The Boston Computer Society TI-99/ 4A User Group Meeting Newsletter May 1989

Edited by J. Peter Hoddie

The June Meeting

The June meeting of the Boston Computer Society's TI User Group will be held on Wednesday June 21 at 7:30 PM. The location is Room 540 of the Massachusetts College of Art on Huntington Avenue in Boston.

There will be new disks in the software library. With a little bit of luck the new software library catalogs should also be complete.

The meeting topic is up in the air. Read the column to the right and you'll understand why, at least a bit. I would suppose that if by some chance Press were to be released that it would be shown but I really can't speculate on that. Depending on the progress I make on my primary project (tentative title: Sign Shop) a version of that program may be demoed.

Beyond that I have no clue. As always, the meeting should be interesting, informative, and entertaining. The June meeting would probably also be a good time for the group to discuss future meeting topics as the well of ideas seems to running a bit dry at present. Think of a meeting topic and suggest it in June. Perhaps even you could do a presentation...

Random Ramblings By J. Peter Hoddie

I have been thinking about how to write this column for several weeks now. Literally dozens of approaches have passed through my mind, but I haven't been able to bring myself to sit down and actually write it. So, as usual, time has run out. It 12:33 in the afternoon on the day of the meeting. Thanks to the miracle of desk top publishing and a new copier at the BCS office this newsletter should still reach your hands at the meeting tonight.

This month, rather than my usual round up of BCS news, TI world news, and the odd program or whatever, I get to make the announcement I had promised last month. In short, I'm leaving the Boston area at the beginning of June. As many of you know, I've been a student since I began volunteering for the BCS. I've spent the last four years at Boston University's College of Engineering. Graduation is this Sunday. That means the next step for me is to seek employment. My current plans are to move out to the Silicon Valley area to search for a job.

Being about 3000 miles away obviously means that I'm not going to be able to continue my current level of involvement with the BCS. I've been codirector of the TI group along with Justin Dowling for about five years now. Since taking over from Bob Schledwitz, Justin and I have helped this user group become one of the most recognized and respected in the country. In that time, the software

The Boston Computer Society
TI-99/4A User Group
One Center Plaza
Boston, MA 02108

library was started, the newsletter has gone through multiple incarnations in reaching its present semi-regular format, we've held four very successful annual TI shows, and put up two bulletin board systems. Most importantly however, we have held a meeting every month.

I do not intend to terminate my involvement with the BCS entirely either. The BCS TI User Group has been a great influence and education for me over the past five years. I couldn't just walk away from it if I wanted to. I hope to continue my monthly contribution to the newsletter. Over the past few months, Justin has been patiently learning how to turn out a newsletter that look like this using the Macintosh, so the format shouldn't be adversely effected. One thing I have always wanted to include in the newsletter is a Ouestion and Answer column. The only problem is that no one ever gives me any written questions to use. Since I won't be at meetings, if you have a question that can't be handled by someone within the group (and with the talent we have that is pretty unlikely) write it down and send it to my through the BCS TI User Group (this applies to members of other user groups who get this newsletter as well) and I'll try to answer it in a future newsletter. I also hope to be able to work some more on the software library. Because of school and such, I have been unable to really put the time into the software library lately that it deserves. I should have a bit more time in the next few months, so I should be able to update some disks and add a bunch more.

Beyond the newsletter and software library I also hope to check into the bulletin boards a couple times a week. This should provide a reasonable means of communication for many. Actually, just last week I picked up a brand new US Robotics Courier 2400e Modem from the BCS office that will be going on BBS #1 pretty soon. From my limited testing over the past few days, the modem works very well and because of some of its features may actually improve BBS service for all callers, not just those at 2400. I also have some half finished enhancements to the BBS that may go into place before I leave.

Just so that everyone really knows what I'm planning in my life over the next few months (i.e. let's squash the rumors before they get silly...) I will be staying temporarily with some relatives until I can find my own place to live, generate some income, and purchase a vehicle (minor details, right?). To keep my relatives from going insane, I will not make their telephone number available to the TI community at large. When I get my own phone number I'll publish it here so y'all can call be as usual. Clearly one cannot hope to make a respectable living in the TI world any longer (although as a college student, I managed quite well from TI world revenues). Thus, I am hoping to work for a company doing work for the Macintosh computer. My primary reason for focusing on the Mac, is that it is a home computer more so than most of what is out there. PC's are big, ugly, scary machines. The Mac in my mind is much closer to what a home computer should be. And because of my long and positive involvement with the TI which was certainly a home computer, I want to continue to pursue that aspect of computers. Business applications tend to be much more boring. Besides, I would much rather go to a trade show and discuss program ideas with a bunch of normal individuals than a bunch of businessmen in suits.

There have been many individuals in the BCS who have helped me out at many levels since I first became involved. There is no way to describe the help provided to me by Tom Ward and Wendell Davis way back in the early days. They loaned me a second disk drive, 1200 baud modem, and countless hours of time, patience, and support in getting the bulletin boards, and software library together. Without their initial support, the I have no idea where me or the user group would be. Also may thanks to the only two people (apart from Justin...) who were at my first BCS TI Meeting, Dennis Kelly and Bill Tormey who have helped set up and take down nearly every meeting for as long as I can remember. There were also the early "techie" crowd who really helped to expand my programming horizons, in particular John Bonavia and David Taub. There's Walt Howe who provided one of my first real links to the rest of the world through his CompuServe activities.

An odd assortment of thanks to one Corson Wyman who put me together with MYARC for what turned out to be a stormy relationship. Arnold Carlson and Stan Jacobsen get credit for actually suggesting that we do a show in Boston, and for working at the BCS shows and also countless Ken Gorden extranvaganzas. There are the now gone people like Jay Giurleo, Greg Knightes (though Greg still frequents Delphi), and (the famous in song and legend) Brenda Lyons. Cynthia Becker has since moved out to Seattle after providing many newsletter articles, disks for the software library, and incredible enthusiasm especially on the BBS. Donald Mahler has provided countless c99 articles, and much BBS support in addition to keeping my honest on several counts and providing many new disks for the library. Mike Wright provided us with some of the most controversial and fascinating perspectives on the TI world for many months, and continues to handle the arduous task of managing the newsletter exchange. Mike has also worked many a TI show for the BCS. Joyce Corker's TI-Writer Tips and Tricks provided a great publicity boon for the BCS and the importance of her work on the early TI Faires can never be overstated. Joe Rawlins has been a long time volunteer, providing articles, lugging equipment to meetings, and generally helping out. Also many thanks to Bill Wallbank.

There are probably a dozen or three people I've left out of the list above and I apologize. Part of the problem is I'm running out of time to type this. The other part of the problem is a failing memory - I hear old age does that to you.

I do want to put in a little bit about the future of the group. In my mind there is some phenomenal talent brewing in our user group. Mark Van Copennole (creator of the Grammulator) is not only talented with hardware, but he has become a formidable assembly programmer and a good speaker as a result of the Grammulator project. Walt Howe was born to lead the TI User group meetings. With his booming voice, incredible breadth of TI knowledge, and ability to give clear explanations Walt should be an important factor in the future. Donald Mahler is not only a c99 guru,

but quite familiar with Multiplan and I suspect many other things. Aaron West is probably the best assembly coder in the TI world today. I'm not kidding either. Some of the code I've seen from him makes Paul Charlton look bad. An example of Aaron's work is elsewhere in this newsletter. Aaron's father Don West has an extraordinary sense of what programs should do, and I expect he could provide some excellent review style presentations given a little encouragement. Mike Wright has one of the most extensive collections of TI memorabilia and such that exists outside of the Traver home. If he could be convinced to just demo one TI rarity every other month or so it would be amazing. Mike has managed to uncover many things TI worked hard to bury. Justin Dowling has been doing some pretty high powered work with data base programs and RAM disks and such and should be able to provide some "power user" presentations. Joe Rawlins has a way of coming up with an interesting little program or modification that's worth a look. I think the answer here, is that with me gone there is a great opportunity for some of our more talented members to step forward. I've been running meetings for like five years and you people must be getting bored of the same old jokes and typos. Support these people, and consider stepping forward yourself once in a while to help out.

And just to make sure no one misunderstands: no I am not leaving the TI community. I am leaving Boston. With some luck I'll make it back for a meeting or two in the future. I have TI projects under way that I intend to finish. I like the TI computer more than any other machine at there because it truly is a home computer. I will continue with Genial Computerware as well. I also have no intention of becoming actively involved with any West Coast TI user groups. I hope to continue my association with the BCS TI Group for as long as possible.

c Column

By Donald L.Mahler

Clint Pulley has released his C99MDOS. Unlike his earlier bits-and-pieces updates for the 9640, which required using

older files, this is complete in itself, so 9640 users can discard all older files and depend completely on this file. Besides a 70 page manual, which includes references to suggested texts, it includes the newest versions of the compiler, QDE, assembler, LDR, headers, a graphics library, and a floating point library. (while we still do not have a floating point data type, it is much simpler to use that the older floating point routines used with the 99/4A). In order to fit everything on a DSSD or two SSSD dsks, all files are archived and a copy of Barry Boone's ARCHIVER is included. The AUTOEXEC file copies the compiler, assembler, loader, and other frequently used files onto the Geneve ramdsk, so editing, compiling, etc. is extremely fast.

The process of writing a c program now runs like this:
1)Call up QDE, using a name something like B:file_c
2)Write your c.file. Instead of using individual extern()
statements, use #include "a:topic_h". For example, stdio_h
will load fopen, fclose, fread, fwrite, getc, putc, printf,
sprintf, scanf, sscanf, etc. Save to dsk by pressing ESC twice,
and answering "Y"
3)Load C99, the automatic compiler assembler. Format is

3)Load C99, the automatic compiler assembler. Format is "C99 Bifile_c Bifile_o" Program will be compiled and assembled, showing any errors.

4)Load LDR. Format is "LDR B:file_o, library, library", as, for example, "LDR B:file_o, A:FLOATLIB,D:C99LIB" (Remember we are in MDOS so we can call drives a,b,c,etc). c program will load and run. If you want to save as a program file which will run from MDOS, use "LDR B:file_o,a:FLOATLIB,d:C99LIB b:programname".

As a trial, I converted some of my old simple programs from earlier c.columns:

```
/* Volume of a box */
#include a: "stdio h"
#include a: "float_h"
#define x 8
char *op;
                      /* for calculations */
main()
  \{float a[x],b[x],c[x];
                             /*declare
floating pt numbers */
   float j[x], k[x]; char *s;
   printf("VOLUME OF A BOX\n\n");
   puts("Enter length:\n");
   fpget(s,a);
                            /*picks up value
and puts in fp a*/
   puts("Enter width:\n");
   fpget(s,b);
   puts ("Enter height:\n");
   fpget(s,c);
   flop(a,"*",b,j);
                      flop(j,"*",c,k);
multiplication */
   locate(30,3);
   printf("Volume = ");
   fpput(k,s);
                  } /*outputs floating
point value */
Or another way of doing this:
/*Volume of a box #2 */
#include a: "stdio h"
```

```
#include a: "float h"
#define x 8
char op[2];
char s[x],a[x],b[x],c[x];
   {float j(x),k(x),l(x),m(x),p(x);
   printf("VOLUME OF A BOX\n\n");
   puts ("Enter
length<sp>height<sp>width:\n");
   scanf("%s %s %s",a,b,c);
   stof(a,j); stof(b,k); stof(c,m);
*string to fp */
   flop(j, "*", k, p); flop(p, "*", m, 1);
   locate(30,3);
   printf("Volume = ");
   fpput(l,s);
Another example:
/* USING SSCANF AND PRINTF */
#include "a:stdio h"
char str[] = "ATDP6177393421";
main()
  { char d[10]; int a,e,n;
    printf("Original string is %s",str);
    sscanf(str,"%45%3d%3d%4d",d,&a,&e,&n); /
*picks bits out of string */
    putchar('\n');
    printf("Dial code is %4s \n",d);
    printf("Area code is %3d \n",a);
    printf("Exchange is %3d ".e);
    printf("and number is %4d".n); }
/* DEMO OF SPRINTF */
#include a: "conio h"
char *fn, *ln;
main()
   (char buf[72],a; int i;
    a = 'L'; fn = "Donald"; ln = "Mahler";
    i = 24:
   printf("First name is %s ,last name is
%s, and inital %c",fn,ln,a);
   printf("\n User number is %d \n",i);
   sprintf(buf, "%-7s%2c%8s%3d\n", fn, a, ln, i);
   printf("Here is full name\n\n');
   puts(buf);
```

For the first two examples, remember to use FLOATLIB and C99LIB with LDR; for the third and fourth examples, you need only the C99LIB. C99MDOS will be added to our library as two SS disks very shortly if it is not already.

Introduction to the UCSD P-System By Ron Williams

This month I will cover the writeln procedure in more detail. This procedure is one of the most used procedures in pascal as most of the writing to the screen and other devices will use this procedure. The writeln procedure writes text files and can not write records like the procedure PUT can. One of

the great things about writeln is that it can also format output to the screen, printer or other device when writing out the data.

The following program is a demo of formatting with the writeln statement as you can see I have gave you the option of directing the output to the screen or to the printer. If you output to the printer you can study the output in much greater detail.

```
program testwrite(input,output);
const
  pi = 3.14159
var
  pfile : text;
  choice : char:
  count, count2 : integer;
(* This program will demo the *)
(* use of writeln in pascal
(* The program uses loops to
(* show output formatting
                               *)
begin
  page (output);
  gotoxy(1,1);
  write('(1)screen (2)printer=>');
  read(choice);
  case choice of
    'l' : rewrite(pfile,'console:');
    '2' : rewrite(pfile,'printer:');
  end:
  gotoxy (1, 4);
  for count:=1 to 6 do
    writeln(pfile,pi:8:count);
  writeln(pfile);
  for count:=1 to 8 do
  writeln(pfile,pi:count:1);
  writeln(pfile);
  for count:=1 to 8 do
    for count2:=1 to 6 do
        writeln(pfile,pi:count:count2);
  close (pfile, lock);
  page (output);
end.
```

This program will first show in the first loop how to print a real number with a different number of spaces after the decimal point. The first value after the colon: is the number of spaces to print over from the left edge of the paper or screen and the second number is the number of decimals to print after the decimal point notice as the values in the loops change the output also changes. If the valve given is not really possible like printing one space over after wanting two spaces after the decimal point the program will print the number in scientific notation so make sure the values work.

A sample of the output from the first loop is shown below:

```
3.14
3.142
3.1416
3.14159
+3.1415900000000e+000
```

A sample of the output from the second loop is shown below:

```
+3.1415900000000e+000
+3.1415900000000e+000
+3.1415900000000e+000
3.1
3.1
3.1
3.1
```

So far I have shown output with real numbers well writeln can also output strings, integers, and packed array of characters. The only real difference is that you would not use two colons like with real numbers. The number after the colon would only be for the number of spaces to print over from the left edge of the paper or from the last write on the same line. The value of the number after the colon if not possible will try to print the number or string as well as possible and if to small will chop off any characters on the end. One more point I would like to make is that write and writeln is right justifed unlike print in basic I know this will sometimes be a problem for beginning pascal programmers but soon you will grow to get used to it and may even find it much easier to format the printout of numbers I know I did. Just keep in mind that the number will be printed from the left over X amount of characters. Well thats it for this month, Thanks.

Music Maker Cracked! By Aaron West

The article "Music Maker on Disk" (December 1987,P16-20) made me a little upset at myself, for not having released this information sooner. I modified the TI-Debugger to debounce our load interrupt switch so that it would return to the calling program, and about a year ago I used it to figure Music Maker's data structure out and transfer a file from cassette to disk. Then I wrote a program to slowly play Music Maker songs in Extended BASIC. I was thinking of writing a program to convert Music Maker songs to format that would play quickly in XBASIC, but I left that project, thinking I'd write a better (although probably simpler) Music Maker later. After seeing the article, I wrote a non-working assembly cassette-to-disk program, and then this simpler, working, load-interrupt switch version:

```
* MUSIC MAKER CASSETTE-TO-DISK#2 (MMCD2)
```

^{*} TO USE:

^{* 1} LOAD IN E/A#3 LOAD&RUN BUT DON'T RUN

^{* 2} INSERT MUSIC MAKER CARTRIDGE

^{* 2} GO TO MUSIC MAKER

```
* 3 PRESS LOAD INTERRUPT SWITCH
* 4 LOAD FROM CASSETTE
* 5 PRESS LOAD INTERRUPT SWITCH
* 6 SAVE TO DISK
* DONE! GO TO STEP 3 FOR ANOTHER
      REF VMBR, VMBW
      BSS 32
WPPCST DATA WS-32+6
      AORG -4
                     LD.INT WP.PC
       DATA WS, MMCD
      RORG
BUFLEN EQU >3FFF->3BE3
RIF
       BSS BUFLEN
                     SAVE DISK BUFFERS HERE
MMCD
      MOV @WPPCST, @-4 LD.INT.TRASH IN RO-R2
       LWPI WS
                     KEEP RTWP VALUES IN R13-R15
       CLR RO
DB
       INCT RO
                     DEBOUNCE LD.INT.SWITCH
       JNO DB
                     16384x; APPROX 1/8 SECOND
       MOV @>8370,R0 ENDVDP->3FFF?
       CI RO,>3FFF
       JEQ MVPTRS
                     YES; MOVE POINTERS
       LI
           RO.>3FFF
       MOV RO, 6>8370 MAKE END VDP=>3FFF
       LI
           RO,>3BE4 DISK BUFFER START
       LI
            R1, BUF
       T.T
            R2, BUFLEN
      BLWP @VMBR
       JMP RTN
* MOVE POINTERS FROM >3FFD TO >3BE1
MVPTRS
           RO, > 3BE3 ENDVDP FOR CALL FILES (1)
      LI
      MOV RO, @>8370
           RO,>3FFD FIRST PTR IS AT END
      LI
           R2,2
                 2 BYTES PER PTR
      LI
           R1, R3*2+WS
MOVPTR
      BLWP @VMBR
                     R3=PTR
      AΙ
          RO,-BUFLEN MOVE DOWN
      BLWP @VMBW
      AI RO, BUFLEN-2 PREV. ADR-2
      MOV R3, R3 = 0?
       JNE MOVPTR MOVE UNTIL PTR=0
           RO,>3BE4 RESTORE VDP
      LI
      LI
           R1,BUF
                     CPU ADR OF DISK BUFFER
           R2, BUFLEN
      LI
      BLWP @VMBW
                     WRITE DATA BACK TO VDP
RTN
      STWP RO
      MOV RO, 0-4 PUT WP BACK IN LD.INT WP
      RTWP
      END
```

Now for the interesting part; Music Maker's data format: Cassette save: VDP >0300 to ENDVDP. ENDVDP=>3FFF without disk, >3BE3 with.

Disk: 128 byte records. REC 0 is at VDP >05C6 to >0645. RECs 1 to 114 are at VDP >0300 to >3BE3. Last byte is REC 114, byte >63.

VDP >059E=Timing: >0404 for 4/4, >0208 for 2/8, etc.

Page 6 VDP > 05A5=#Sharps; VDP > 05A6=#Flats VDP >05AE=LOW (>006E=110 Hz) & VDP >05B0=HIGH (>0370=880) for discrete S.G. VDP >05C5=Speed 1-30; REC 6, byte 69. VDP >0606-0641=2 byte frequencies for discrete sound graphs; starts at 110. 7 note values per octave. Note numbers 0,2,3,5,7,8,10 in each octave. (GROM>6160-619B=Default frequencies for discrete sound VDP >0784-07B9=>8314-8349; scratch pad memory is used by Music Maker. VDP >0784=Mode:>0000=traditional, >0001=sound graph. (>8314)VDP >0787=Sound graph mode:>00=discrete, >01=Continous. (>8317) VDP >07B2=No. of measures (2 bytes); REC 10, byte 50-51. VDP >1000=First measure; REC 27, byte 0. ENDVDP-2=Pointer to first measure (>1000). ENDVDP-2*(measure#-1)=Pointer to measure. VDP >0680-073F=temp. measure buffer for edit. VDP >0520-052F=Sound list buffer: GROM>A040-A04F=Default sound list:(>0B 8C1A AC1A CC1A E0 9FBFDFFF 02\{00 0520\} GROM>A050-A057=Noise values >E0E1E2E3E8E9EAEB. Last 4 should be >E4E5E6E7. Because of this bug noises 5-8 are the same as 1-4 (white GROM>A058-A0ED=Sound chip note value table. Each 2-

GROM>A058-A0ED=Sound chip note value table. Each 2-byte (word) value has offset >8000 (refer to E/A ref.manual.) GROM>A0EE-A12D=byte note nos. First is >24, last is >00. >00 is 110 Hz, >01 is

110*2^(1/12) Hz, >02 is 110*2^(2/12) Hz, n is 110*(2^(1/12))^n Hz.

GROM>A12E-A152=Note table offsets for 37 notes on a staff

1. Traditional mode; four bytes per note:

Byte 1:Volce >01 to >04 or >FF (-1) to >FC (-4); upside down note is negative.

Byte 2:Note table pointer; even no. >7E to >00. >7E is lowest. -6—next higher

note, subtact 2 from >7E-(>06*note) for sharp, add 2 for flat. >FF=blank.

This byte is the offset to the table at GROM>A058.

Byte 3:Volume >90-9F,>B0-BF,>D0-DF,>F0-FF for voice 1,2,3,4.

Byte 4:Length >01-10 (note lengths 1-16), >21-30 (rest lengths 1-16). The

length is the number of spaces used by the note, rest, or blank (invisible)

rests. 1=16th note, 2=8th, 3=dotted 8th, 4=4th, 6=dotted 4th, etc.

2. Sound graphs; 3 bytes per note. Length is always 1: Byte 1:>00-03 voice (corresponds to voice 1-4)
Byte 2:note >00-FF. In discrete mode, >00 (lowest) to >1D are offsets to the table at VDP >0606. >FF is a blank space. Continuous mode allows >00 to >77.

Byte 3:Volume >90-9F,>B0-BF,>D0-DF,>F0-FF as above.

All 4 notes are always specified, in order, e.g. >000590 01FFBF 02FFDF 03FFFF.

Each measure is 20 notes * 4 voices * 3 bytes = 240 bytes! That allows only 46 measures with disk, 51 without. There are no measure pointers. Continuous mode uses a simple multiplier to convert note numbers into frequencies. This multiplier seems to be INT((880-110)/120) default or INT((HIGH-LOW)/120) if the range values are changed, and is placed at >8346 (2 bytes.)

VDP >05AE=LOW (2 bytes) & VDP >05B0=HIGH.

The following program converts a traditional mode Music Maker file to one of three data formats, and the next three programs play the song in the corresponding formats.

```
90 REM MMCONV: Music Maker Convert, version
1.4 by Aaron West.
100 DIM NF(64):: FOR I=0 TO 63 ::
NF(I)=ASC(SEG$("@AABCBCDDEFFGHGHIIJKKLMMNONOPP
QRRSTSTUUVWWXYYZ[Z[\\]^^_'_aabccd",64-I,1))-
64 :: NEXT I
110 DISPLAY AT (2,0) ERASE ALL: "MUSIC MAKER
FILE CONVERTER": : "ENTER DSKn. FOR A W
HOLE DISK": :"INPUT FILE NAME?": :"DSK1."
120 DISPLAY AT(10,1):"OUTPUT FILE NAME?":
:"DSK1 ": : : "MODE: 1=COMPRESSED DF128"
: :" 1 2=NUMERIC
                       IF252": :"
                                       3=DATA
MERGE DV163"
130 ON WARNING NEXT :: ACCEPT AT(8,0)SIZE(-
28) BEEP: IF$ :: ACCEPT AT(12,0) SIZE(-2
8) BEEP:OF$ :: ACCEPT AT(17,3) SIZE(-1) BEEP:SM
140 DIM F$(127):: ID$=IF$ :: OD$=OF$ ::
AF=LEN(IF$)=5 :: IF AF THEN OPEN #1:IF$,
INPUT , INTERNAL, FIXED ELSE 170
150 INPUT #1:F$(FC),A,B,C :: IF A=1 AND B=59
AND C-128 THEN PRINT IF$6F$(FC):: F
160 IF B THEN 150 ELSE CLOSE #1 :: FC=FC-1 ::
PRINT : "WRITING:"
170 IF AF THEN IF NOT FC THEN
IF$=ID$&F$(FC):: OF$=OD$&SEG$(F$(FC),1,8)&"/
"&STR$
(SM):: PRINT OF$ :: FC*FC-1 ELSE STOP
180 OPEN #1:IF$, RELATIVE, INPUT
190 SY$="" :: LN=1000 :: LINPUT #1,REC 6:L$
:: LINPUT #1,REC 10:M$ :: NM=ASC(SEG
$(M$,51,1))*256+ASC(SEG$(M$,52,1))
200 IF ASC(SEG$(M$,6,1)) THEN PRINT
"INVALID:SOUND GRAPHS MODE" :: GOTO 310 ELSE
T(1), T(2), T(3)=1 :: ND, P, L=0 :: LINPUT #1, REC
27:P$
210 IF SM=1 THEN OPEN #2:OF$, FIXED 128, OUTPUT
ELSE IF SM=2 THEN OPEN #2:OF$, INTE
RNAL, FIXED 252, OUTPUT ELSE OPEN
#2:OF$, VARIABLE 163, OUTPUT
220 MT=(ASC(SEG$(L$,31,1))*32/
ASC(SEG$(L$,32,1)))^3 :: FOR M=1 TO NM :: FOR
X=1
TO 3 :: V$(X) = "" :: NEXT X
230 IF LEN(V$(1)) *LEN(V$(2)) *LEN(V$(3))=MT
```

```
THEN 260 ELSE IF P>127 THEN P=0 :: LI
NPUT #1:P$
240 FOR X=1 TO 4 :: D$(X) -SEG$(P$,P+X,1)::
NEXT X :: P=P+4 :: V=ASC(D$(1)):: IF
V>4 THEN V=256-V
V$(V) = V$(V) & RPT$(D$(2) & D$(3), ASC(D$(4)) AND
31):: GOTO 230
260 FOR N=1 TO LEN(V$(1))STEP 2 ::
Y$=SEG$ (V$ (1), N, 2) & SEG$ (V$ (2), N, 2) & SEG$ (V$ (3)
,N,2):: IF Y$=$Y$ AND L<15 THEN L=L+1 ELSE
GOSUB 320 :: L=0 :: SY$=Y$
270 NEXT N :: $Y$=Y$ :: NEXT M
280 GOSUB 320 :: X=-1 :: IF SM-2 THEN ND=99
:: GOSUB 350 :: GOTO 300
290 GOSUB 350 :: IF ND THEN 290
300 CLOSE #2
310 CLOSE #1 :: IF AF THEN 170 ELSE STOP
320 IF SY$="" THEN RETURN
330 FOR V=1 TO 3 ::
NT(V) = NF(ASC(SEG\$(SY\$, V+V-1, 1))/2 \text{ AND } 63)::
VL(V)=ASC(SEG$(S
Y$, V+V, 1) ) AND 15 :: NEXT V
X,W=(NT(1)+NT(2) *40+NT(3) *1600) *65536+VL(1) *256+VL(2) *16+VL(3) +124096
350 IF SM>1 THEN ND-ND+1 :: IF X+1 THEN X=W/
104857600 :: GOTO 380 ELSE 380 ELSE
360 IF X<0 THEN W=2^32-1
370 FOR X=1 TO 4 :: W=INT(W)/256 ::
O$=CHR$((W-INT(W))*256)&O$ :: NEXT X :: PRIN
T #2:0$;:: RETURN
380 IF SM=2 THEN IF ND<28 THEN PRINT #2:X,::
RETURN ELSE PRINT #2:X :: ND=0 :: R
ETURN
390 X$=STR$(X)::
$3$=$3$&CHR$(200)&CHR$(LEN(X$))&X$ :: IF
ND<12 THEN S3$=S3$&CHR
$(179):: RETURN ELSE ND-0
400 PRINT #2:CHR$(INT(LN/
256)) &CHR$ (LN) &CHR$ (147) &S3$&CHR$ (0)::
LN=LN+10 :: S3$=
"" :: RETURN
90 REM MMPLAY1: Enter the name of your type 1
compressed file to play.
100 NL=150 :: DIM F(40), PL$(99):: H=2^(1/
12):: F(0)=110 :: FOR I=1 TO 40 :: F(I)
=F(I-1)*H::NEXTI
110 DISPLAY AT (3,0) ERASE ALL: "MUSIC MAKER
PLAY1": :: "INPUT FILE NAME?": :"DSK
120 ACCEPT AT (9,0) SIZE (-28) BEEP: IF$ :: OPEN
#1:IF$, RELATIVE, INPUT
130 FOR I=0 TO 99 :: TF EOF (1) THEN 150 ELSE
LINPUT #1:PL$(I)
140 NEXT I
150 P$=PL$(PL):: PL=PL+1 :: FOR I=1 TO 128
STEP 4 :: Y = (ASC(SEG\$(P\$, I, 1)) *256 + AS
C(SEG\$(P\$,I+1,1)))/1600 :: N3=INT(Y) :: IF
```

```
N3>39 THEN STOP
160 Y=(Y-N3)*40 :: N2=INT(Y) :: Y=(Y-N2)*40 ::
N1=INT(Y)
170 L=ASC(SEG$(P$, I+2,1))::
 Y=ASC(SEG$(P$,I+3,1)):: V1=L+L AND 30 ::
L=INT(L/16)
+1 :: V2=(Y AND 240)/8 :: V3=Y+Y AND 30
180 CALL
SOUND (NL*L, F (N1), V1, F (N2), V2, F (N3), V3):: NEXT
I :: GOTO 150
MMPLAY2: Enter the name of the type 2 file to
100 NL=150 :: DIM F(40), N(1500):: H=2^(1/
12):: F(0)=110 :: FOR I=1 TO 40 :: F(I)
=F(I-1)*H::NEXTI
110 DISPLAY AT (3,0) ERASE ALL: "MUSIC MAKER
PLAY2": :: "INPUT FILE NAME?": :"DSK
120 ACCEPT AT (9,0) SIZE (-28) BEEP: IFS :: OPEN
#1:IF$, INTERNAL, FIXED, INPUT :: FOR I
=0 TO 1500 STEP 28 :: FOR J=I TO I+26 STEP 9
130 INPUT
1:, (8+L), (7+L), (6+L), (6+L), (CH2), (CH2)
NEX
T J :: INPUT #1:N(J):: IF N(J)<0 THEN 150
140 NEXT I
150 FOR I=0 TO 1500 :: Y=N(I):: N3=INT(Y)::
IF Y<0 THEN STOP
160 Y=(Y-N3)*40 :: N2=INT(Y) :: Y=(Y-N2)*40 ::
N1=INT(Y):: Y=(Y-N1)*16 :: L=INT(Y)
):: Y=(Y-L)*4096
170 V1-(Y AND 3840)/128 :: V2=(Y AND 240)/8
:: V3=Y+Y AND 30
180 CALL
SOUND (NL*L+NL, F(N1), V1, F(N2), V2, F(N3), V3)::
NEXT I
MMPLAY3: To create a song program: 1. Load this program
2. MERGE your type 3 file in. 3. Save the finished version
under a new name.
100 NL=150 :: DIM F(40), NF(64):: H=2^(1/12)::
F(0)=110 :: FOR I=1 TO 40 :: F(I)=
F(I-1)*H :: NEXT I :: Z=1/104857600
110 READ Y :: IF Y<0 THEN STOP ELSE Y=Y*Z ::
N3-INT(Y)
120 Y=(Y-N3)*40 :: N2=INT(Y) :: Y=(Y-N2)*40 ::
N1=INT(Y):: Y=(Y-N1)*16 :: L=INT(Y
):: Y = (Y - L) * 4096
130 V1=(Y AND 3840)/128 :: V2=(Y AND 240)/8
:: V3=Y+Y AND 30
140 CALL
SOUND (NL*L+NL,F(N1),V1,F(N2),V2,F(N3),V3)::
GOTO 110
MMPLAY/AS: Requires MMPLAY object code from
MMPLAY/S in DSK1.
100 DIM PL$(99):: DISPLAY AT(3,0) ERASE
ALL: "MUSIC MAKER PLAY1, ASSEMBLY": : : :"I
NPUT DRIVE?": :"DSK1.": :"FILE?": :"NOTE
```

```
LENGTH? 4"
110 DISPLAY AT(15,1):"TRANSPOSE? (0-40) 0" ::
ON WARNING NEXT
120 ACCEPT AT (9,0) SIZE (-28):D$ :: ACCEPT
AT(11,7)SIZE(-10):F$ :: OPEN #1:D$&F$,R
ELATIVE, INPUT
130 FOR I=0 TO 99 :: IF EOF(1) THEN CLOSE #1
:: GOTO 150 ELSE LINPUT #1:PL$(I)
140 NEXT I
150 ACCEPT AT (13, 14) SIZE (-2):NL :: IF NL THEN
ACCEPT AT (15,19) SIZE (-3): T :: ON E
RROR 170 :: CALL
LINK ("PLAY", I, PL$(), NL, T, T, T) ELSE 120
160 ON ERROR STOP :: GOTO 150
170 CALL INIT :: CALL LOAD ("DSK1.MMPLAY")::
RETURN
Here is the assembly play routine!
MMPLAY/S: assemble before using PLAY/AS.
* PLAY ROUTINE FOR 3 VOICES: CALL LINK
* ("PLAY", P$, NL: SECONDS * 60, t1, t2, t3)
* ("PLAY", #elements, P$(), NL, t1, t2, t3)
* P$ includes 32 note groups of 4
* bytes each. The 4 bytes (2 words) are
* 3 note#s (0-39) in the first word,
* and 3 volumes (0-15) and a time (0-15)
* in the 2nd. The time is actually 1-16.
* 1st word=n1*40^2+n2*40+n3 (0-63999)
* 65535 (>FFFF) flags end of list.
* n1-3=half step numbers: 0=110 Hz.
* 12=220 Hz, 24=440 Hz, etc.
* 2nd word=time*16^3+v1*16^2+v2*16+v3
* NL=no. of 60th sec's per 16th note
* P$(0 | #elements 1) are moved to himem
* so your program must not be too large.
* t1-3 are transpose values for voice1-3
* CALL LINK("PST",P) returns P=no. of
\star notes to play, 0 if done.
***
       COPY "DSK6.XBEQU"
 DORG >2000 XB EQUATES
NAMLNK BSS 2
FFALM BSS 2
LFALM BSS 2
INIFLG DATA >AA55
NUMASG BSS 4
NUMBEF BSS 4
STRASG BSS 4
STRREF BSS 4
XMLLNK BSS 4
KSCAN BSS 4
VSBW BSS 4
VMBW BSS 4
VSBR BSS 4
VMBR BSS 4
VWTR BSS 4
ERR
ENDHM EQU >8384
RORG
*** End of XBEQUates.
       DEF PLAY, PST (PST=play status)
       EQU >A101 HiMem buffer for array
BUF
       DORG >A040 put values here
ADRE
       BSS 2 adr.of end of P$() buffer
```

```
2 seconds*60 per 16th note
NTLEN BSS
           2 duration of note group
NTDUR BSS
       BSS
           2 ptr.to current note group
4 transposition #s for v1-3
TRANS BSS
PINTWS BSS 32 my workspace
       RORG $ restore relocation counter
* Lowest 12 half steps: counter values
* First octave is these values shifted
* right 1, 2nd is shifted 2, etc.
SNDTBL DATA 2034,1920,1812,1710,1614
       DATA 1524,1438,1357,1281,1209
       DATA 1141,1077
       DATA 12 half steps per octave
       DATA 40 possible half steps
QUIET DATA >9FBF, >DF0F sound values
* Play STatus: returns 0 if not playing
       MOV @>83C4,@>834A ISR playing?
       BLWP @XMLLNK return ISR address
       DATA >20
                 Convert Int.to Float
       CLR RO
                    not an array
       LI
           R1,1
                    parameter#1
       BLWP @NUMASG return the value
       JMP GRT
                    back to XB
GETNUM MOV R11,R12 save return adr
       MOV R10,R1 into R1
       CLR RO
                   not array
       BLWP @NUMREF get float.value
           @>12B8 CFI:float.to int.
* CFI uses R0 to R6 & R11
       INC R10
                   next parameter
            *R12
                   return RO-int.
* Load P$() into high memory;>A140 on
           R10,1
                   parameter counter
       MOV R10,1
                     default 1 element
       MOVB @>8300,R1 1st param. ID.
       SRL R1.1
                     string parameter?
       JNC PLAY2
                     no, skip
       BL
            @GETNUM get no. of elements
PLAY2 MOV RO.R5
                     save it
       CLR RO
               element 0 or not array
       MOV R10,R1 param# to R1
           R2, BUF
                    buffer for array
       MOV R4, @BPTR point to 1st group
       SETO R6
                     max len.255
GETSTR MOVB *R2,R4
                     save old byte
      MOVB R6.*R2
                     set len.limit
      BLWP @STRREF get P$(R1) or P$
      MOVB *R2,R3
                    get len in R3
      MOVB R4,*R2
                     restore old byte
       SRL R3,8
                    R3=string length
      Α
           R3,R2
                     go to end of string
       INC RO
                     next element
           R0,R5
                     done enough?
       JL
           GETSTR
                     no, go on
       INC R10
                     next param.
      MOV R4, @ADRE store adr of end
      BL
           @GETNUM
                    get note length
      MOV RO, ENTLEN save it
      LI
           R9, TRANS point to buffer
           @GETNUM get a transpose#
TRANS2 BL
      MOVB @>83E1, *R9+ save low byte
      CI R9,TRANS+3 comp.with end
       JNE TRANS2
                     loop 3 times
PLAYS I.T
           R1,PINTR play interrupt adr
      MOV R1,@>83C4 install ISR
```

```
init.pointer-
       LI
            R1,BUF+1
            R1,@BPTR to first group
       MOV
       CLR
           entdur
                      init for first
            0>6A
                      return to XB
* Play INTeRrupt does the work
PINTR DEC
           entdur
                      count down
       JCT RT
                      if >0 then leave
       LWPI PINTWS
                      use my workspace
            R3,>8400 sound chip write
       LI
       MOV @BPTR, R2 start now
      MOV *R2+,R6
                      get note#s
            R6,64000 in range?
       CI
       JHE DONE
                      no, end
           *R2+,R7
                      get volumes & time
      MOV
       MOV R2, @BPTR save pointer
      MOV R7,R1
                      >Tabc a=v1,b=v2,...
       SRL R1,12
                      >000T in R1
                      make it 1 to 16
       INC RI
       MPY
           0NTLEN,R1 Time*NL
       MOV R2, @NTDUR store duration
       SLA R7.4
                     >abc0 for loop
            R4,>8000 note/volume setter
            R9, TRANS transpose #s
PINT2 CLR R5
                      init for div
       DIV @D40,R5
                      div note#s by 40
* R5=what's left; R6=modulo 40 remainder
       CLR RO
                     init for div
       MOVB *R9+,R1
                      get transpose#
       SRA R1.8
                     make it a word val
           R6,R1
                     Rl=note# for div
       Α
       DIV @D12,R0
                     div by 12
* R0=octave; R1=half step 0-11
       SLA R1,1
                     *2=table offset
       MOV @SNDTBL(R1),R1 =note
       INC RO
                   make it nonzero
       SRL R1, R0
                     shift by octave+1
      MOV R1, R0
                     =0abc order
       SRC RO.4
                    =c0ab
       SRL RO, 4
                     =0c0a
           R4,R0
                     =Vc0a Voice=E,C,8
      MOVB RO, *R3
                     write >Vc (4 LSbs)
       SLA R1,4
                    =abc0
      MOVB R1,*R3
                     write >ab (6 MSbs)
      MOV R5, R6
                     for next div.
      ΑI
           R4,>1000 vol.set >90,B0,D0
       SRT. R7,4
                      >0abc, >0bc0, >0c00
      AB
           R4, R7
                      set voice command
      MOVB R7, *R3
                     write volume
       SLA R7,8
                     >bc00,>c000,>0000
      A1
           R4,>1000 next voice
           R4,>E000 noise set command
      CI
       JLT PINT2
                     don't set noise
RT
      LWPI >83E0
                     restore GPLWS
      RT
            return to interrupt handler
DONE
      CLR @>83C4
                     wipe out ISR link
           R1,QUIET stop sound
      LI
      MOVB ^R1+, ^R3 >9F, BF, DF, OF
       JLT Q2
                     for v1, v2, v3, stop
      JMP
           RT
                     return
      END
```