVERTS TO B$
*        HEX CODE IN PRINTER FORMAT
*        CALL LINK("CODE",A(),B$)
*
CODE   MOV   R11,@SAVE RETURN ADDRESS
       LWPI  MYREG       SET UP REGISTERS
       BL    @GETA       A() -> BUF1
       BL    @SCRPRT     BUF1-> CVT-> BUF2
       BL    @HEXSTR     BUF2 -> B$
       B     @DONE       BACK TO BASIC
*
* SIZE - A() CONVERTS TO B$ HEX CODE
*        HEX CODE IN NUMBER FORMAT
*        CALL LINK("SIZE",A(),B$)
*
SIZE   MOV   R11,@SAVE RETURN ADDRESS
```

```
...INSTANCE 4

       LWPI MYREG     SET UP REGISTERS
       BL   @GETA     A() -> BUF1
       BL   @TOBUF2   BUF1 -> BUF2
       BL   @HEXSTR   BUF2 -> B$
       B    @DONE     BACK TO BASIC
*
* NUMB - A() CONVERTS TO A() NUMBER
*        NUMERIC IN PRINTER FORMAT
*        CALL LINK("NUMB",A())
*
NUMB   MOV  R11,@SAVE RETURN ADDRESS
       LWPI MYREG     SET UP REGISTERS
       BL   @GETA     A() -> BUF1
       BL   @SCRPRT   BUF1-> CVT-> BUF2
       BL   @HEXNUM   BUF2 -> A()
       B    @DONE     BACK TO BASIC
*
**************
*            *
* SUBROUTINES *
*            *
**************
*
GETA   CLR  RØ        LOOP COUNTER A()
       LI   R1,1      VARIABLE NUMB A()
       LI   R2,>1ØØØ  LENGTH OF B$ = 16
       LI   R5,BUF1   BUFFER FOR B$
       MOVB R2,*R5+   STORE LENGTH OF B$
LOOP1  INC  RØ        POINT TO ELEMENT
       BLWP @NUMREF   GET NUMBER
       LI   R2,>4Ø41  OFFSET FOR Ø, 1ØØ
       CB   @FAC,R2   IS IT < 1ØØ?
       JEQ  V99       YES JUMP TO 99
       LI   R2,1ØØ    VALUE 1ØØ TO 9999
       CLR  R3        SET UP CONVERSION
       CLR  R4        SET UP CONVERSION
       MOVB @FAC+1,R3 HIGH ORDER VALUE
       SWPB R3        RIGHT SIDE OF R3
       MPY  R2,R3     MULT BY 1ØØ, -> R4
       CLR  R3        SET UP LOW ORDER
       MOVB @FAC+2,R3 LOW ORDER VALUE
       SWPB R3        RIGHT SIDE OF R3
       A    R3,R4     R4 IS HEX ØØ - FF
       SWPB R4        PUT IN LEFT SIDE
       JMP  STO1      JUMP STORE IN BUF1
V99    CLR  R4        SET UP FOR Ø TO 99
       MOVB @FAC+1,R4 GET VALUE
STO1   MOVB R4,*R5+   PUT IN BUF1
       CI   RØ,8      FINISHED WITH 8?
       JNE  LOOP1     NO, DO AGAIN
       RT             YES, RETURN
*
*************************
*
SCRPRT LI   R1,>ØØ8Ø  POSITION VALUE
       LI   R2,BUF1+9 END POSITION
       LI   R6,BUF2+1 POINTER FOR BUF2
```

```
       LI   R9,>7FFF  BIT MASK
SLOOP2 LI   RØ,>ØØ8Ø  BIT VALUE
       LI   R5,BUF1+1 POINTER FOR BUF1
       CLR  R7        OUTPUT BYTE
SLOOP1 CLR  R3        INPUT BYTE
       CLR  R4        INPUT BYTE
       MOVB *R5+,R3   GET BYTE
       SZCB R9,R3     GET ON BIT
       SWPB R3        PUT IN LOW BYTE
       MPY  RØ,R3     ADJ FOR BIT
       DIV  R1,R3     ADJ FOR POSITION
       SWPB R3        MOVE TO HIGH BYTE
       AB   R3,R7     ADD TO OUTPUT
       SRC  RØ,1      POINTER TO RIGHT
       C    R5,R2     LOOP DONE?
       JNE  SLOOP1    NO, DO AGAIN
       MOVB R7,*R6+   SAVE BYTE
       SRC  R9,1      ADJUST MASK
       SRC  R1,1      ADJUST POINTER
       CI   R6,BUF2+9 LOOP DONE?
       JNE  SLOOP2    NO, DO AGAIN
       RT             RETURN
*
*************************
*
HEXSTR CLR  RØ        LOOP COUNTER
       LI   R5,BUF1+1 POINTER FOR BUF1
       LI   R6,BUF2+1 POINTER FOR BUF2
LOOP2  INC  RØ        INCREMENT COUNTER
       CLR  R4        SET UP REGISTER
       MOVB *R6+,R4   GET VALUE
       SWPB R4        TO LOW POSITION
       MOV  R4,R3     COPY VALUE
       ANDI R3,>ØØFØ  GET LEFT VALUE
       SLA  R3,4      IN HIGH ORDER BYTE
       CI   R3,>ØAØØ  VALUE < 1Ø?
       JL   CONLL     JUMP IF LOW
       AI   R3,>Ø7ØØ  ADJUST FOR ABCDEF
CONLL  AI   R3,>3ØØØ  ADJUST FOR STRING
       MOVB R3,*R5+   STORE LEFT VALUE
       ANDI R4,>ØØØF  RIGHT VALUE IN B$
       CI   R4,>ØØØA  VALUE < 1Ø?
       JL   CONRL     JUMP IF LOW
       AI   R4,>ØØØ7  ADJUST FOR ABCDEF
CONRL  AI   R4,>ØØ3Ø  ADJ. STRING VALUE
       SWPB R4        MOVE TO HIGH BYTE
       MOVB R4,*R5+   RIGHT VALUE IN B$
       CI   RØ,8      FINISHED A() ?
       JNE  LOOP2     NO, DO AGAIN
       CLR  RØ        B$ IS NOT AN ARRAY
       LI   R1,2      B$ IS VARIABLE 2
       LI   R2,BUF1   LOCATION OF B$
       BLWP @STRASG   WRITE B$ STRING
       RT             RETURN
*
*************************
*
```

...INSTANCE 5

```
TOBUF2 CLR  RØ          LOOP COUNTER
       LI   R5,BUF1+1   POINTER FOR BUF1
       LI   R6,BUF2+1   POINTER FOR BUF2
LOOP3  INC  RØ          INCREMENT COUNTER
       MOVB *R5+,*R6+   MOVE VALUE
       CI   RØ,8        FINISHED WITH A()?
       JNE  LOOP3       NO, DO AGAIN
       LI   R6,BUF2+7   SIZE VALUE
       CLR  R3          CLEAR REGISTER
       MOVB *R6,R3      GET SIZE
       SLA  R3,3        MULTIPLY BY 8
       MOVB R3,*R6      STORE SIZE
       RT
*
*************************
*
HEXNUM CLR  RØ          LOOP COUNTER A()
       LI   R1,1        VARIABLE NUMB A()
       LI   R5,BUF2+1   BUFFER FOR B$
       CLR  R6          ZERO
LOOP4  INC  RØ          POINT TO ELEMENT
       CLR  R3          SET UP WORK AREA
       CLR  R4          SET UP WORK AREA
       MOVB *R5+,R4     GET NUMBER
       CI   R4,>64ØØ    GREATER THAN 1ØØ?
       JL   N99         NO, LESS THAN 1ØØ
       LI   R2,>41ØØ    VALUE 1ØØ TO 9999
       MOVB R2,@FAC     STORE X1ØØ
       LI   R2,>ØØ64    DIVISOR = 1ØØ
       SWPB R4          IN LOW POSITION
       DIV  R2,R3       R3+R4/R2
       SWPB R3          PUT IN HIGH BYTE
       MOVB R3,@FAC+1   STORE HIGH BYTE
       SWPB R4          PUT IN HIGH BYTE
       MOVB R4,@FAC+2   STORE LOW BYTE
       JMP  ST02        GOTO WRITE
N99    LI   R2,>4ØØØ    SET FOR Ø TO 99
       MOVB R2,@FAC     STORE X1
       MOVB R4,@FAC+1   STORE NUMBER
       MOVB R6,@FAC+2   BALANCE IS ZERO
ST02   MOVB R6,@FAC+3   BALANCE IS ZERO
       MOV  R6,@FAC+4   BALANCE IS ZERO
       MOV  R6,@FAC+6   BALANCE IS ZERO
       BLWP @NUMASG     WRITE A() NUMBER
       CI   RØ,8        FINISHED WITH 8?
       JNE  LOOP4       NO, DO AGAIN
       RT               YES, RETURN
*
*************************
*
DONE   LWPI GPLWS       GPL WORKSPACE
       CLR  @STATUS     CLEAR STATUS REG.
       MOV  @SAVE,R11   RETURN POINT
       B    *R11        TO EXTENDED BASIC
       END
```

```
filename: TI-WORLD_I

5,1Ø
Ø,Ø,63,21,31,2Ø,28,24
Ø,Ø,199,255,1Ø9,56,4Ø,56
Ø,Ø,249,82,243,8Ø,112,48
Ø,Ø,227,1Ø7,54,28,2Ø,28
Ø,Ø,192,32,96,Ø,Ø,Ø
16,32,Ø,Ø,Ø,Ø,Ø,Ø
4Ø,56,4Ø,56,4Ø,56,4Ø,56
16,8,Ø,Ø,Ø,Ø,Ø,Ø
2Ø,28,2Ø,28,2Ø,28,2Ø,28
Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
12,16,8,7,Ø,Ø,Ø,Ø
4Ø,56,1Ø8,215,Ø,Ø,Ø,Ø
96,19,34,193,Ø,Ø,Ø,254
2Ø,28,119,235,Ø,Ø,Ø,Ø
Ø,96,32,192,Ø,Ø,Ø,Ø
Ø,Ø,Ø,Ø,Ø,1,2,2
7,24,33,66,132,8,16,17
125,186,85,84,146,146,146,17
192,48,8,132,66,33,16,16
Ø,Ø,Ø,Ø,Ø,Ø,128,128
4,4,4,8,8,8,15,8
33,33,34,66,66,66,255,66
17,17,16,16,16,16,255,16
8,8,136,132,132,132,255,132
64,64,64,32,32,32,224,32
8,8,4,4,4,2,2,1
66,66,34,33,33,17,16,8
16,16,16,17,17,17,146,146
132,132,136,8,8,16,16,33
32,32,64,64,64,128,128,Ø
Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
132,66,33,24,7,Ø,Ø,Ø
146,84,85,186,125,254,Ø,Ø
66,132,8,48,192,Ø,Ø,Ø
Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
7,24,48,112,8Ø,112,8Ø,112
252,15,6,3,1,1,3,7
1,6,14Ø,22Ø,84,22Ø,84,22Ø
255,3,1,Ø,Ø,Ø,Ø,1
Ø,192,16Ø,24Ø,8Ø,112,2Ø8,24Ø
56,15,Ø,Ø,96,48,56,4Ø
13,249,1,1,1,1,2,3
78,195,64,192,88,2Ø4,142,1Ø
3,254,Ø,Ø,Ø,Ø,Ø,Ø
8Ø,112,8Ø,112,8Ø,112,16Ø,192
6Ø,1Ø3,Ø,Ø,Ø,Ø,Ø,Ø
14,248,Ø,Ø,Ø,Ø,Ø,Ø
15,25,Ø,Ø,Ø,Ø,Ø,Ø
3,254,Ø,Ø,Ø,Ø,Ø,Ø
128,Ø,Ø,Ø,Ø,Ø,Ø,Ø

--------------------------------------
```

PUTTING GROM BASED CARTRIDGES IN YOUR CONSOLE (the easy way)
taken from HOCUS
By JOHN F. WILLFORTH

If you would like to install several(up to 6 GROM chips) inside your console without any circuit boards, and just a little wire, listen up!

Because the addressing is internal in a GROM, they can be stacked, and all of the GROM sockets in the TI-99/4A are placed for pin identical to each other. Take GROMS 0,1 and 2 from their sockets above the CPU chip, and stack them carefully and solder ALL 16 of their legs together. Now plug the whole 3-chip unit inthe GROM socket next to the Sound chip, observing direction(pin 1 location). Now you will note that you have two empty GROM sockets with the potential of six of these little beasties being stacked right on the CPU board. First though you better test the console to see that you have everything still operational.

Multi-Plan requires five GROMS, and EDITOR Assembler one, for a total of six and this will be one example of a boat for these sockets without cutting a hole in the RF shield to stack these chips to sky-scraper proportions. You may prefer TI-Writer, one GROM, Disk Manager II, two GROMS, Or any of the many GROM only cartridges that TI made, even games could be included in this list.

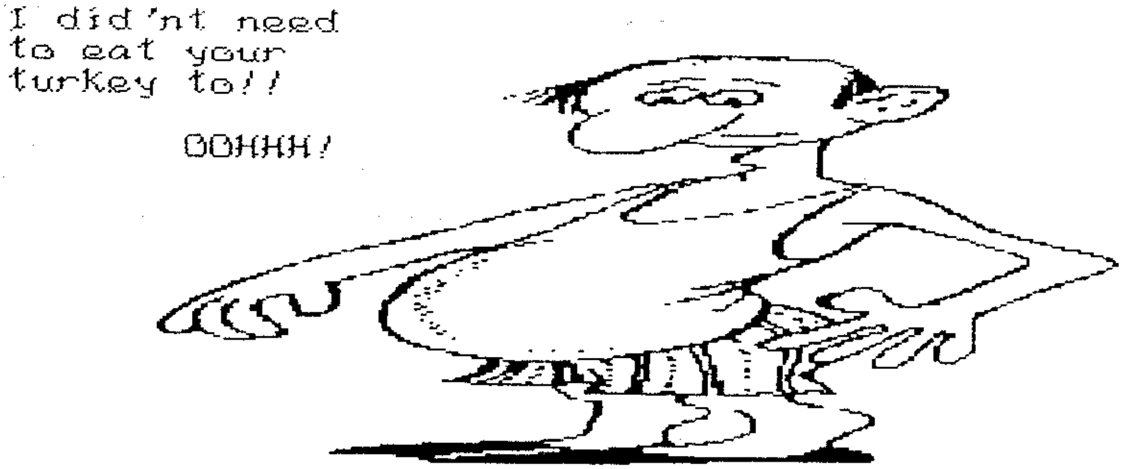
To Keep this simple, however, I reference Multi-Plan, and Editor Assembler, Remove the GROMS from their circuit boards carefully. Since you must Keep all five of the MP chips selected at the same time, I would recommend that you make note of the E/A chip so that it dosen't get mixed with the MP chips just yet. Stack any three of the MP chips and solder them together as you did to the console GROM chips earlier. Cut the part of pin 14 off the bottom chip of this 3-chip unit so that when this unit is inserted in the middl of the three sockets, there will be no connection to the corresponding pin in the GROM socket, but be sure that all three GROM pin 14s are soldered together. Now take the other two MP GROMS and piggyback them, and cut the bottom of pin 14 as before. Pick up the E/A GROM, and bend pin 14 straight out. You don't have to cut its pin. Slip the two MP GROMS on the TOP of the E/A GROM, and solder ALL but 14.

Using 3-12" lengths of multi-stranded wire (ribbon cable works well), attach the center wire to the column of three pin 14s on the console GROMS by using a low wattage soldering iron, and one of the other two wires to the single pin extended from the E/A GROM, and the remaning wire to the two pin 14s of the MP immediately above the E/A GROM. Connect the two MP GROM pin 14s to the three MP GROM pin 14s in the middle GROM socket using a short length of wire.

Using a SPDT switch (one that is OFF in the center, and will stay on when it is thrown to either side, solder the center wire (from pins 14) of the console GROMS to the center lug on the switch.

Before buttoning up the console, test the switch to see that if the switch is in the middle on lower lug, only Console BASIC is on the menu. When the switch is thrown in one of the two possible directions, on power-up, BASIC and EDITOR-ASSEMBLER will appear on the screen, and in the third position, on power-up, BASIC and MULTI-PLAN will appear on the screen. Then locate the switch inside the console in a convenient position and button it up.

If you desire more selections, follow this same scheme but use a rotary switch to allow more selections. This works and should take very little time. You may want to order your GROMS from TI, instead of using a cartridge



MACFLIX : A TECHIE TIP
by Robert j. Coffey (Western New York)

The key to getting a perfectly proportional picture is to have the density of pixels thesame both horizontally and vertically. A ratio of 1:1 to what you'd like, but not all printers offer it! In the vertical direction, printers(9-pin) have a density of 72 dots per inch. (So what you would like ideally is 72 per inch horizontally!)In Medium Resolution, MacFlix will select the graphic mode of #4, which is 80 dots per inch horizontally. That is pretty good, but we can do better than that! Graphics mode #5 offers 72 dots per inch horizontally(sxactly what we want!). In order to get MacFlix to choose #5 instead, you will need to go in to sector editor (Disk Utilities is one of the better ones) and search for the hex string "1B2A04", After you have found it, you'll want to change that "04" to "05". Save that secto back out and you're all set! Now when you select Medium Resolution the MacPaint pictures will go across the full 8 inches of the page, like they were suppose to! Why J.P.HODDIE did'nt do this in the first place is beyond me! (Ed. note J.P.Hoddie is forgiven!) Remember, only the newer printers have this capability!

Ogdens 99'er User Group
Mail Address: 1396 LINCOLN Ave. Apt.B Ogden Utah 84404
GROUP OFFICERS
President: JimBuck 773-2552
Vice President: Harold Bingham 394-6382
Secretary-Treasurer: Richard Scott 776-2551
Librarian: Harold Hilburn 773-0622
Asst. Librarian: Mel Bragg 393-9605
Newsletter Editor: Mel Bragg 393-9605
Associate Editor: Harold Hilburn 773-0622

NOVEMBER 1989 NEWSLETTER

OUR NEXT MEETINGS ARE:

SATURDAY: NOVEMBER 04 TIME: 0900 hrs.
TUESDAY: NOVEMBER 21 TIME: 1900 hrs.

We will be meeting in the CIVIL AIR PATROL building at the OGDEN MUNICIPAL AIRPORT AIRPORT ROAD.

OGDEN TI USERS GROUP
1396 LINCOLN APT #B
OGDEN, UTAH 84404