BADA — MANUAL VAX/VMS

Second updated edition, May 1986
by Tamas Ungvary

1 BADA

BADA , I.E. DACTEST is a package of programs for

1. sound synthesis
2. signal processing
3. testing synthesis and processing
4. miscellaneous

FILES to be used :

.DAC	sound files
.WAV	periodic function files
.REV	parameterfile for REVERB
.SPL	users ASCII file for multiple splice
.DAT	combined command and data file for BADA and
.WSP	command files used within WSP

OVERVIEW.

CALLABLE ROUTINES IN DACTEST.

Synthesis	Wave	Processing	Miscellaneous
TAPE			
SRATE		CONVOL*	
CHANS	DRAWAV¨	COPYDA	
CSTEP	DEWAVE	DELETE	
PTRIG	ASSWAV	DISTRI	
	LISTWV	DIVIDE	CALL
FMCON	MIXWAV	EDITDA¨	COUNT
MULCON	IFFT*	GAIN	EXIT
FGT	WAVWSP	MAXIMI	INTRPT
CLEMS		MIX	LENGTH
		MIX2	LEVEL
		MMIX	PLAY
CDA		MSPLIC	TWAVE
FGF		REVERB*	WSP
CDA		RING	WRITDK
FGA	EXTWAV	SPECSH*	
FGW	EXTENV	SPEED	
FGP		SPLICE	
FM			
TIME			
ENDPLY			

Others: AMPLQ, CDQ, CONNEC, DISCON, FGQ, RECORD

NOTE ! The programs marked * use the array processor
¨ use the Tektronix terminal

```
---------------------------------------------------------------------
!                                                                   !
! FILENAMES must  N O T  contain more than  6  characters           !
!                                                                   !
! for BADA . If a file will not be accepted remember this warning.!
!                                                                   !
! ALL BADA commands must be typed as UPPERCASE characters .         !
!                                                                   !
---------------------------------------------------------------------
```

DACFILES.

According to the convention at EMS, the soundfiles are situated at the disk DRA1: and have the extension .DAC, thats why they are also called DAC files. Every DAC file contains an indefinite number of data items, organised in RECORDs of 4096 data items.
A mono DAC file contains the sampling values following each other.
The stereo DAC file contains the data items as follows :

```
|c1|c2|c1|c2|c1|c2|  |
|s1|s1|s2|s2|s3|s3|  | etc.
----------------------------------------------------
```

```
        where   c1  and  c2         are the channel numbers
                s1, s2, s3, sn      are the sample numbers
```

The four channel DAC file has the same kind of format but for 4 channels.

```
|c1|c2|c3|c4|c1|c2|c3|c4|
|s1|s1|s1|s1|s2|s2|s2|s2| etc.
----------------------------------------------------
```

The range of the sampling values is between + and - 32767. The value -32768 is used to detect overflow i.e. distortion.

At present, all DAC files use one physical DA converter, which limits the sampling rate one can use. The max. sampling rates for

```
mono file          65000
stereo file          40000
4 channel file          20000
```

By default, BADA is computing with 20000 samplingrate.

WAVFILES.

These files are used as

1. waveforms for synthesis
2. control functions

A .WAV file can contain up to 2048 data items, where the f i r s t data item is an I N T E G E R telling the programs how many values will follow. The sampling values of the waveforms are of type real and are between -1. and 1. (The file can contain any no. of elements between 1 and 2047. The reading programs adjust (by interpolation) the no. of elements to their requirements.

The .WAV files shall and will be situated on DRA1: too.

ROUTINES FOR WAVEFILES.

DRAWAV

Allows the user to define wave-forms by drawing on the T4112 terminal.
THE FOLLOWING KEYS ARE THEN INTERPRETED AS INSTRUCTIONS:

-` move cursor RIGHT distance specified by previous number
´- move cursor LEFT - " -
? move cursor UP - " -
V move cursor DOWN - " -
= move cursor to middle y value, and set breakpoint
0 - 9 set EITHER distance for moves made by arrows OR curve SHAPE, if preceded by S
S set curve SHAPE to the number immediately following (there may be minus or plus sign between S and number)
- put shape sign to MINUS (valid one shape number only)
+ put shape sign to PLUS
/ DEFINE BREAKPOINT at the position of the cursor
X DELETE BREAKPOINT, if there is one within 5/1024 points on the screen
I INTERPOLATE segment, from the breakpoint nearest left to the breakpoint nearest right, using the curve form defined by S
N fill segment from the breakpoint nearest left up to, but not including, the breakpoint nearest right, with the same value as at the left-hand breakpoint
D DELETE segment between nearest left breakpoint and nearest right breakpoint, but not the breakpoints themselves
P PLOT whole wave, as defined so far, screen cleared first
W WRITE on screen current values of wave
L LOGARITHMIC/LINEAR switch
M MAXIMIZE amplitude of current internal wave
O OUTPUT wave file; program asks for name
R READ in wave file; program asks for name
E display EXTERNAL .WAV file, but without altering internal buffers
C CLEAR; i.e. reset program to its start condition

(continued on next page)

H HELP: print menu of commands on terminal
CTRLZ return to calling program
CTRLG clear screen

ASSWAV

assigns a waveform to the internal or external wave table.

The program contains 7 wave tables to be used , numbered from 1 to 7. Each of them contains 2048 data items representing one preiod of an oscillation. These tables contain one period of the following waveforms , by default :

1	SINE	
2	PARABOLA	
3	TRIANGLE	
4	SAWTOOTH	
5	SQUARE	
6	PULS1	one-sided pulse
7	PULS2	double-sided pulse

These waveforms are the same ones as the waveforms of the analog generators of the old computer studio.

The user can however reassign the functions to one of the waveform numbers by

EITHER

choosing the same or three more available functions from the INTERNAL waveform buffers,

as

RANDOMT	random table
RANDOMR	rectangular random function - frequency has no effect
RANDOML	logarithmic random function - frequency has no effect

OR

EXTERNALLY, by assigning a specified wave file (.WAV) to one of the waveform numbers 1 - 7. Any wave file previously assigned to this number is automatically deassigned; the wave form can then be accessed by using its number in calls to FGQ and FGW.

MODE (I=INTERNAL E=EXTERNAL):	as described above
NAME :	.WAV file name (without extension, but with device and directory specifier if necessary)
NUMBER:	waveform number (1-7) to which this file is to be assigned

on command/data file :

```
ASSWAV
E          means EXTERNAL
P3         name of the .WAV file
1          wavetable 1
```

DEWAVE

de-assigns a waveform from the external wave table

EXTWAV

Extracts specified samples (i.e. one period of an oscillation from a .DAC file and converts them to the format required for .WAV files, and outputs the result to a named file.

```
NAME OF INPUT .DAC FILE (NO EXTENSION)
  You cannot give a directory here !
NUMBER OF CHANNELS IN .DAC FILE
CHANNEL TO BE EXTRACTED TO MAKE .WAV FILE
FIRST SAMPLE TO BE EXTRACTED
LAST SAMPLE TO BE EXTRACTED
DESIRED SIZE OF OUTPUTFILE
  Maximum is 2047 . (Last sample - first sample ?)
NAME OF OUTPUT .WAV FILE (NO EXTENSION)
```

EXTENV

Extracts an envelope from a specified part of a .DAC sound file and writes the result to a .WAV file. Used e.g. for amplitude modulation.

```
DACFIL name (no extension) of .DAC file to be read from
NUMBER OF CHANNELS ON DACFIL
CHANNEL WHOSE ENVELOPE IS TO BE EXTRACTED
NUMBER OF THE FIRST SAMPLE TO BE EXTRACTED
NUMBER OF THE LAST SAMPLE TO BE EXTRACTED
SAMPLE PERIOD, i.e. number of .DAC file samples to be examined
   for each ENVfil sample (max 2047)
NAME (NO EXTENSION) OF OUTPUT .WAV FILE
```

The values in the envelope are adjusted to the range -1 to 0.

FORTRANWAVE

Instead of using DRAW, you can produce your own .WAV file by writing a FORTRAN routine.

```
c       the formalities :
        REAL BUF(2047)
        INTEGER ICOUNT
        CHARACTER NAME          ! the file name
        NAME='MYWAV1'
        OPEN(
                UNIT=91',
                FILE='DRA1:'//NAME//'.WAV',
                RECL=2048
                FORM='UNFORMATTED'
                STATUS='NEW')
```

fills the buffert BUF(2047) with own values. ICOUNT tells how many values you do have (not necesserily 2048) . The values must be between -1. and 1. (X). If your algorithm produces values between 0. and 1. (Y) , transform them like X=2*Y-1

```
        WRITE(91)ICOUNT,BUF
        CLOSE(91)
        STOP
        END
```

LISTWV

Lists the assigned waveforms within BADA .

MIXWAV

Mixes two .WAV files, writing the result to a third .WAV file. The length of the output file is that of the longer input file; new values are interpolated if necessary in the shorter input file.
The .WAV files must be situated on DRA1:

WAV1	name (no extension) of first input wave
WAV2	name (no extension) of second input wave
OUTWAV	name (no extension) of output wave

IFFT

Does an inverse FFT on a .WAV file, and writes the result to another .WAV file.
The .WAV file must be situated on DRA1:

SIZE OF OUTPUT WAVE: MUST BE 128, 256, 512 OR 1024
NAME OF INPUT .WAV FILE (NO EXTENSION)
NAME OF OUTPUT .WAV FILE (NO EXTENSION)

WAVWSP

Converts the function of a .WAV file into a function of either a WSP ´function box´ or of a WSP ´sequencer box. The output is a WSP file (.WSP) containing a simple FUNCTI or SEQUENCE definition. This definition must be MODIFied (or edited on the file), i.e. must be suplied with SWITCHes na d TRIGgers, etc.

TOTAL DURATION (SECS), of the output function

Note ! the total duration of the music may be different depending on the value of the ´studio sample´ (default=10 ms). If you havwe a segment duration of e.g. 0.051 sec, the program rounds it to 0.06

MIN VALUE, MAX VALUE of the output function
"FUNCTI" or "SEQUEN"
NAME OF BOX

ROUTINES FOR PROCESSING

CONVOL

Performs convolution on two sound files

```
TYPE  0: convolute .DAC file with a .WAV file that
         represents a frequency spectrum for filtering
      1: convolute two .DAC files with one another
DIRECTION  +1: forward (i.e. cross-correlation)
           -1: backwards (convolution)
INPUT1              name of first input file
FIRST SAMPLE        start position in file 'INPUT1'
LAST SAMPLE         last position in 'INPUT1'
INPUT2              name of second input file
FIRST SAMPLE        start position in file 'INPUT2'
GAIN FACTOR         real number . factor on 'INPUT2'
                    (amplitude = gain/sqrt(windowsize))

OUTPUT              name of output file
RECORD SIZE         number of samples to be dealt with at
                     a time (1-2048).
MIN.WINDOW          smallest window size (3-maxwin)
MAX.WINDOW          largest window size (minwin-2047)
INCREMENT           window size increment: added to or
                    subtracted from window size every
                    'RECSIZ' samples
START WINDOW        window size at beginning
```

COPYDA

Copies a specified part of a .DAC file to a specified part of another .DAC file, overwriting any information that may have existed previously in the output file at the position(s) in question.

```
INPUT :                name of input file (no extension)
FIRST SAMPLE :         start position in input file
LAST SAMPLE :          last position in input file
OUTPUT :        name of output file (no extension)
FIRST SAMPLE :  start position in output file
```

DELETE

Copies one .DAC file to another, but omitting a specified portion of the input file.

```
INPUT           name of input file (no extension)
FIRST SAMPLE    start position in input file
LAST SAMPLE     last position in input file
OUTPUT          name of output file (no extension)
```

DISTRI

Reads a mono .DAC file and creates a stereo or quadrophonic file with distribution controlled either by a periodic function or by a fixed factor.
As periodic function you use one of the waveforms stored within BADA.

```
                INPUT FILE:          name of input file (no extension)
                FIRST SAMPLE:        in input file
                LAST SAMPLE:         in input file (-131070=end of file)
                OUTPUT FILE:         name of outputfile
                NUMBER OF OUTPUT CHANNELS (2 OR 4), SAMPLING RATE :
                                sampling rate in Hz of INPUT file

======= IF       STEREO OUTPUT :
                 distribution mode
                        0    = fixed control for distribution between
                               left and right (on X axis, if 4 channels)
                        1-7  = periodic function control ( from internal
                                                            wavetable )
   --------IF       DISTRIBUTION MODE HAS BEEN DEFINED    0
                give position        ( i.e. distribution factor)
                                   -1=LEFT, 0=CENTRE, +1=RIGHT)

   --------IF       DISTRIBUTION MODE HAS BEEN DEFINED    1-7
                delay mode
                give frequency in HZ of periodic function
                        0 = fixed control for distribution between
                            left and right (on Y axis, if 4 channels)
                        1-7=periodic function control

   --------IF        DELAY MODE HAS BEEN DEFINED    0
                give delay factor in samples ( -4096 to +4096 )

   --------IF        DELAY MODE HAS BEEN DEFINED    1-7
                give frequency in HZ of periodic function for delay

======== IF         QUADROPHONIC OUTPUT :

                X axis mode
                        0 = fixed control for distribution between
                            left and right (on X axis, if 4 channels)
                        1-7=periodic function control


  --------IF        DISTRIBUTION MODE HAS BEEN DEFINED    1-7        :

                give FREQUENCY in HZ of periodic function


                LEFT LIMIT, RIGHT LIMIT (X AXIS)
                LEFT LIMIT, RIGHT LIMIT (Y AXIS)
```

DIVIDE

Subroutine that divides a stereo .DAC file into two mono .DAC files, copying each channel to specified parts of other .DAC files, and ERASING any information (after FIRST2 and FIRST3)that may have existed previously there.

INPUT - character - name of input file (no extension)
FIRST1 - integer - start sample-number in input file
LAST1 - integer - last sample-number in input file
OUTPUT1 - character - name of output file 1 (no extension)
FIRST2 - integer - start sample-number in output file 1
OUTPUT2 - character - name of output file 2 (no extension)
FIRST3 - integer - start sample-number in output file 2

NOTE ! FIRST2 and FIRST3 must be in the range 1 to current length of the file + 1 (If new outputfile(s), they must be 1)

GAIN

Amplifies a .DAC file
NOTE ! As periodic function you use one of the waveforms stored within BADA.

INPUTFILE	name of input file (without extension)
FIRST SAMPLE	first sample in input file
LAST SAMPLE	last sample in input file
OUTPUTFILE	name of output file
FIRST SAMPLE	usually 1, but if the OUTPUTFILE.DAC already exists, you can choose to re-write it from the FIRST SAMPLE.
MODE	0: single gain value throughout 1-7: control gain with periodic function -1: follow the amplitude of a .DAC file
GAIN	if MODE has been defined 0 gain factor (`0) 1-7 frequency of periodic function

SAMPLING RATE in HZ of INPUT file

======= IF MODE has been -1 then

NAME OF CONTROL FILE :
CONTROL SAMPLE SIZE (128,256,512,1024):
START AT SAMPLE NO. : ;

MAXIMIZE

maximizes the amplitude of a .DAC file

INPUT name of file

MIX

Mixes two .DAC files, writing the result to a third file. The two inputfiles may be the same ones b u t MIX takes always the l a t e s t version of the named inputfiles.

INPUT 1	name of the input file to be mixed
FIRST SAMPLE	start sample of sound to be mixed
NUMBER OF SAMPLES	to be mixed
INPUT 2	name of the second inputfile
FIRST SAMPLE	start sample of the soundfile to be mixed
OUTPUT	name of the output file
FIRST SAMPLE	if there is an existing .DAC file and you

named it as outputfile (the latest version of it will be taken), you can keep a portion of it by answering this question. Otherwise give the value 1.

MIX TYPE 0 : straightforward addition of the two files
1 : maximize in proportion (output level=1.0)
2 : maximize 50/50 (- " -)

MIX2

Mono/stereo mixer . It writes a parameter file that can be called by "MMIX" to perform mixing of up to 100 mono or stereo .DAC file segments to one mono or stereo file, with parameters controlled by ONE OF THE INTERNAL WAVETABLE (numbered from 1 to 7 .) The resulting parameterfile must be executed by "MMIX".

The program asks for :

MIN GAIN, MAX GAIN, GAIN .WAV, .WAV FREQ:

Every filesection which will be choosen to be mixed, gets a gain value. This gain value is defined as a sample value of the GAIN .WAV (one of the internal tables) at the starttime of the particular mix,between MIN and MAX values. The given value of .WAV FREQ defines which section of the wavetable should be read or how many times should be read respectively.This frequency value must be scaled to the total duration , i.e. to the last start time of a mix. (see example below.)

MIN DEN, MAX DEN, DEN .WAV, .WAV FREQ:

The density defines time between start-time of mixings, i.e. how many mixes should occure in comparison to the total duration (: i.e. to the last start time of a mix. (see example below.)

MIN POS, MAX POS, POS .WAV, .WAV FREQ:

Position in the room (-1 to + 1)

MIN DELAY, MAX DELAY, DELAY .WAV, .WAV FREQ:

Delay between channels (if stereo) in samples (max 4096)

MIN FADE, MAX FADE,LAST TIME(secs.):

Fade in/out in samples and The last time a mixing shall start.

DEFINE YOUR INPUTFILES ; CLOSE WITH EMPTY LINE.

You can give as many file definitions as you wish. The program will choose randomly a file at each start time.

FILE NAME :

CHANNNELS, FIRST SAMP, LAST SAMP, MIN SIZE, MAX SIZE :

Define the section of the file where all parts should be taken from (FIRST SAMP and LAST SAMP).Define the MIN/MAX size of the part within the section to be taken and mixed. The actual length of the part is then dependent on a random choice.

FILE NAME :

CHANNNELS, FIRST SAMP, LAST SAMP, MIN SIZE, MAX SIZE :

etc. (end with RETURN only)

Example :

Suppose you wish to mix 5000 sample long sections of a file, choosen between samplings 100000 and 200000 . The last mix on the resulting file should occur at time 20. sec .This 20 seconds is our reference duration .
At the start you wish to have 1 mix per second , on the middle of this file (totally 20. sec long!) 10 mixes per second and at the end 1 mix per second again. These values define a half period of a triangle waveform (stored at the 3-rd internal wavetable) .

```
max        *
          * *
         *   *
        *     *     .
                  .   .
                   . .
min                 .
```

The minimum value of the waveform itself is NOT identical with our density minimum because of that the period of the waveform starts in in the middle of between min/max . Our density minimum should be therefore be 2.sec instead. Observe, we want to use only the half of the period (i.e. freq. .5) during a time of 20. seconds . Therefore the whole periodtime is 40. sec. To get the appropriate density maximum (which should be 10 mixing / second = 0.1) over time ,we have to devide 1 second by 40. which gives us 0.025. for the value of frequency.
Our values then :

MIN DEN, MAX DEN, DEN .WAV, .WAV FREQ:2,.1,3,0.025

To get the opposite function APPLIED FOR GAIN, i.e. the more mixings the less gain, (wanted min= .5 , wanted max = 1.)

MIN GAIN, MAX GAIN, GAIN .WAV, .WAV FREQ:1.5,0.5,3,0.025

As a result for these parameters above you get the following outputfile (to be processed by MMIX) . Examin the time and the gain.

```
Start                          gain
time
0. TDAC 156845 161845 1849 1935 1. 0. 0 1
1.050 TDAC 190015 195015 1294 1918 0.9475 0. 0 1
2. TDAC 145648 150648 1638 1921 0.899988 0. 0 1
2.860 TDAC 144624 149624 1858 1423 0.856989 0. 0 1
3.639 TDAC 170820 175820 1937 1154 0.818075 0. 0 1
4.343 TDAC 144521 149521 1093 1957 0.782858 0. 0 1
4.980 TDAC 181846 186846 1937 1629 0.750986 0. 0 1
5.557 TDAC 192786 197786 1616 1139 0.722142 0. 0 1
6.079 TDAC 170287 175287 1979 1776 0.696039 0. 0 1
6.552 TDAC 167927 172927 1040 1004 0.672415 0. 0 1
6.979 TDAC 141384 146384 1781 1208 0.651036 0. 0 1
7.366 TDAC 101500 106500 1199 1232 0.631687 0. 0 1
7.716 TDAC 136900 141900 1102 1553 0.614177 0. 0 1
8.033 TDAC 101383 106383 1544 1192 0.598330 0. 0 1
8.320 TDAC 169800 174800 1211 1433 0.583989 0. 0 1
8.580 TDAC 179729 184729 1673 1171 0.571010 0. 0 1
8.815 TDAC 186948 191948 1291 1396 0.559264 0. 0 1
9.027 TDAC 175485 180485 1291 1853 0.548634 0. 0 1
9.220 TDAC 190396 195396 1226 1547 0.539014 0. 0 1
9.394 TDAC 117146 122146 1376 1482 0.530307 0. 0 1
9.551 TDAC 153816 158816 1660 1911 0.522428 0. 0 1
9.694 TDAC 178616 183616 1920 1701 0.515297 0. 0 1
9.823 TDAC 143497 148497 1152 1502 0.508844 0. 0 1
9.940 TDAC 158762 163762 1815 1145 0.503004 0. 0 1
10.046 TDAC 166012 171012 1223 1580 0.502281 0. 0 1
10.150 TDAC 160058 165058 1074 1043 0.507498 0. 0 1
10.264 TDAC 107236 112236 1328 1202 0.513210 0. 0 1
10.389 TDAC 102615 107615 1176 1673 0.519466 0. 0 1
10.526 TDAC 152999 157999 1008 1777 0.526315 0. 0 1
10.676 TDAC 178536 183536 1775 1999 0.533815 0. 0 1
10.841 TDAC 120255 125255 1862 1315 0.542027 0. 0 1
11.020 TDAC 126867 131867 1771 1716 0.551020 0. 0 1
11.217 TDAC 177365 182365 1123 1511 0.560866 0. 0 1
11.433 TDAC 193188 198188 1748 1458 0.571649 0. 0 1
11.669 TDAC 167975 172975 1918 1937 0.583455 0. 0 1
11.928 TDAC 113868 118868 1431 1821 0.596384 0. 0 1
12.211 TDAC 143089 148089 1143 1590 0.610540 0. 0 1
12.521 TDAC 125134 130134 1161 1283 0.626041 0. 0 1
12.860 TDAC 105301 110301 1790 1166 0.643015 0. 0 1
13.232 TDAC 100039 105039 1683 1682 0.661602 0. 0 1
13.639 TDAC 160929 165929 1920 1616 0.681954 0. 0 1
14.085 TDAC 189114 194114 1931 1964 0.704240 0. 0 1
14.573 TDAC 119937 124937 1721 1384 0.728642 0. 0 1
15.107 TDAC 181627 186627 1231 1008 0.755363 0. 0 1
15.692 TDAC 168180 173180 1674 1986 0.784623 0. 0 1
16.333 TDAC 145888 150888 1515 1717 0.816662 0. 0 1
17.035 TDAC 131665 136665 1234 1769 0.851745 0. 0 1
17.803 TDAC 177673 182673 1043 1870 0.890161 0. 0 1
18.645 TDAC 131330 136330 1984 1827 0.932226 0. 0 1
19.566 TDAC 187360 192360 1870 1787 0.978288 0. 0 1
```

MMIX

Multi-mixer . It reads from a parameter file containing any number of records. The parameterfile can be created with MIX2 or in any other ways (record format see below).

It asks for :

```
OUTPUTFILE :         name of outputfile withouth extension (.DAC)
                     ( It could be an already existing one)
SAMPLINGRATE, NUMBER OF OUTPUTCHANNELS :
PARAMETERFILE (incl extension)
 The previously created parameterfile (e.g. with MIX2)
INCLUDE CURRENT OUTPUT (I) OR OVERWRITE (O) :
```

One record of the parameterfile must contain the following values:

```
TIME (SECS) : start time of one mix (GLOBAL, in seconds)
FILE        : name of .DAC file withouth extension
FIRSTSAMPLE : start sample in file
LASTSAMPLE  : last sample in file,( negative number = END-OF-FILE)
FADEIN      : number of samples for FADE-IN
FADEOUT     : number of samples for FADE-OUT
GAIN        : gain factor for this file (1.0 = 0 DB, 0.0 = -100 DB)
DISTRIBUTION: position between two speakers (-1 = LEFT, +1 = RIGHT)
DELAY       : delay between channels in samples (0 TO 4096)
CHANNELS    : number of channels (1 or 2) on this file
```

All parameters, except "file", may be written as integers or real numbers. Numbers may be omitted from the end of the line, in which case the last defined value for this parameter remains valid.

MSPLIC

Multiple splice. This program produces an .DAC outputfile by splicing unlimited (SIC !) amount of .DAC-file sections together, according to the specification of the user, stored on an ASCII file, with default extension .SPL . This file is supposed to exist i.e. created previously. You can create an .SPL file with the EDITOR or you can use the program SHUFFLE to create it for you by choosing randomly among defined sound files, segments etc. ($SHUFFLE).

NOTE ! if you are using a .SPL file created by SHUFFLE, you have to have a PAUSE.DAC file on your DRA1: . You can COPY it from DRA1:[SAFE] Here followes an example of an .SPL file :

```
T5                        the name of the outputfile (T5.DAC)
TALK 20000 35000          the FORMAT for each subsequent line is :
TALK 2000 14000 500         INPUTFILE  STARTSAMPLE  ENDSAMPLE [OVERLAP]
TALK 1000 3000 100
TALK 1000 3000          NOTE ! you can write comments as much
TALK 60000 134000           as you like after the data required by
TALK 40000 45000 5000       the program. Data can be separated by any
TALK 100 50000              number of spaces, but NOT COMMAS.
```

NOTE ! The default value of overlap is 100 ms, or the last value defined for it. The value of overlap must be maximum half of the length of the preceding and following segment (i.e. no empty "tape" operations are allowed.)

REVERB

```
                      d1
                   '--------`
                    d2
            '---------------`
                 d3
      '---------------------`

I------------------------------X------------------------------I
                               *amp (direct lev.)    input file
                  -------------+-|
                 '     ---------+-|
               '  rl2*'     ---+-|
       rl3*'       '  rl1*'      |
         '      '       '        |
I--------*-----*------*--------Y------------------------------I
         t3    t2     t1                            output file
```

t tap
d delay of the tap in samplinges
X current samplings of the inputfile
Y resulting sample of the output file
rl reverbaration level

HOW MANY TAPS : (MAX 64 ; -1=FILE): define "N" number of taps OR write -1 to read in a .REV file created at a previous run.

DEFINE "N" DELAYS (SAMPLES): give PRIME NUMBERS as delay, otherwise you get "ringing" .

! A list of prime numbers (on LPA0:) is available for everyone by doing :

$@[sys2]prime

DEFINE "N" REVERBERATION LEVELS (NORMALISED AUTOMATICALLY):
You can give these levels in any scale you wish, their relationship counts only . They will be scaled to appropriate levels.

TO BE NORMALISED TO: means : GIVE OVERALL REV.LEVEL

DIRECT LEVEL (0-1):

NAME OF .REV FILE (RETURN FOR NO SAVE): saves your data on a file you name

You can create your own .REV file using the EDITOR. The data format is :

```
direct level,reverb level(normalisation factor)
delay1, unnormalised rev.level1 (relational data)
delay2, unnormalised rev.level2 (relational data)
          etc.
```

RING

Ring-modulates the contents of specified locations of two .DAC files, writing the result to a third file.

SPECSH

Spectrum shifter. This program analyses (Fourier transform) a .DAC file (input file), alters the frequency and amplitude characteristics of the file according to the following functions :

Function No.	
1	shift frequency bands
2	swap frequency bands
3	pre amplify a frequency band (i.e. before shift or swap)
4	amplify a frequency band on result (i.e. after shift or swap)
5	pre envelope (i.e. filter before shift or swap according to a 3 segment envelope given by you)
6	envelope on result (i.e. filter after shift or swap according to a 3 segment envelope given by you
7	window size
0	perform

Both of the analysis and re-synthesis part of the program divides the frequency spectrum of the files (from 0 Hz to samplingrate/2 Hz)into 4096 frequency components (FCMP) .The user must give all of the data, concerning the definitions of the frequency bands, in FCMP´s instead of Hz.

About the input data for the Functions :

FUNCTION		
all	INPUT	the name of the input .DAC file
	OUTPUT	the name of the output .DAC file

1 (Shift band .)

BAND-SIZE: the size of the frequency band to be shifted, in FCMPs

LOW COMPONENT: the starting FCMP of the band-size to be shifted

DISTANCE: defines the starting FCMP of the new, shifted band , calculated as :

new starting FCMP=LOW COMPONENT + DISTANCE

NOTE ! Shift does NOT erase the original bandsize to be shifted, it copyes it only to a new place.

FUNCTION

2 (Swap band)

BAND-SIZE : the size of the frequency band to be swapped, in FCMPs
BAND1 LOW: the starting FCMP for the first freq. band to be swapped
BAND2 LOW: the same for the second freq. band

3 (Pre-amplify)
BAND-SIZE: the size of the frequency band to be amplified , in FCMPs
LOW-COMPONENT: the starting FCMP for the bandsize to be amplified
AMPLITUDE: multiplication factor

4 (amplify result) the same as Function 3

5 (pre-envelope)
FOUR COMPONENT BREAK-POINTS:
four FCMPs to define the three segment envelope
AMPLITUDES - START , HOLD , DECAY:
three amplitude values

```
                                          *       amplitudes
                                          ?       breakpoints


                 *-------*
                /         \
               /           \
              /             \
             /               \
I-----------*-----------------*------------------------------I
0           ?    ?       ?    ?                            4096
FCMP
```

6 (envelope on result) the same as 5.

7 (window size)

WINDOW SIZE (POWER OF 2): you can change the default windosize (4096) to be 512 or 1024 or 2048

0 (perform) by calling this function , the program will calculate the outputfile according to the users definitions.

Example :
Suppose, we want to filter out the frequencies below 300 HZ .

```
                           *--------------------------------*-----------*
                          ,
                         ,
                        ,
  ---------------     ,
I---------------*------------------------------------------------I
0               ?    ?                             ?             ?
               123  140                          3000          4096
FCMP
```

NOTE ! NOT TO SCALE.

To be able to do this filtering, we have to use two functions. We define first the envelope part of the filter (function 5), after that we set all of the amplitudes between FCMP 0 and 123 to 0.

```
'---------------` '------------------------------------------------`
  funtion 3                              function 5
  (pre-amplify)                        (pre-envelope)
```

Our samplingrate is 20000, therefore the highest frequency is 10000 Hz at FCMP 4096. The difference between two consecutive FCMPs is 10000./4096. Hz, i.e. 2.441 Hz . To find the approximation for our FCMP number at 300 Hz we do :
300./2.441=122.9 , i.e. our FCMP is 123

FUNCTION	GIVEN VALUES	
5	123,140,3000,4096	four breakpoints in FCMP
	0,1,1	amplitudes
3	123,1,0	
0	=PERFORM	

SPEED

Is designed to alter the speed (sampling rate)of a named .DAC file according to a specified factor. It outputs a mono file.

```
              INPUT FILE         name of input file (no extension)
              FIRST SAMPLE       in input file
              LAST SAMPLE        in input file (-131070=END OF FILE)
              OUTPUT FILE        name of output file
              MAX NUMBER OF OUTPUT SAMPLES (-131070=UNLIMITED)
              MODE     0=fixed control
                       1-7=periodic function control with the waveforms
                           in the internal wavetable
              SAMPLING RATE in HZ of input file
 ======= IF  MODE has been defined     0
              give SPEED factor

 ======= IF  MODE has been defined    1-7
              give FREQUENCY in HZ of periodic function,     and
              SPEED LIMITS (-4096 TO 4096 ) for the function.
```

SPLICE

Splices two .DAC files, writing the result to a third file.

MODEL:

```
                 ----------------------
                 |                     `
                 |                       `
                 |                         `
                 |                           `
        first1  '-----------INPUT1-----------`  last1
                                               -----------------
                                             '                  |
                                           '                    |
                                         '                      |
                                       '                        |
                          first2  '---------INPUT2---------`  last2

                                   'overlap`


             ----------------OUTPUT-----------------------`
```

ROUTINES FOR SYNTHESIS

This package has been build to model the EMSDEV package of the PDP 15/XVM system and apply it for software synthesis.

Initialising.

TAPE

TAPE NR where NR is an integer less then 100. It initialises a synthesis file with file name TapeNR.DAC .e.g. TAPE03.DAC

"Studio" and system setup.

SRATE

Samplingrate

CHANS

Number of output channels (1-4)

CSTEP

Like ESTEP and GSTEP in EMSDEV. Do not change the default value.

PTRIG

Phase trigger.
FG NUMBER, 3 OUTPUT FG NUMBERS :

FGNR generator whose phase is to act as control

OUT1, OUT2, OUT3 generators whose phases are to be set to zero every time generator ´fgnr´ passes the end of one wave period. Generator ´FGNR´ is not automatically cleared by its own trigger, though it will be cleared if one of the ´out´ numbers is the same as ´FGNR´. However, the phase of generator ´FGNR´ is automatically MODULO´d to keep it within 0 and 1. Zero values indicate that no generator is to be controlled.
On command/data file :

```
PTRIG
1 12 13 14
```

FMCON

TYPE, FROM(GEN), TO(GEN/CD):

It is used to make connections between generators(to create FM and AM generators), or between generators and channel distributors. Multiple connections cannot be made or maintained with this routine: single (dis)connections only are made, the remaining connection points being cleared.

TYPE : 0 = disconnect,
1 = connect generator to another generator´s FREQUENCY input
2 = connect generator to another generator´s AMPLITUDE input
-1 = connect generator to channel distributor

On command/data file :

```
FMCON
1 1 2
FMCON
-1 34 2
```

MULCON

Makes multiple connections from one generator output to 8 device inputs (generator frequency, generator amplitude, or channel distributor).

FG NUMBER, 8 CONNECTIONS :

FGNR generator whose output is to be connected

CONNECTION connection data, each word means:

```
-N           = connect to channel distributor N
0            = disconnect
0 ´ N ´ 57   = connect to FREQUENCY input of generator N
56 ´ N ´ 113 = connect to AMPLITUDE input of generator N-56
N ` 112      = error
```

On command/data file :

```
MULCON
12 -1 -2 13 14 0 0 0 0
```

FGT

NUMBER, TYPE:
assignes generator type to generator number.

Type :
0 = inactive
1 = smooth wave-form with intermediate values interpolated between the defined points
2 = no interpolation; defined values kept until new value encountered
3 = pulsed; defined values in wave are active only for a specified percentage of time. Otherwise wave gives 0.
e.g. 903 = wave active 90% of time
153 = wave active 15% of time

On command/data file :

```
FGT
22 3
```

CLEMS

Program that clears:

generator frequency and intensity
amplifier intensities
all connection points
generator "active" flags

CDA

Sets amplitudes on channel distributor amplifiers.

CD no	channel distributor number (1 - 8)
CHANNEL no	channel number (1 - 4)
AMPLITUDE	amplitude (0. - 1.)

On command/data file :

```
CDA
1 1 1
```

FGF

Sets frequency (0. -samplingrate/2.) on generator number (1-56)
NUMBER, FREQUENCY :

On command/data file :

```
FGF
1 234
```

FGA

Sets amplitude (0.='amp'1.) OR modulation index on generator number (1-56).
NUMBER, AMPLITUDE(0.='amp'1.) OR MOD INDEX (0-100):

On command/data file :

```
FGA
1 0.98
FGA
2 56.7
```

N O T E ! ! ! Do not use amplitude value 1., give 0.999 instead.

FGW

Puts a proportional value in the range 0 to 1 on one of the 7 wave-forms for a specified generator (you can mix different spectrums).

GENERATOR NUMBER
WAVE FORM NUMBER (1 - 7)
PROPORTION (0 - 1) TO BE SET ON THIS WAVE

On command/data file :

```
FGW
2 6 1
```

N O T E ! ! ! ALL of the generators have the default wave 1 . To be able to assign a NEW waveform you should DEASSIGN first like :

```
FGW
2 1 0          deassign
FGW
2 6 1          assign
```

NOTE ! ! ! You can mix the available waveforms by assigning different proportions, like :

```
FGW
2 1 0          deassign
FGW
2 3 .5         assign 50%
FGW
2 5 .5         assign 50%
```

FGP

NUMBER, PHASE (REAL):

sets an absolute phase value in the range 0.-1., on a specified generator. Values outside this range are automatically adjusted (with 'MOD')

GENERATOR NUMBER (1 - 56)
PHASE VALUE (0. - 1., NORMALLY)

on command/data file :

FGP
56 0.5

FM

NUMBER,MOD FREQU,MOD IND,FREQU,INTENSITY(0-100):

stores fm parameter information.

GENERATOR NUMBER (1 - 16)
MODULATION FREQUENCY
MODULATION INDEX
CARRIER FREQUENCY
INTENSITY IN DB (0.0 - 100.0)

generator parameter information is put into the same tables as are used by 'FGQ'. FM(1) uses positions for FG(25) and FG(26); FM(2) uses positions for FG (27) and FG (28); etc.

when 'TAPE' is called, these generators are automatically connected to form simple frequency modulation generators; The outputs of the carrier generators are connected by default to CHANNEL DISTRIBUTOR 1.

INTENSITIES OF ZERO ARE CONVERTED TO AMPLITUDE ZERO

on command/data file :

FM
16 200 10 400 90

TIME

writes music information (amplifier levels, generator frequencies and levels, etc) to sound for a given number of milliseconds.

ENDPLY

it closes music sound files and returns information about duration and number of studio samples.

EXAMPLE OF SYNTHESIS

EX1

Suppose, we want to generate 10 notes.

The first note shall have a SINUS waveform amplitude modulated by an envelope like

```
      *
     * *
     * *
     * *
     *  ***
    *      ****
   *           *******
```

The last tone shall have a TRIANGLE wave form amplitude modulated by an envelope like

```
        *
       * **    *
      *    *** *
     *          **
    *             *
   *               *
```

The notes in between shall change their waveform and modulation characteristics to achieve a gradual transformation. They shall have a slight vibrato, i.e. frequency modulation.

The instrument used is

```
 --------
| OSC2   |
|        |----|
 -------      |
              |Frequ.
              |-----|
                    |
                    V
 --------        --------
| OSC1   |  Ampl |OSC3   |
|        |-------`|       |------`CD
 --------        --------
```

```
Functions          The values given         Explanation

DRAW               to make 2 waveforms for ampl. modulation. We call them
                   A1 and A2 . NOTE ! The curves must be drawn to lie
                   between -1 and 0.

ASSW                                assigns A1 to external wave 7
                                    assigns A2 to external wave 6

TAPE               to initialise synthesis and  .DAC file .
        1          Note ! According to the input value "1", the
                   name of the output file shall be TAPE01.DAC
CONN                                OSC3 connected to CD 1
        3       1
FMCON      2       1        3
FMCON      1       2        3
CDA
        1       1       1.
FGF       1       1.
FGA       1       1.
FGF       2       4.                  vibrato frequency
FGA       2       0.02                vibrato amplitude, i.e. mod.ind.
FGF       3       400.                fundamental frequency
FGA       3       1.

FGW       1       7        1.
FGW       1       1        0.   get rid of the original sinus wave
    NOTE ! OSC2 and OSC3 have sinus waveforms automatically.
TIME        980     NOTE ! We defined 1 HZ (1000  ms) for the amplitude
                    modulation (duration of the envelope) however the
                    program reaches the end of the table slightly earli-
                    er (20 ms !). In order to avoid continued generation
                    (starting a new envelope at the end of the note) we
                    give 980 ms for duration.
FGP       1       0.             Set phase to zero before generating
                                 the next note.
FGW       1       7        0.9       change waveform proportions
FGW       1       6        0.1       of amplitude modulation
FGW       3       1        0.9       change waveform proportions
FGW       3       4        0.1       of SINUS and TRIANGLEL.
TIME       980
FGP       1       0.                  Set phase to zero .
FGW       1       7        0.8
FGW       1       6        0.2       change waveform proportions
FGW       3       1        0.8
FGW       3       4        0.2       change waveform proportions
TIME       980
FGP       1       0.                  Set phase to zero .
FGW       1       7        0.7
FGW       1       6        0.3       change waveform proportions
FGW       3       1        0.7
FGW       3       4        0.3       change waveform proportions
TIME       980
FGP       1       0.                  Set phase to zero .
:
```

```
:
FGW        1       7       0.0
FGW        1       6       1.0          change waveform proportions
FGW        3       1       0.0
FGW        3       4       1.0          change waveform proportions
TIME        980
PLAY
TAPE01                         the name of the .DAC file to be played
```

MISCELLANEOUS

CALL

Calls a combined command and data file. The file must consists of BADA commands and data following each other in the same order as they are given interactively . When certain subroutines asks internal questions, the processing of this command/data file is interrupted temporarely until the user supplies the program with needed data FROM THE TERMINAL.

NOTE ! 1.) CONVOL does NOT work calling it from this file

COUNT

Switch to display the current sample number during processing. Default is OFF . When OFF, CTRL T displays the current number. DURING execution of a function CTRL N can be used to change the switch. NOTE ! CTRL T does NOT work within the WSP package of BADA.

EXIT

Exits from BADA

INTRPT

The execution of the BADA package may be interrupted by CTRL Z any time. This facility can be turned OFF by assigning the value 0 .It can be usefull when you run BADA in batch.

LENGTH

Displays the length of a soundfile (.DAC)

LEVEL

Displays the maximum level of a soundfile (.DAC)

PLAY

Plays .DAC files, (file names (8) with space between, plays after each other successively.

TWAVE

Plays .WAV files which should be situated on DRA1: .

The Program WSP within BADA

You enter the VAX version of the program package WSP. To be able to work with WSP within the BADA package you have to follow the procedure below .

1. initialise
1.1 open sound file (TAPE)
1.2 fill waveform tables (ASSWAV) with .WAV files if you want to change the default wavetables
1.3 assign waveforms to generators (FGW)
2. connect (FMCON)
3. set amplitude on channel distributors (CDA)
4. call WSP
5. work with WSP
6. exit WSP
7. listen to the result by calling PLAY
8. call WSP
9. work with WSP
10. exit WSP
11. listen to the result by calling PLAY
12. close soundfile (ENDPLY)

NOTE ! CTRL Z does NOT work within WSP. You have to write EXIT instead.

To learn WSP you have to study the WSP MANUAL first.

In order to help you in practical programming, there a couple of files collected on the directory [WSPBADA] . Copy them to your directory .
These files can be grouped into different pedagogocal stages:

Stage 1.

TEST1.WSP followed by TEST2.WSP and TEST3.WSP.
They have corresponding .BAD files to initialise BADA.
Study even BADAWSP.TXT . It contains information about these files.

Stage 2.

TFM1.WSP , TFM2.WSP , TFM3.WSP and TFM4.WSP .
They have corresponding .BAD files to initialise BADA and corresponding RUNFM1.WSP , RUNFM2.WSP , RUNFM3.WSP and RUNFM4.WSP which should be called after the TFMx.WSP files. They triggers the generation.

Stage 3.

BASFMV1.WSP , SEQUFMV1.WSP together with RUNFMV1.WSP ,INITFM.WSP and corresponding FMV1.BAD .

Stage 4.

Study the file SUGGEST.TXT . Print it out.