INTERACTIVE

MUSIC

PERFORMANCE

AND

COMPOSITION

designed and programmed by

MICHAEL HINTON

Stockholm, December 1979

Acknowledgements

Many thanks to ...

Erik Nyberg, the inventor of real-time control at EMS, who, apart from writing special versions of his direct-play and TV-handler programs (DIP1 and TVEMS) for IMPAC, sacrificed a great deal of time in order to instruct me in the mysterious ways of the PDP-15/XVM;

Tamas Ungvary, who gave encouragement and wrote a preliminary draft summary of the program commands;

Lars-Gunnar Bodin and Peter Shore, who read the manuscript of this manual;

Ann, who was patient.

Contents

1 Introducing IMPAC

1.1 Principles

IMPAC is a computer-based system for musical performance and composition. Its design is based on three fundamental principles.

1.1.1 The music is created primarily by direct control over compositional parameters such as pitch and amplitude range, and density; from these the program calculates the actual instrumental activity on frequency generators.

1.1.2 The system is easy to use, requiring no previous knowledge of computers or programming; it can be mastered within a matter of hours. The only requirement is that the user has basic knowledge of the functions and operation of the EMS studio; this includes the computer peripherals such as dec-tape units, magnetic-tape stations, and the Tektronix display terminal.

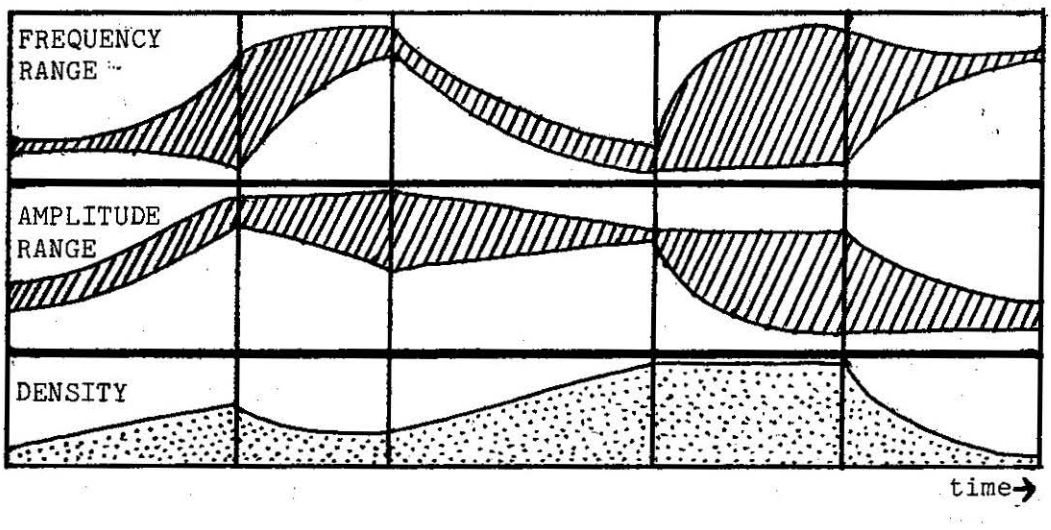
1.1.3 A fast interactive approach is adopted to allow the user to respond immediately to what he hears; this extends to real-time manual control over the compositional parameters.

1.2 Working with the program

IMPAC has two interdependent parts, the composition program and the real-time program. It is possible to work with one of these elements without reference to the other: the user can, for example, choose to work with compositional structures only, exercising no real-time control whatsoever; or he can use the real-time facilities alone, in which case the system functions purely as a musical instrument. But the most powerful feature of the system is that these two methods can be combined: composed structures can be played and modified in real-time.

1.2.1 In the composition program the user creates musical structures, or BLOCKS, which can be played at once, altered in various ways, and mixed with other BLOCKS.

The score for a compositional BLOCK would typically look like this:



Frequency and amplitude are to lie within the limits described by their respective curves; density means "number of notes per second" - so a low value gives a sparse texture with few notes sounding.

Working procedure would typically follow this pattern:

1) Define parameters for one BLOCK.
2) Listen to the BLOCK - it is automatically stored as a file which can be retrieved at any time.
3) Make necessary alterations to any of the parameters. Each time an alteration is made a new file is created and stored. Go back to (2) and repeat until a satisfactory result is achieved.
4) Listen to all of the files so far created as a further check, and note the number of the file(s) to be used in the final composition.
5) Repeat (1) - (4) for all compositional BLOCKS.
6) When all BLOCKS have been thus defined and tested, define the sequence in which the selected files are to be played, together with their start-times if they are to be mixed vertically.
7) Listen to the final result.

For the example on p.1, this whole procedure should not take longer than a few minutes. It is therefore possible to realize and test quite long and complicated structures in one 2-3 hour session with IMPAC. Moreover, BLOCKS and combinations of BLOCKS can be stored on dec-tape for future working sessions with the program.

1.2.2 In the real-time program these parameters, and others which define timbre, vibrato, attack-time, and so on, can be controlled manually with the help of the Joystick, the Digitizer and one of the manuals on the Tektronix terminal, which functions as a twelve-key keyboard. Or they can be controlled by four programmed pseudo-generators, producing random, sinus, triangle and square-wave oscillations; these can, in turn, be controlled manually by the user. So, for example, at the same time the user might be changing frequency limits with the Joystick X and Y axes respectively, amplitude with the sine-wave generator (whose frequency is being controlled by the Tektronix keyboard), attack-time with a combination of random generator and the digitizer's X-axis, reverberation with the triangular wave generator, and so on.

And finally, it is possible to exercise a more structured control over some of the parameters with BLOCK-definitions made with the composition program, while at the same time controlling other parameters manually.

1.3 System development

IMPAC has been developed, modified and expanded over a period of a few years. At first it was nothing but a primitive version of the composition program, which wrote music information for the twenty-four frequency generators to magnetic tape; the studio facilities allowed for no real-time control, and feedback from composition to composer was unbearably slow. But gradually it was realized at EMS that most composers require a direct contact with their music, and recently several advances have been made in that direction. These include Erik Nyberg's program DIP1, which allows the computer to control the studio devices at the same time as it calculates music, and, on the hardware side, the installation of a manually controlled Joystick, a Digitizer, and a number of FM generators. IMPAC has all the time been expanded to include the new facilities, and it is expected that it will continue to develop in the future at the same pace as the studio itself.

2 Formalities

2.1 The disks

IMPAC reads information from two disk areas:
1) DK <IMP>, which is a pre-defined area containing all the files belonging to the IMPAC system;
2) A user-defined disk area.

2.1.1 The following files are on DK <IMP>:

```
IMPAC XCT
IMPAC XCU
IMPAC BAT
0     BIN
0     CON
1     CON
2     CON
3     CON
4     CON
```

IMPAC XCT and IMPAC XCU are the EXECUTE files which contain the IMPAC system.

IMPAC BAT is a BATCH file which performs some necessary operations at the beginning of a run.

0 BIN is an OBJECT or BLOCK file, containing default values for the music parameters; if the user plays music with IMPAC before he has defined any parameters, the musical result is governed by the information in this file. BLOCK files are described in more detail in §§4.2 ff, 0 BIN in §5.1.2.

0 CON - 4 CON are connection files: they contain the information needed to make various device/amplifier connections in the studio. They are described in detail in §2.4.

2.1.2 As with all other programs and systems, the user must define part of the computer's disk as his own working-area. (Note that "$" is never written by the user - it is written by the computer to show that the system monitor is ready to receive commands.)

```
$LOGIN USE
```

where USE can be any combination of three letters except SCR, EMS, or IMP.

N.B.

All user information files created during a run will be on DK <USE> when the run is complete.

If the user has files created on previous runs which are to be used during the current run, he must ensure that they are on the defined working-area, not DK <IMP>.

2.2 Starting the program

To start IMPAC
a) write on the decwriter (TT) after the monitor's $-sign:

```
B DK <IMP> IMPAC
```

b) turn on the main studio switch and the Tektronix.

2.2.1 When the command in (a) above is written, the computer reads in a series of instructions from the BATCH file "IMPAC BAT" on DK <IMP>. The following text is then printed automatically on TT:

```
$B DK <IMP> IMPAC

XVM/DOS V1A000

$$JOB
$PIP
PIP XVM V1A000                                          (1)

>T DK_DK <IMP> 0 BIN

>
XVM/DOS V1A000

$$JOB
$BUFFS 4                                                (2)

$A DK 14,15,16,17/DK <IMP> -4,1/TW1 4,11,20             (3)

$A NON 2,3,5,6,12,13                                    (4)

$E IMPAC
EXECUTE XVM V1A000
```

(1) PIP is called, in order to transfer the default BLOCK file from DK <IMP> to DK <USE>. Note that if there is already a file called 0 BIN on DK <USE>, it will be destroyed.

(2) The command "BUFFS 4" increases the amount of core available to IMPAC.

(3) Assignments are made here to instruct the system monitor that:
   a) DK <IMP> contains the required EXECUTE files (-4), and the CONNECTION files (1).
   b) DK <USE> will be used for reading and writing all user files (14, 15, 16, and 17).
   c) TW1 (Tektronix) will be used for user commands, interactive data input, error messages, etc. (4, 11) and for printing user files with the WR command (20).

(4) .DAT slots not used by IMPAC are assigned to NON; this is another way of ensuring that IMPAC has sufficient core for buffers and data.

N.B.
The assignment "MTO 7" is assumed, since this is standard for all programs at EMS.

2.2.2 It may occasionally be desirable or necessary to alter the above assignments.

1) If MT1 is to be used instead of MT0, write:

```
$K ON

$A MT1 7

$B DK <IMP> IMPAC
```

and at the end of the run, after exiting from IMPAC, write:

```
$K OFF
```

2) If interactive input/output is to take place at the decwriter instead of the Tektronix, TW1 must not be assigned to 4,11. Write:

```
$BUFFS 4

$A DK 14,15,16,17/DK <IMP> -4,1/TW1 20

$A NON 2,3,5,6,12,13

$E IMPAC
```

3) If the WR command is to print BLOCK files on the line-printer instead of TW1, write:

```
$BUFFS 4

$A DK 14,15,16,17/DK <IMP> -4,1/TW1 4,11

$A NON 2,3,5,6,12,13

$E IMPAC
```

N.B. It is important to note that when alternative 3 is used, or when any other deviation from the required assignments is written, there will not be sufficient space in core for the magnetic tape (MT) routines to be able to function. Any attempt to use MT will result in the error message being written:

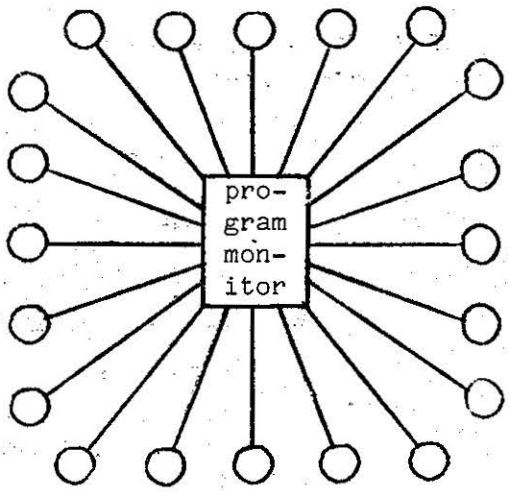
```
NO ROOM FOR MT
```

However, if the user is certain that he is not going to write to MT during a run, then the instruction "BUFFS 4" and the assignments to NON can be left out.

It has been stated that interactive input/output takes place on .DAT slots 4 and 11, and that these units can be assigned to either TW1 or TT. Note, however, that the input/output referred to here does not include the functions that can be controlled in real-time; this control can be exercised only from the Tektronix (TW1).

2.2.3 The clock switch on the computer console must be UP (off) all the time during an IMPAC run. If the switch is on, the run will probably be interrupted at unexpected moments by IOPS error messages on the decwriter.

2.3 The IMPAC program monitor



This is the part of the IMPAC system that reads user commands, checking their validity and issuing error messages when required; when a legal command is given, this monitor invokes the appropriate subprogram which is to carry out the command.

2.3.1 Input commands are written in one of the following formats:

1) >XX

2) >XX n1

3) >XX n1 n2

where: > is printed by the computer to show that the program is ready to receive a command;
XX is a mnemonic consisting of two letters, or one letter and a number (e.g. F1, A2), or one letter only.
n1 and n2 are numbers defining the command more closely.

Examples of the three types:

1) >EX "Exit" from IMPAC; this is the instruction written at the end of a run.

2) >WF 4 frequency generators are to have wave-form 4 (sawtooth).

3) >FG 1 12 music is to be played by generators 1 to 12 inclusive.

Commands are closed with carriage return.

N.B.

- If no value is given to n1, it is assumed to have the value 0. Thus:

  >WF

  and:

  >WF 0

  are from the program's point of view identical.
  The command

  >FG

  is therefore illegal, since no generator has the number 0.

- The only characters that may appear in n1 and n2 are: 0123456789-; if any characters other than these are written, the number is put to zero.
  Thus:

  >FG X

  is interpreted as:

  >FG 0

  which is illegal. However:

  >WF X

  is interpreted as

  >WF 0

  which is allowed. There is therefore no error message.

2.3.2 In some cases, a command to the program monitor is answered with

:

instead of

>

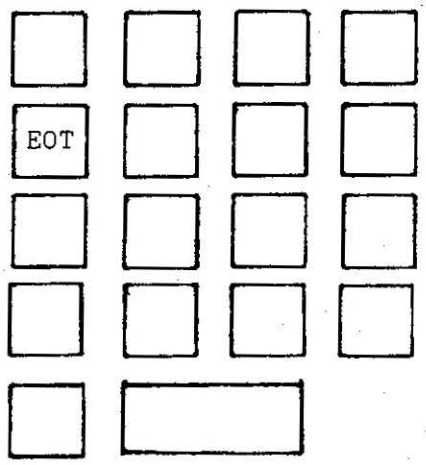
This indicates that a SUBMONITOR has temporary control; each submonitor has its own set of instructions and formats, and will keep control until the command "return to program monitor" is given. "ED", for instance, which is used to edit compositional blocks, has fifteen instructions, some of them similar to the program monitor commands - e.g. A2, which alters definitions of the A2 curve - while others are used only in ED; "AB", for example, means "abort", and is used to cancel any alterations made, i.e. to retain the original block definitions.

>ED 10	Edit block number 10.
:A2 -8	A2 curve is to be 2 dB less than it was originally.
:	Empty line means "complete editing and return to program monitor".
10 BIN OK	Program answers that editing is complete.
>	Program monitor has control.

The mnemonics and instruction formats for each submonitor are described in the relevant sections of chapter 4.

2.3.3 The function key EOT on the Tektronix has a number of special uses; it always:

1) cancels everything that has been written in the current line;
2) clears the screen;
3) moves the cursor to the top left hand corner of the screen.



If conversation takes place on TT instead of TW1, the place of this function is taken by control D (üD); in this case only (1) above is performed.

When the submonitors have control, EOT has various uses in addition to the three already mentioned. In "ED", for example, it performs the same function as AB (see above); after the "SC" command (scale definition) it signals "end of definition and return to program monitor".

Explanations of the special uses of EOT are given in the individual descriptions of commands in chapter 4.

2.3.4 Comments can be written on any line that begins with the program-monitor's >-sign.

1) If an instruction is written first in the line, the comment must be preceded by at least one space. For example:

```
>FG 1 40 USE ALL GENERATORS
```

2) If no instruction is written, the comment must be preceded by at least two spaces.

```
>  THIS IS A COMMENT
```

Note that the program ignores comments completely.

2.3.5 The command "PL" (PLAY, cf §4.2.1) hands over control to the real-time submonitor. Here the keys on the Tektronix have completely different functions from the characters normally associated with them; consequently, when a key is pressed, the character is not echoed on the screen. Instructions are given by pressing sequences of one, two or three keys. Return to program monitor is achieved by pressing the SPACE key.

The real-time program and its command set are discussed in detail in chapter 5.

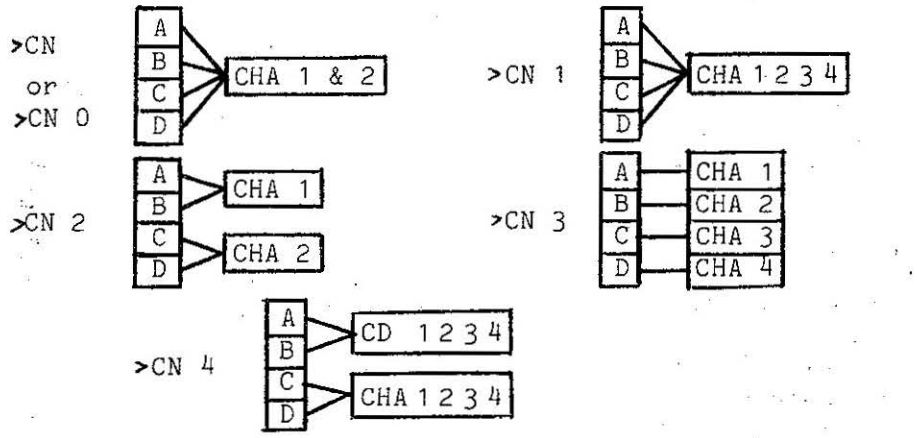
2.4 Studio connections

The command "CN" makes connections in the studio.

2.4.1 The forty generators can be thought of as being divided into four groups:

A: FGs 1 to 6 and FM oscillators 25 to 28.
B: FGs 7 to 12 and FM oscillators 29 to 32.
C: FGs 13 to 18 and FM oscillators 33 to 36.
D: FGs 19 to 24 and FM oscillators 37 to 40.

"CN" can be used to connect these groups of generators as follows:



Here CHA stands for "channel output amplifier", and CD stands for "channel distributor".

Note that connections to appropriate reverberation units are made at the same time as the channel output connections; intensities of 100 dB are also set on the channel output amplifiers.

2.4.2 Information about the different ways of connecting generators to studio outputs is contained in the CON files on DK <IMP>; the number written after the command "CN" refers to the numbers (0-4) of these files.

It is possible for the user who can write FORTRAN programs to create his own connection files. These are given names of type "n CON", where n is a whole number in the range 5 to 99999; and they can be called with the "CN" command in exactly the same manner as the standard connection files.

The techniques used in creating "CON" files are described in §6.9.3.

2.4.3 Intensities and decay times can be set on the reverberation units with the command "RV". Format:

```
>RV n1 n2
```

where n1 is the decay time, from 0 to 15;
n2 is the intensity, from 0 to 400 (1 unit = $\frac{1}{4}$ dB).

These values are set immediately on all four reverberation units.

2.4.4 All connections and amplifier intensities, as well as generator frequencies and amplitudes, are cleared with the command:

```
>CL
```

To change connections which have been set before, write:

```
>CL
>CN n
```

where n is the number of the new connection file to be read in.

The studio can also be cleared manually with the "nollställ" button; similarly, connections and amplifier intensities can be set in the studio by hand.

2.5 Magnetic tape

The command "MT" functions as a switch to enable/disenable storage of music on magnetic tape.

2.5.1 To enable storage on magnetic tape, write:

```
>MT
```

or

```
>MT n
```

where n is a positive whole number.

After this command has been given the MT-switch is ON, and music played in the studio with "PL" is written to magnetic tape at the same time.

2.5.2 To disenable magnetic-tape storage, write:

```
>MT -1
```

This turns the MT-switch OFF. Note, however, that it does not cause a STOP-mark to be recorded on magnetic tape - see §2.5.5 below.

2.5.3 When the MT-switch is on, the following commands cause information to be written to magnetic tape as well as to the studio:

CN, RV, CL, and PL.

Connections must be written to tape before any other information is written there. If any attempt is made to write music before connections, the message:

CONNECT MT!

appears on TW1 (or TT), and control is returned to the program monitor. For example:

>CN 2	make connections in studio
>RV 10 320	set reverb decay and intensity in studio
>MT	turn on MT-switch
>PL	attempt to play music
CONNECT MT!	but connections were written before MT command
>CN 2	this goes to tape as well as studio, since MT-switch is on.
>RV 10 320	put reverberation on tape
>PL	now we can play music to studio and magnetic tape

2.5.4 IMPAC has no facilities for playing back information recorded on magnetic tape; this is done with the program A2 at the end of an IMPAC run.

A2 plays magnetic tapes from the beginning of the tape until a STOP-mark is found; the CONTINUE command is then used to play the tape from this point until the next STOP-mark is found; and so on.

2.5.5 In IMPAC, a STOP-mark is written to tape with the command:

>CT (= CLOSE TAPE)

It is also written automatically at the end of a run in which magnetic tape has been used but no "CT" has been issued.

"CT" turns off the MT-switch automatically; to record more information on magnetic tape, the user must therefore write:

>MT

>CN

etc.

2.6 Completing the run

To complete a run with IMPAC, write:

>EX

This clears the Tektronix screen, and returns control to the system monitor.

3 The Music Parameters

Music parameters in IMPAC are of three main types: primary, note-defining, and spacial.

3.1 Primary parameters are those which define frequency range, amplitude range, and density.

3.1.1 DENSITY - DE - represents the statistical probability of a given number of new notes being started per second. Thus, if DE = 1, a new note will be started roughly once every second; if DE = 100, there will be about a hundred new notes every second. However, these notes will not start at regular time intervals, since density is by definition a random process.

Legal values for DE: 0 to 2000

N.B. 1) A density of zero (DE = 0) means that no new notes are started; however, notes started before density is set to zero continue sounding for their full duration.

2) It is unlikely that a density as high as 2000 can ever be achieved when music is played directly in the studio; the time required for calculating the music will in practice always restrict the density to about 700 or 800 notes per second.

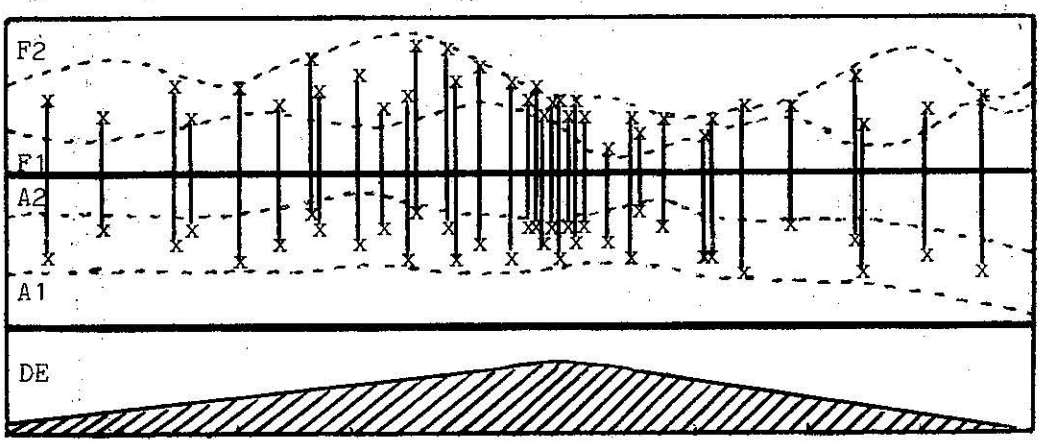
3.1.2 FREQUENCY RANGE is defined by two values, F1 and F2, which determine the limits within which frequencies are to be chosen. Exact choice of pitch is made by a random number generator, but the user can determine the random distribution - rectangular or exponential - and the type of interval relationships (scales) to be used.

Legal values for F1 and F2: 0 to 15999 Hz

3.1.3 AMPLITUDE RANGE is defined by A1 and A2; amplitudes on individual notes are chosen by the random number generator between these two limits.

Legal values for A1 and A2: 0 to 400, where each unit is equivalent to ½dB. Thus 100 = 25dB, 400 = 100dB, and so on.

3.1.4 The main structural principle of IMPAC is thus that timing of notes is controlled by DE on a probability basis; pitch and intensity are set at random between limits defined by the user.

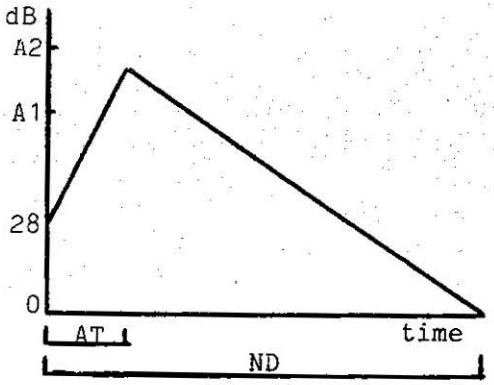


3.2 Note-defining parameters

Once DE, F1, F2, A1 and A2 have determined that a note is to start at a particular time and with a given pitch and intensity, the character of the note is formed by the note-defining parameters.

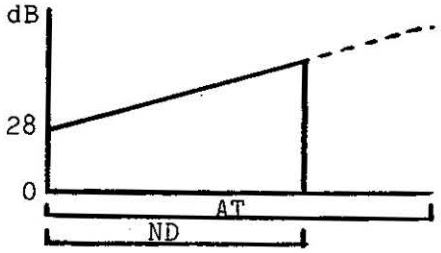
3.2.1 NOTE-DURATION (ND) defines the length in milliseconds of each note.

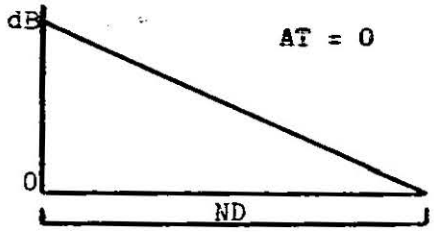
Legal values: 10 to 131000 ms.



3.2.2 ATTACK TIME (AT) is an amplitude envelope factor; it defines the time in milliseconds that it takes for a note to reach its maximum amplitude, which is the value chosen at random between A1 and A2.

N.B.
1) If AT is longer than ND, the note will stop before it reaches maximum intensity. And if AT is zero, the note will start at maximum amplitude.





2) The amplitude at the beginning of a note is 28dB unless:
   a) the generator in question already has an amplitude greater than 28dB; in this case the start amplitude will be 4dB less than the generator's current amplitude.
   b) AT is zero (see point (1) above).

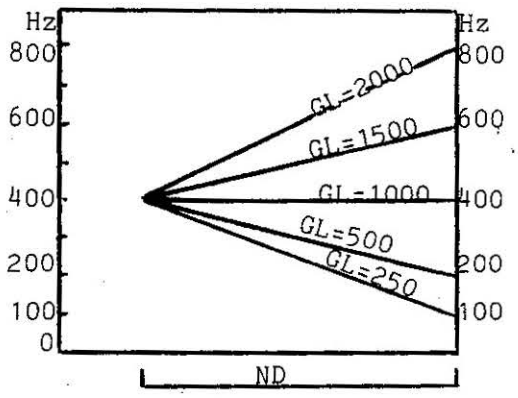
3) Envelopes are calculated on a linear basis. Amplitude decay is such that it reaches zero at the end of ND; this means that a note may become inaudible before it has sounded for its full duration.

Legal values for AT: 0 to 131000 milliseconds.

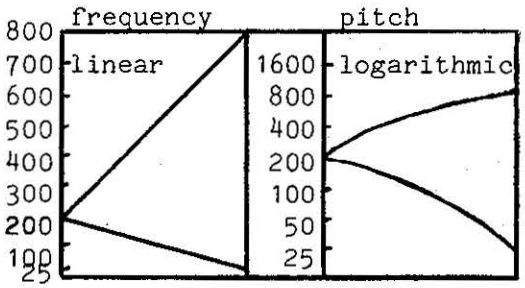
3.2.3 GLISSANDO (GL) defines the relationship between the frequencies at the beginning and end of a note. Values are given "per mille", in such a way that:

1000 = no glissando
2000 = glissando up one octave (start freq. x 2.0)
8000 = gliss. up three octaves (start freq. x 8.0)
500 = gliss. down one octave (start freq. x 0.5)
250 = gliss. down two octaves (start freq. x 0.25)

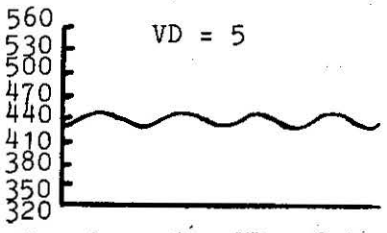
Legal values for GL: 0 to 8192.

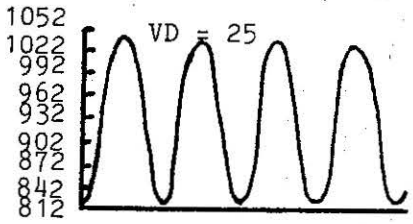


N.B.
GL is calculated on a linear basis, which means that rising glissando seems to move faster at the beginning of the note, while descending glissando is faster at the end of the note.



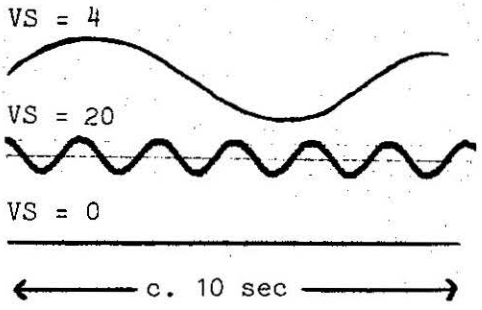
3.2.4 VIBRATO is defined by VD and VS (vibrato depth and vibrato speed).
1) Values for VD are given in percent, such that the difference between the vibrato's highest and lowest points is "VD"-percent of each note's basic frequency. For example:
- If VD = 5 and a note's frequency is 440, then the depth (amplitude) of the vibrato will be: 5/100 x 440, that is, 22 Hz. Vibrato will therefore be between 429 Hz and 451 Hz, 11 Hz on each side of the basic frequency.
- If VD = 25 and frequency is 932, then vibrato depth will be 25/100x932 (= 233 Hz). Vibrato will be between 815.5 Hz and 1048.5 Hz.





Legal values for VD: 0 to 1000.

2) Vibrato is calculated by looking up values in the SINE-WAVE table (cf §6.10). VS determines the speed at which a pointer is to go through this table; thus
- if VS = 4, this pointer is moved in turn to every fourth value;
- if VS = 100, every hundredth value is taken;
- if VS = 0, the pointer, and thus vibrato too, come to a halt.



Legal values for VS: 0 to 2880.

3.2.5 TIMBRE is defined by:
1) WF for analogue frequency generators;
2) MI and MF for digital frequency-modulating generators.

WF stands for WAVE-FORM; the values which can be set on the frequency generators are:

0 & 1	sinus	2	parabolic
3	triangular	4	saw-tooth
5	square	6	single pulse
7	double pulse		

WF has no effect on the FM oscillators.
Legal values: 0 to 7.

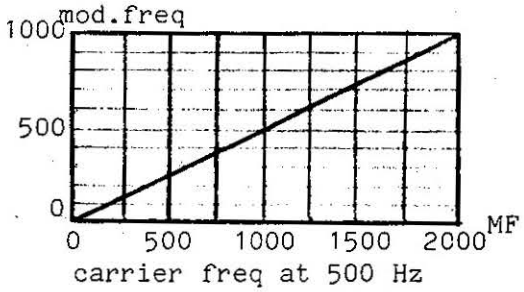
MI and MF stand for modulation index and modulation frequency.

The user defines MI as modulation-index x 100 (e.g. to get an index of 1.5, the user gives MI the value 150.

MF represents, not the actual modulation frequency, but the ratio between carrier and modulation frequencies. Values are assigned "per mille"; for example:

1000: mod freq = carrier freq
2000: mod freq = car freq x 2
500 : mod freq = 0.5 x car freq
100 : mod freq = 0.1 x car freq

Legal values for MI: 0 to 4095
Legal values for MF: 0 to 2000

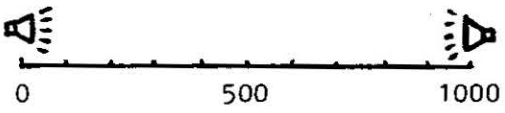


3.3 The spacial, or room-defining parameters are moving-sound and reverberation.

3.3.1 MOVING-SOUND (MS) has two slightly different functions:

1) In the real-time program it controls the positioning of sound between two of the studio's loud-speakers. If MS = 0, the sound is located at one of the loud-speakers; if MS = 1000, it is located at the other; if MS = 500, it is located at a point equidistant from the two speakers; and so on.

Legal values for MS: 0 to 1000.



2) In connection with the MIX facility (§§4.5 ff), MS can be used to control the quadraphonic location of sound; here the user describes movements in terms of room-radius, angle, and acceleration.

3.3.2 REVERBERATION (RV) also has various functions:

1) By giving the command RV from the program monitor, the user can set both times and levels on the reverberation units. See §2.4.3.

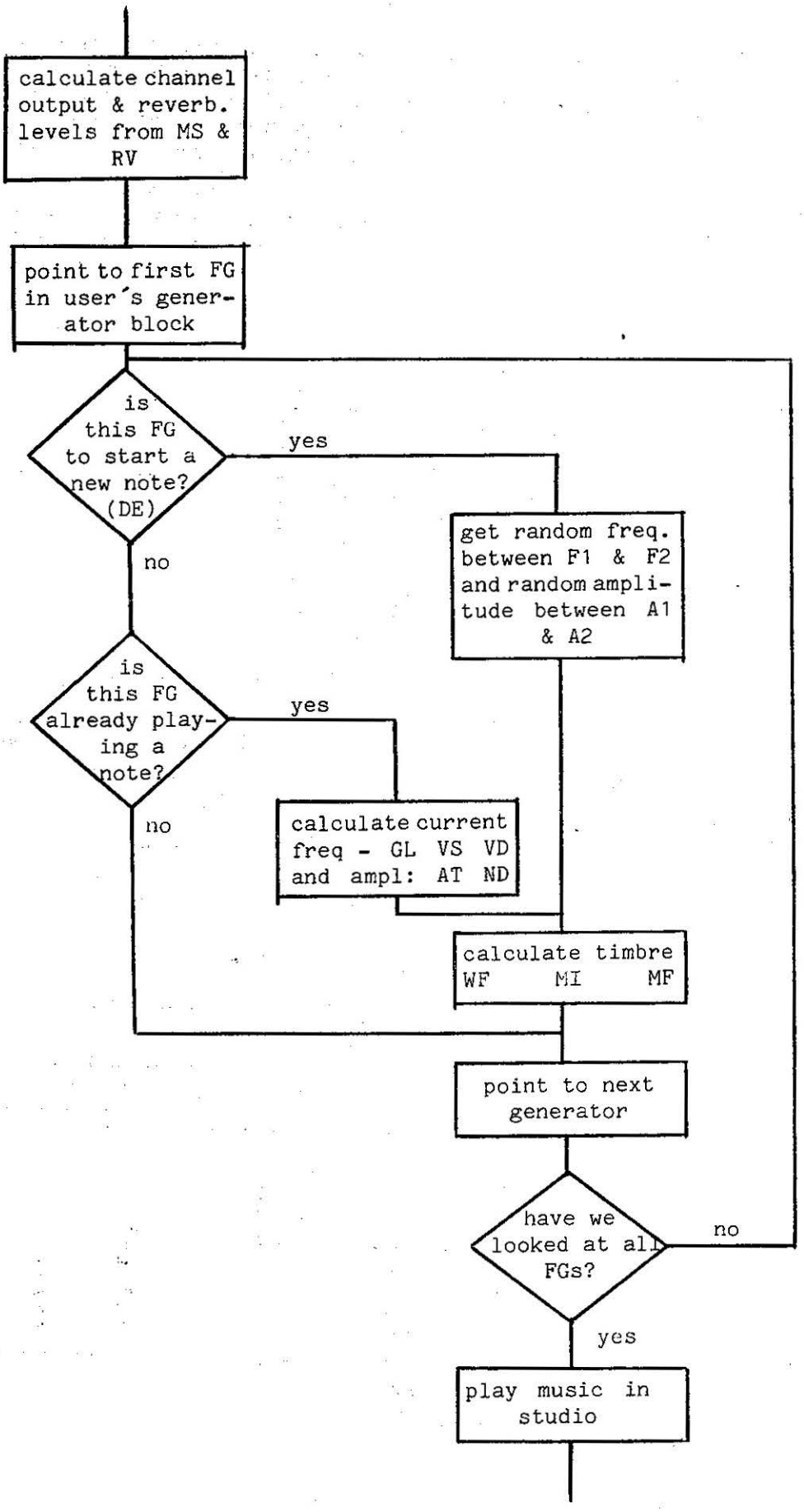
2) In the real-time program RV controls reverberation levels only. Legal values: 0 to 400, where each unit is equivalent to ¼ dB - 100 = 25 dB, 400 = 100 dB, etc.

3) It is also possible to set reverberation levels with connection files - see §2.4.

3.4 Miscellaneous parameters

There are a number of other parameters which do not easily fit into the above categories.

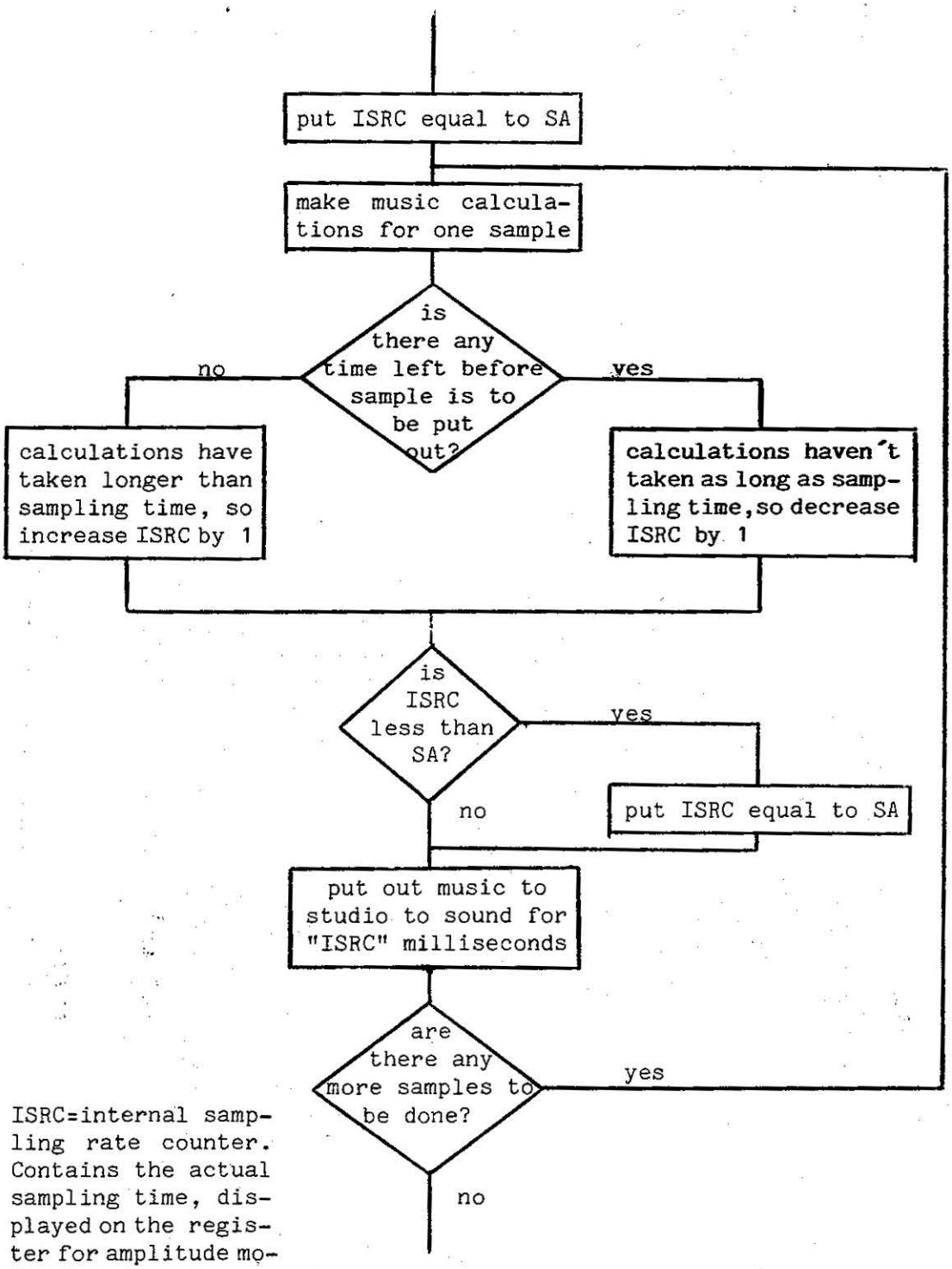
3.4.1 FG (FREQUENCY GENERATORS) defines the number and type of generators to be used: the analogue frequency generators are numbered 1 to 24, while the digital FM oscillators are numbered 25 to 40. Normally the user defines a block of generators, e.g. 1 - 12, or 7 - 10, or 25 - 36, or 20 - 30. The last example means that frequency generators 20 - 24 and the first six FM generators will be used. Allocation of individual notes to generators is then automatic and random, depending entirely on the current value of DE (density). The program does not look for free generators; notes can therefore be interrupted before they have sounded for their full duration.



Flow-chart showing music calculations for one sample, using primary, note-defining, and spacial parameters.

3.4.2 SA (SAMPLING) determines the rate in milliseconds at which music information is output to the studio; e.g. if SA = 10, new information is sent to the studio every ten milliseconds. However, if calculations take longer than the time defined by SA, then SA is ignored; in other words, SA defines only the minimum sampling rate.

The program maintains an internal sampling rate counter, independent of SA; when the necessary calculations for a sample have been made, this counter is adjusted to take into account the actual time it took to calculate the sample. The logic of this is illustrated in the following flow-chart.



ISRC=internal sampling rate counter. Contains the actual sampling time, displayed on the register for amplitude modulator 1.

SA can also be used, in conjunction with the MX and MS facilities of the "composition" program, to determine the sampling rate for quadraphonic moving-sound (cf §4.5.2).

Legal values for SA: 1 to 65536 milliseconds.

3.4.3 SE (SEGMENTS) and DU (DURATION) are structural parameters used only in the "composition" part of the program. SE defines the number of segments a compositional BLOCK is divided into, while DU defines the duration in milliseconds of each segment. For details of these parameters, see §4.1.1 and §4.1.5.

Legal values for SE: 1 to 50 (i.e. a maximum of 50 segments per block is allowed).

Legal values for DU: 10 to 131000 ms.

3.4.4 DS LF HF LA HA
These parameters are related to DE, F1, F2, A1, and A2 respectively, and are used in the "composition" program only. They define values for the primary parameters at the start of a compositional block. They are described in detail in §4.1.4.

3.4.5 SC (SCALE) defines the scale-patterns (interval relations within one octave) which govern the random choice of frequency between F1 and F2.

IMPAC contains a table of quarter-tones from 12 Hz to 19345 Hz (a little more than $10\frac{1}{2}$ octaves); a scale is defined by signalling which of the 24 $\frac{1}{4}$-tones within an octave are to be allowed; thus a whole-tone scale can be defined by taking every fourth $\frac{1}{4}$-tone: 1 5 9 13 17 21.

The program's procedure for finding a frequency for a new note is:
1) Calculate a random frequency between F1 and F2.
2) Look in the $\frac{1}{4}$-tone table to find the pitch which lies nearest below this random frequency.
3) See if this particular $\frac{1}{4}$-tone is contained in the current scale-pattern; if it is, then use it. If not, then go upwards through the table until a pitch is found which <u>is</u> allowed.

There are nine pre-defined scales, each of which can be accessed instantaneously in the real-time program (cf "pitch mode" §5.7.2). These scales are:

```
1:  1  5  9 11 15 19 23 ..............major
2:  1  5  7 11 15 17 23 ..............minor harmonic
3:  1  5  7 11 15 17 21 ..............minor melodic
4:  1  5  9 15 19 .....................pentatonic
5:  1  5  7 11 13 17 19 23 ...........alternating ½/whole-tones
6:  1  5  9 13 17 21 ..................whole tones
7:  1  4  7 10 13 16 19 22 ...........3/4-tones
8:  1  3  5  7  9 11 13 15 17 19 21 23..semitones (12-tone scale)
9:  1  2  3  4  5  6  7  8  9 10 11 12..quarter-tones
   13 14 15 16 17 18 19 20 21 22 23 24
```

During real-time performance scales can be transposed a given number of $\frac{1}{4}$-tone steps (cf §5.7.2). When they are not transposed, position 1 is the note G.

The user can replace any or all of the scales by giving the SC command from the program monitor. Command format:

```
>SC n
```

where n is the number (from 1 to 9) of the scale to be defined. The program answers

```
:
```

and waits for the user to define the $\frac{1}{4}$-tone positions which are included in the scale. This is done by writing not more than 24 numbers from 1 to 24 with one space or one carriage return between each number; after every carriage return the program writes

```
:
```

and waits for further numbers. The user indicates that input is complete by pressing [EOT] immediately after the program's ":".

```
>SC 2
:3 7 11 13 17 21 1
:[EOT]
>SC 3
:2 6 8 12 14 20
:22 24
:[EOT]
```

Numbers less than 1 or greater than 24 are ignored; i.e. there is no error message. However, if all the numbers are illegal or if no numbers are written, the program writes

```
TRY AGAIN
```

followed by

```
:
```

and waits for the user to redefine the scale.

```
>SC 6
:[EOT]
TRY AGAIN
:
```

3.5 Summary of the music parameters

PRIMARY	NOTE-DEFINING	SPACIAL	MISCELLANEOUS
frequency range F1 F2	note duration ND	moving-sound MS	frequency generators FG
amplitude range A1 A2	attack time AT	reverberation RV	sampling rate SA
density DE	glissando GL		segments and segment duration SE DU
	vibrato VD VS		start values LF HF LA HA DS
	timbre WF MI MF		scales SC

4 The Composition Program

In the composition program the user defines, tests and modifies compositional BLOCKS; these BLOCKS can be combined with one another both vertically and horizontally, while all the time the user has complete control: he can listen, interrupt, and alter at will.

4.1 Compositional BLOCKS

The main structural element in IMPAC is the compositional BLOCK, within which the primary parameters DE, F1, F2, A1 and A2 are defined as curves with up to fifty segments.

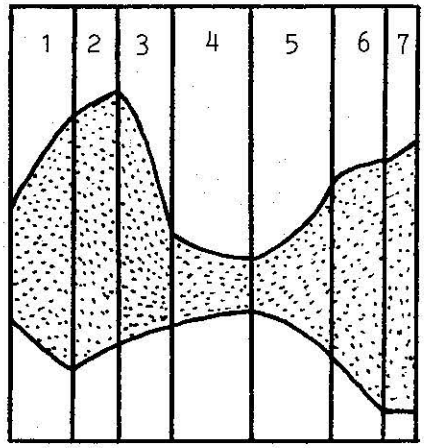
4.1.1

SE defines the number of segments in a BLOCK. Format:

>SE n

where n is a number from 1 to 50. Default value: 50

Within a segment a parameter curve can go in one direction only: up, down, or straight. To define the curves in the example, seven segments are used; the segment dividing lines mark the points at which at least one of the curves changes direction.



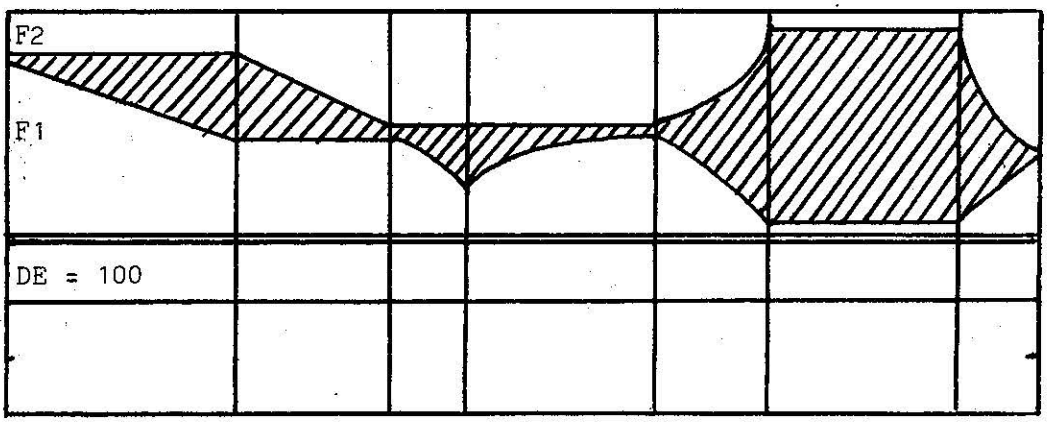
4.1.2

DE, F1, F2, A1, and A2 are "dynamic" parameters, in that they can be defined by curves. There are three ways of assigning values to them. (DE is here given as an example; the others are defined in exactly the same way.)

1) To assign one and the same value throughout a BLOCK, write:

>DE n

where n is a positive legal value (from 1 to legal maximum).



DE is here defined as

>DE 100

since it has the value 100 throughout the BLOCK.

2) To assign specific individual values to the curves, write:

```
>DE
```

or

```
>DE 0
```

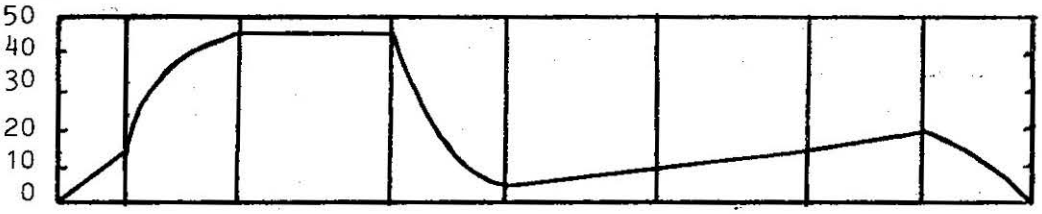
The program writes

```
:
```

and waits for the user to give two values for each curve segment. These values are:

a) the point reached at the end of the segment;
b) the shape of the curve during the segment. Shapes are defined as concave (-1 to -9), convex (1 to 9), or straight lines (0).

Suppose that the following density curve is to be described:



In segment one the curve moves up to 15 with shape 0 (a straight line); then it rises to 47 with shape 6, and stays there - i.e. it goes to 47 with shape 0. The curve is defined as follows:

```
(>SE 8)
>DE
:15 0 47 6 47 0 5 -5 10 0 15 0
:20 0 0 4
>
```

Between each number there is either one space or one carriage return; if carriage return is written before values for all segments (the current value of SE) have been defined, the program writes

```
:
```

and waits for further values.

N.B.
Two consecutive spaces, two consecutive carriage returns, and one space followed immediately by carriage return, must never be written within these curve definitions; they will cause values to be assigned to the wrong segments, and may occasionally give rise to error messages.

Input is terminated when either all values for the BLOCK have been written, or the key [EOT] is pressed. If [EOT] is pressed before all segments have been defined, then the values previously defined for the remaining segments are kept; if no values have been given previously, the default values are kept.

Default values:

F1	16	F2	3000
A1	280	A2	320
DE	0		

When input is terminated, the program checks the values. First it looks at the numbers defining points at the end of segments; if one is found to be illegal, a message is written together with the offending value. For example:

```
ILLEGAL VALUE
 123456
:
```

The user must then give one new value to replace this number; if the new number is illegal, or if EOT is pressed, the error message is repeated, until a legal value is given.

When all the point-defining numbers have been checked, the shape values are inspected one by one in the same manner.

Suppose that the density curve on the previous page is wrongly defined; conversation with the program might look like this:

```
>DE
:15 -10 47 6 4447 0
:-5 -5 10 0 15 0
:20 20 0 4
ILLEGAL VALUE
   4447                maximum legal density is 2000
:47                    type in the desired value
ILLEGAL VALUE
    -5                 minimum legal density is zero
: EOT                  EOT doesn't help here!
ILLEGAL VALUE
     -5
:                      an empty line is interpreted as
                       zero, which is legal
ILLEGAL VALUE
    -10                -9 is the lowest legal shape
                       value
:0
ILLEGAL VALUE
     20                9 is the highest legal shape
                       value
:                      empty line means zero
>                      now everything is legal
```

3) It is also possible to assign values for each segment with random and periodic functions. Format:

>DE n

where n is a negative whole number from -1 to -6.

-1	random	-2	sinus
-3	triangle	-4	sawtooth
-5	square	-6	curve

-1 RANDOM

a) The program asks:

RANDOM LIMITS:

and waits for two numbers, specifying the limits of variation. For example:

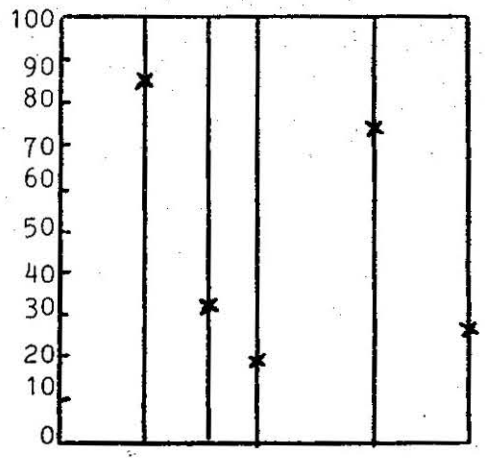
RANDOM LIMITS:10 100

Both numbers must be legal values for the parameter in question; if either or both of them are illegal the program writes:

ILLEGAL VALUE

and asks again:

RANDOM LIMITS:



random values 10-100

EOT returns control to the program monitor without altering the existing curve definitions.

b) The program then asks:

SHAPE LIMITS:

and waits for two more numbers, defining the limits of variation for the curve shapes (-9 to 9). Shape values are then chosen at random between these two numbers. For example:

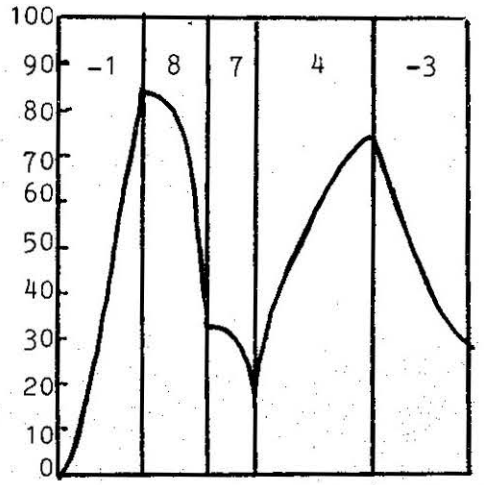
SHAPE LIMITS:-9 9

allows all possible curve shapes. If either number is illegal, the program writes:

ILLEGAL VALUE

and asks again:

SHAPE LIMITS:



as above plus random shapes -9 to 9

EOT returns control to the program monitor without altering the existing shape values.

To define the example in (b), write:

(>SE 5)

>DE -1

RANDOM LIMITS:10 100

SHAPE LIMITS:-9 9

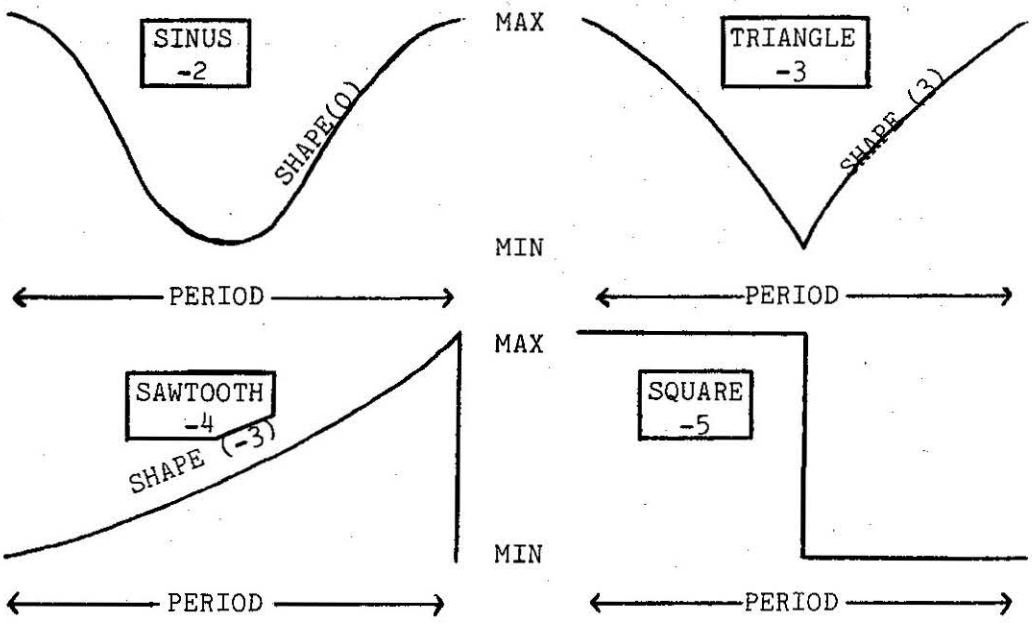
-2 to -5 SINUS TRIANGLE SAWTOOTH SQUARE

If DE is followed by one of the numbers -2, -3, -4, or -5, the program asks:

MIN,MAX,PERIOD,SHAPE:

and waits for four numbers to be written, defining the structure of a periodic oscillation.

MIN and MAX define the lower and upper limits of the oscillation.
PERIOD defines the period length in milliseconds.
SHAPE defines the curve form deviation (-9 to 9); this is not valid for -5 (square).



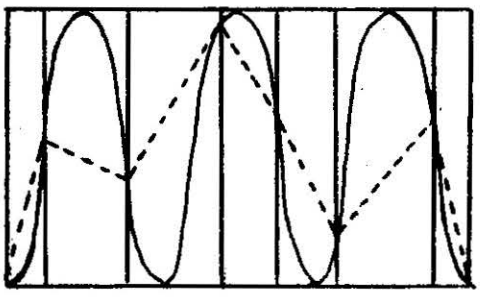
MIN and MAX are checked for illegal values; if either of them is outside the parameter limits, the message:

ILLEGAL VALUE

is written, and the program asks again:

MIN,MAX,PERIOD,SHAPE:

PERIOD should be in the range 10 to 131000 milliseconds, and SHAPE should be in the range -9 to 9. If either of these values lies outside its prescribed range, it will be difficult to predict the structure of the resulting curve.



When these values are defined, the program creates an imaginary oscillation, from which it takes values for the end of each segment. Note that the parameter does not in fact get a smooth oscillation: instead it gets straight lines between the points where the segments and oscillation intersect. Note also that the durations of each segment must be defined first (see §4.1.5 below). If they are defined afterwards, the values calculated for oscillations are not adjusted, which means that the oscillations become distorted.

EOT returns control to the program monitor.

-6 CURVE

The program asks:

MIN,MAX,PERIOD,SHAPE:

and waits for four numbers.

MIN is the starting point.
MAX is the end point.
PERIOD is the duration in milliseconds of the curve (10 to 131000).
SHAPE is the curve-form deviation (-9 to 9).

MIN and MAX must be legal values for the parameter being defined. If either of them is illegal, the program writes:

ILLEGAL VALUE

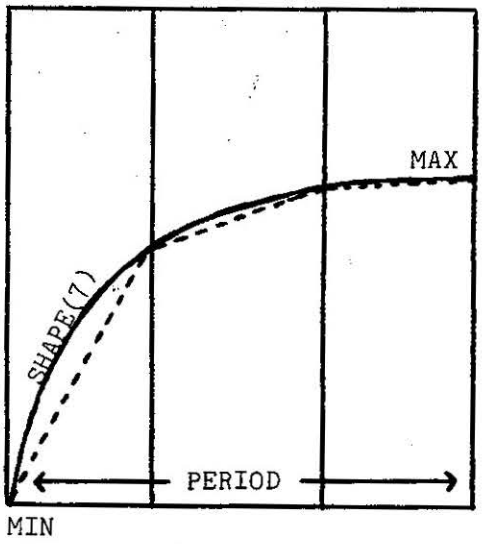
and asks again:

MIN,MAX,PERIOD,SHAPE:

EOT returns control to the program monitor without altering the existing curve definition.

If PERIOD is shorter than the total duration of the BLOCK, the value MAX is kept from the point where it is first reached until the end of the BLOCK.

If PERIOD is longer than the BLOCK duration, MAX is never reached.



As with the periodic functions, the program creates an imaginary curve from the data given (including segment durations); values for each segment are then calculated at the curve intersections.

4.1.3 Summary of the definition of dynamic-parameter curves: DE, F1, F2, A1, and A2.

>DE n	>DE	>DE -n
same value throughout the BLOCK	individual values and shapes for each segment	automatic functions: -1: random -2: sinus -3: triangle -4: sawtooth -5: square -6: curve

4.1.4 The starting values for the primary parameters at the beginning of a BLOCK are defined with:

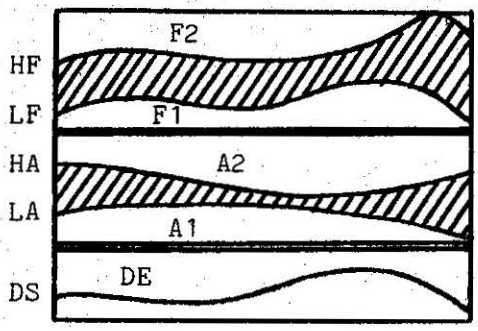
>DS n (DENSITY START) for DE curve
>LF n (LOW FREQUENCY) for F1 curve
>HF n (HIGH FREQUENCY) for F2 curve
>LA n (LOW AMPLITUDE) for A1 curve
>HA n (HIGH AMPLITUDE) for A2 curve

In all cases n must be a legal value for the parameter concerned.

Note that if a primary parameter is defined by the first method described above (one value throughout the BLOCK), the equivalent start-parameter is automatically given the same value. Thus:

>F1 220

automatically assigns 220 to LF as well.



5 DU (DURATION) defines the length in milliseconds of each segment. As with the dynamic parameters, there are three methods of giving values to DU:

>DU n	>DU	>DU -n
same value throughout the BLOCK; e.g. >DU 1000 if all segments are to be 1000 milliseconds long.	individual values for every segment; e.g.: (>SE 5) >DU :1000 500 5200 :750 2500	automatic functions: -1 random -2 sinus -3 triangle -4 sawtooth -5 square -6 curve questions and answers as for dynamic parameters.

N.B.

1) Values assigned to DU are always rounded down to the nearest multiple of ten. Thus 1317 becomes 1310, etc.

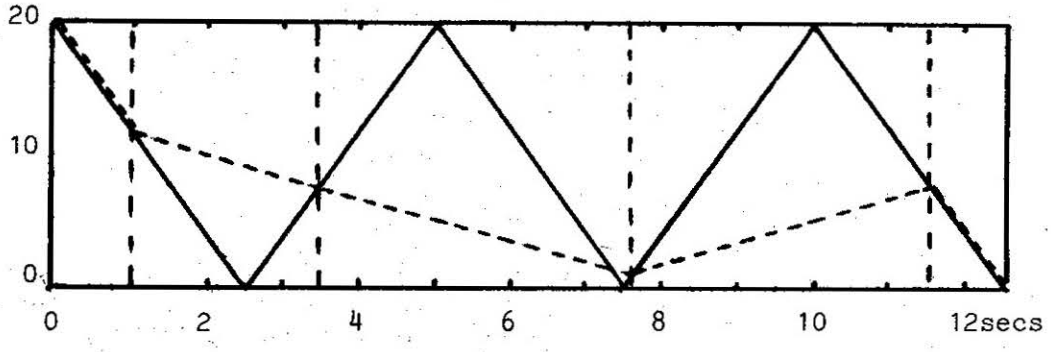
2) The current values of DU are always inspected when automatic functions are calculated for the dynamic parameters; it is therefore essential that DU be defined before such functions are used. Suppose, for example, that density is defined as follows:

```
>DE -3
MIN,MAX,PERIOD,SHAPE:0 20 5000 0
>DS 20
>
```

The shape of the resultant curve will depend on the current values of DU.

——— >DU 2500

- - - - - - >DU
:1000 2500 4100
:3900 1000



4.1.6 FG, ND, GL, and WF are "static" parameters; they can be controlled by single values only, not by curves.

1) The normal way of defining static parameters is to write the mnemonic followed by a number, indicating that there is to be a constant value throughout the BLOCK. Thus:

>ND 1000	note duration is 1000 milliseconds
>GL 2000	glissando goes up one octave
>WF 2	wave-form 2 (parabolic) on all generators
>FG 6 13	use generators 6 to 13 inclusive

Note that two numbers are required for FG; if only one is given, only one generator will be used. Thus:

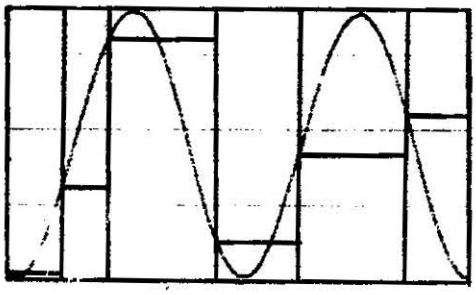
>FG 8

gives exactly the same result as

>FG 8 8

If no number is written after the mnemonic, zero is assumed.

2) It is also possible to assign automatic functions to the static parameters. Definition and question / answer formats are the same as those for the dynamic parameters, with the exception that function -1 (random) has no question about "SHAPE LIMITS".



-1	random	-2	sinus
-3	triangle	-4	sawtooth
-5	square	-6	curve

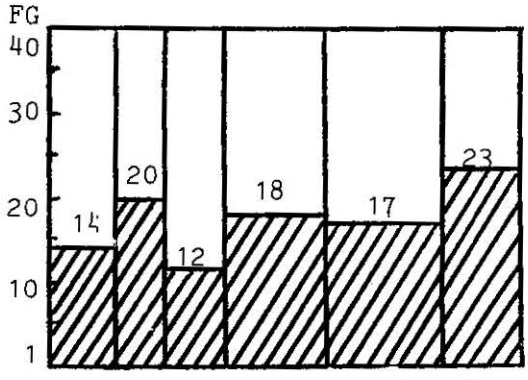
One value is calculated for each segment, and kept throughout the segment.

Note that when FG is defined by an automatic function, the user gives values denoting the total number of generators to be used; thus the sequence

>FG -1

RANDOM LIMITS:12 24

>



means that a minimum of 12 and a maximum of 24 generators will be in operation; the program chooses generators starting at number one.

N.B.

Automatic functions on static parameters are calculated when the music is played, not at the time of definition; it is therefore not necessary to define DU before these functions are used.

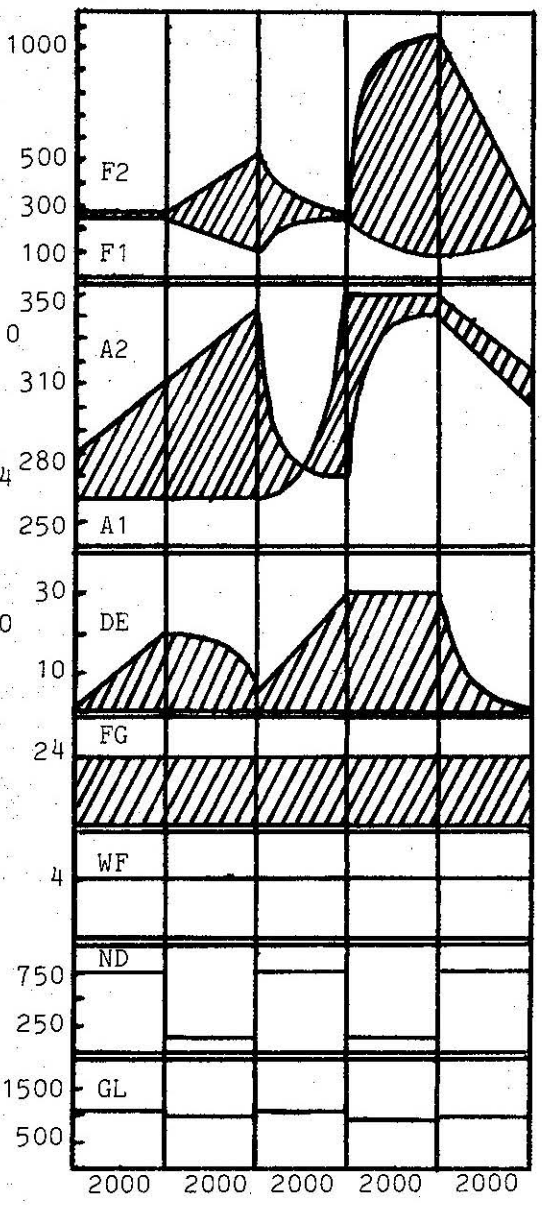
4.1.7 Summary of the composition-program parameters:

Structural:	SE	DU			
Dynamic:	DE	F1	F2	A1	A2
Curve-start:	DS	LF	HF	LA	HA
Static:	FG	ND	GL	WF	

These parameters constitute the BLOCK definition, and values assigned to them are associated absolutely with a particular BLOCK; so they remain unchanged every time a BLOCK is played.

Here is an example of a graphic score for a compositional BLOCK, together with the parameter definitions needed to create the BLOCK.

```
>SE 5
>DU 2000
>FG 1 24
>WF 4
>GL -1
RANDOM LIMITS:990 1012
>ND -5
MIN,MAX,PERIOD,SHAPE:50 750 4000 0
>LF 247
>F1
:247 0 123 0 247 5 110 -4 220 -4
>HF 262
>F2
:262 0 523 0 262 -4 1046 7 262 0
>LA 260
>A1
:260 0 260 0 350 -5
:350 0 315 0
>HA 280
>A2
:310 0 340 0 270 -7 340 6 300 0
>DS 1
>DE
:20 0 5 7 30 0 30 0 1 -5
>
```



4.1.8 SA (SAMPLING) is not associated with a BLOCK definition, but can be freely changed without affecting BLOCK structures. Format:

```
>SA n
```

where n is a whole number from 1 to 65536 defining the sampling-step time (cf §3.4.2).

Default value: 5 milliseconds

4.1.9 Values assigned to the BLOCK-defining parameters are stored in the BLOCK TABLE; this table can be listed with the command:

```
>L
```

The BLOCK definition above, for example, would be listed as follows:

```
SE     5
FG   124
SA     5
WF     4
GL    -1
ND    -5
LF   247
HF   262
LA   260
HA   280
DS     1
F1
    247      0    123      0    247      5    110     -4    220     -4
F2
    262      0    523      0    262     -4   1046      7    262      0
A1
    260      0    260      0    350     -5    350      0    315      0
A2
    310      0    340      0    270     -7    340      6    300      0
DE
     20      0      5      7     30      0     30      0      1     -5
DU
   2000   2000   2000   2000   2000
```

Note

1) FG 124

Values for "FG" are defined by the user as

>FG n1 n2

They are stored in the BLOCK table as one value: (n1 x 100) + n2; "FG 124" in the table means, therefore, "FG 1 24".

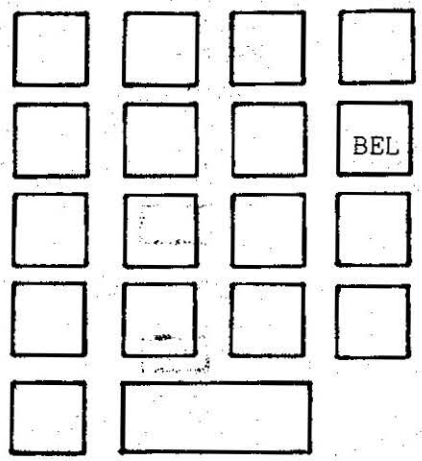
2) SA 5

SA is listed in the BLOCK table even though it is not part of a BLOCK definition.

3) GL -1 ND -5

When static parameters are controlled by automatic functions, only the function number is stored in the table; the values for each segment are calculated when the music is played. However, when dynamic parameters are assigned automatic functions, the segment values are calculated immediately and stored in the BLOCK table; so they can be checked at once with "L".

4) The BLOCK table has room for values for 50 segments; the "L" command lists as many segments as are specified by "SE". Thus if SE = 30, values for 30 segments are listed, and so on. If a BLOCK contains a large number of segments, the values will not all fit on to the Tektronix screen. When the screen is full, the program waits for the user to press the function button [BEL]; this clears the screen, and allows listing to continue.



4.1.10 The BLOCK table can also be inspected parameter by parameter:

```
>L n
```

where n is a parameter number according to the following list:

1	SE	7	LF	12	F1
2	FG	8	HF	13	F2
3	SA	9	LA	14	A1
4	WF	10	HA	15	A2
5	GL	11	DS	16	DE
6	ND			17	DU

For example:

```
>L 16
DE
   20     0     5     7    30     0    30 ... 0     1    -5
```

4.1.11 The normal way to change values in the BLOCK table is to redefine the parameters. Thus:

```
>WF 2                    defines wave-form
>WF 4                    gives wave-form a new value
```

The dynamic parameters are redefined in much the same way:

```
>DE -1                   define density curve (random)
RANDOM LIMITS:0 25
SHAPE LIMITS:-2 3
>DE -2                   redefine curve (sinus)
MIN,MAX,PERIOD,SHAPE:0 25 15000 -2
>etc.
```

It is also possible to alter values for individual segments with the CS (CHANGE SEGMENT) command. Format:

```
>CS n
```

where n is the number of the segment to be altered. The program then writes out the values and shapes of the dynamic parameters for this segment, waiting after each one for the user to accept or alter the values.

- To keep old values, press [EOT].
- To assign new values, type them after the program's ":".

For example:

```
>CS 5                          change segment 5
F1    220     -4               current values
: EOT                          keep them
F2    262      0
: EOT                          keep these, too
A1    315      0
:280 -4                        make this quieter, with
                               faster decay
A2    300      0
:250                           this one quieter, too
:0                             I forgot to give a value
                               for shape!
DE      1     -5
: EOT                          leave this as it is
DU   2000
:
                               carriage return immedi-
ILLEGAL VALUE                  ately after ":" is in-
                               terpreted as zero, il-
:2000                          legal for DU
>
```

4.1.12 There are several other parameters which cannot be controlled from the composition program.

Parameter	Default value	Parameter	Default value
MI	100	MF	1000
VS	100	VD	5
MS	500	RV	240
AT	20		

1) Single values can be set on these parameters with the command:

```
>SP n1 n2                      ( = SET PARAMETER)
```

where n1 is the number of the parameter to be set (see below);
n2 is the value to be assigned.

1	ND	2	GL	3	MI	4	MF
5	F1	6	F2	7	A1	8	A2
9	DE	10	AT	11	MS	12	RV
13	-	14	VS	15	VD	16	-

For example:

```
>SP 10 40
```

puts AT (ATTACK TIME) to 40 milliseconds.

2) The current values of these parameters can be obtained with:

>SP n

where n is the number of the parameter. The program prints the value, followed by ":", and waits for the user to give a new value. If the old value is to be kept, press EOT. For example:

```
>SP 11                          inspect MS

MS  500                         program prints current
                                value
:EOT                            keep it

>SP 14                          inspect VS

VS    0                         program prints current
                                value
:175                            type in a new value

>
```

WARNING

It will be noticed that SP can put values on the dynamic parameters DE, F1, F2, A1, and A2, and on two of the static parameters, ND and GL. The user is advised to use this facility sparingly: SP has the effect of "freezing" a parameter, so that it is no longer controlled by curves or other values assigned in a BLOCK definition. The procedure for "unfreezing" a parameter is not complicated, but it is easy to forget that it must be done. **This function belongs properly to the real-time program, and is discussed more fully in §5.2.5. The beginner should therefore use SP for MI, MF, AT, MS, RV, VS, and VD only.**

4.2 Playing a BLOCK

When a compositional BLOCK has been defined, it can be played in the studio with one of the commands PL or PP.

4.2.1

To play a BLOCK from its beginning, write:

>PL[n] (= PLAY)

where n is a positive whole number.

N.B.
The parenthesis [n] indicates here and later that this is an optional part of the command; the parentheses must not be written by the user.

1) >PL

When this command is given, a check is made first to see if any values in the BLOCK table have been changed since the last time PL or PP was done.

- If changes have been made, then this is assumed to be a new BLOCK definition, and accordingly the BLOCK table is saved. A file is created on DK <USE> containing this BLOCK's information; it automatically receives a number, such that the first file created during a run is called "1 BIN", the second is "2 BIN", and so on. The program then reads from the newly created file to play the music in the studio. The number of the most recently created BLOCK file is displayed at all times on the studio register for the Noise Generator.

- If no changes have been made, the most recently created BLOCK file is played; and if no file has been created, values in the default file "0 BIN" are read in and played.

2) >PL n

This command plays BLOCK-file number "n".
- If n is negative or zero, it is interpreted as being undefined, and the command is carried out as in (1) above.
- If n is positive, but there is no file on DK <USE> with this number, the program prints:

FILE n BIN NOT FOUND

and control reverts to the program monitor.

4.2.2 To play part of a BLOCK, write:

>PP[n] (= PLAY PART)

where n is a positive whole number indicating a BLOCK-file number.

- If n is undefined, zero or negative, the most recently created BLOCK-file is read from.

The program asks:

SEGMENTS:

and waits for the user to give two values stating which segments in the named BLOCK file are to be played. For example:

SEGMENTS:10 20

indicates that segments 10 to 20 inclusive are to be played.

- If the file contains fewer than 10 segments, the program writes:

SEG 10 NOT FOUND

and control is returned to the program monitor.

- If the second number is greater than the total number of segments in the BLOCK, the BLOCK is played to the end; there is no error message.

4.2.3 Playing is completed either when the last segment in the BLOCK has been played, or when the SPACE key is pressed. In the latter case, the program prints:

STOPPED IN SEG n

where n is the number of the segment being played at the time of the interruption; and control reverts to the program monitor.

4.2.4 BLOCK files can also be created (but not played) with the commands F and PF.

1) >F (= FILE)

creates a file from the current values in the BLOCK table; this file is numbered in the normal way (see §4.2.1).

2) >PF[n] (= PART FILE)

creates a new BLOCK file from a section of an already existing one. The program asks:

SEGMENTS:

and waits for the user to give two numbers stating which segments in the named file are to be included in the new file.

For example:

```
>PF 5                          create a new file from
                               part of BLOCK file no.5
SEGMENTS:4 8                   it will consist of seg-
                               ments 4 to 8 inclusive
17     BIN OK                  program answers that a
                               new file has been cre-
>                              ated called "17 BIN".
```

Files created with "PF" are numbered automatically in the same way as all BLOCK files.
- If n is undefined, negative or zero, the most recently created BLOCK-file is read from.

4.2.5 Suppose that the BLOCK definitions in §4.1.7 have been made at the beginning of a run. This BLOCK might then be tested and adjusted in the following way.

```
>CN 2                          make studio connections
                               first
>PL                            play the BLOCK - a file
                               is created: "1 BIN"
>WF 2                          see what it sounds like
                               with wave-form 2
>PL                            "2 BIN" is created and
                               played
STOPPED IN SEG      3          playing is interrupted
                               with SPACE key
>WF 3                          try it with wave-form 3
>GL 1000                       and without glissando
>PL                            "3 BIN" is created and
                               played
>ND 10000                      try it with very long
                               notes
>DE                            and with density half
                               of what it was before
:10 0 2 7 15 0 15 0 1 -5
>PL                            "4 BIN" is created and
                               played
>SP 15                         have a look at vibrato
                               depth (VD)
VD     5                       it's 5%
:12                            make it 12%
>SP 14 220                     and make vibrato speed
                               (VS) a bit faster
>PL                            no change to BLOCK table
                               so "4 BIN" again
>PL 1                          play "1 BIN" - with new
                               vibrato
>PF 1                          make a file of segments
                               4 & 5 from BLOCK no. 1
SEGMENTS:4 5
5      BIN OK                  this file gets name "5
                               BIN"
>PL                            new file no.5 is played
```

4.2.6 Normally BLOCK files are numbered automatically in the order in which they are created. The numbering can, however, be controlled with:

```
>FP 0 n                    ( = FILE POINTER)
```

where n is a positive whole number defining the number which is to be given to the next BLOCK file created.

Suppose for example that ten BLOCKS, numbered 1 to 10, are created during a run; in the next run it is desired to keep these ten files on the disk, and to start numbering new BLOCKS from 11 onwards. Write:

```
>FP 0 11
```

"FP" is discussed more fully in §6.3.5.

4.2.7 BLOCK files can be renamed with the command:

```
>RN n1 n2                  ( = RENAME)
```

where n1 is the number of an existing BLOCK file;
n2 is a positive whole number, stating the new number that the file is to receive.

1) If file "n1 BIN" is not found on DK <USE>, the program writes:

```
n1 BIN NOT FOUND
```

and control reverts to the program monitor.

2) If n2 is undefined, negative or zero, the program writes:

```
ILLEGAL VALUE
```

and control reverts to the program monitor.

3) If there is already a file on DK <USE> called "n2 BIN", the program writes:

```
FILE ALREADY ON DK. ARE YOU SURE?Y/N
```

The user can answer "Y" or "N".
Y: The existing file "n2 BIN" is deleted, and "n1" becomes "n2".
N (or any character other than Y): Control is returned to the program monitor without the re-numbering taking place.

When the operation is complete, the program writes:

```
n2    BIN OK
```

4.2.8 BLOCK files can be inspected with the command:

```
>WR[ n]                    ( = WRITE)
```

where n is the number of the BLOCK file to be listed; if n is undefined, zero or negative, the most recently created BLOCK file is listed. Listing with "WR" is performed on .DAT slot 20 (usually assigned to TW1); if there is too much information in the file to fit on to the screen, the BEL key must be pressed to clear the screen and allow listing to continue.

The format of these files is discussed in detail in §6.9.1.

4.2.9 Summary of BLOCK-file manipulations

Play	PL (PLAY)	PP (PLAY PART)
Create	F (FILE)	PF (PART-FILE)
Name	RN (RENAME)	FP (FILE-POINTER)
List	WR (WRITE)	

4.3 EDIT

IMPAC has an internal EDIT facility which can be used to alter parameter values in previously created BLOCK files.

4.3.1 EDIT is a subprogram with its own submonitor and set of commands. To enter EDIT, write:

```
>ED[ n]
```

where n specifies the BLOCK file which is to be edited; if n is undefined, negative or zero, the most recently created BLOCK file is opened for editing.

4.3.2 When the submonitor is ready to receive a command, it prints:

```
:
```

Commands are of two types:
1) parameters: DU DE F1 F2 A1 A2 FG WF GL ND
2) file-manipulation: D G AB EOT SE

4.3.3 The parameters are changed as follows

1) Multiplication

```
:DU n          (0 ≤ n)
:DE n          (0 ≤ n)
:F1 n          (0 ≤ n ≤ 8192)
:F2 n          (0 ≤ n ≤ 8192)
```

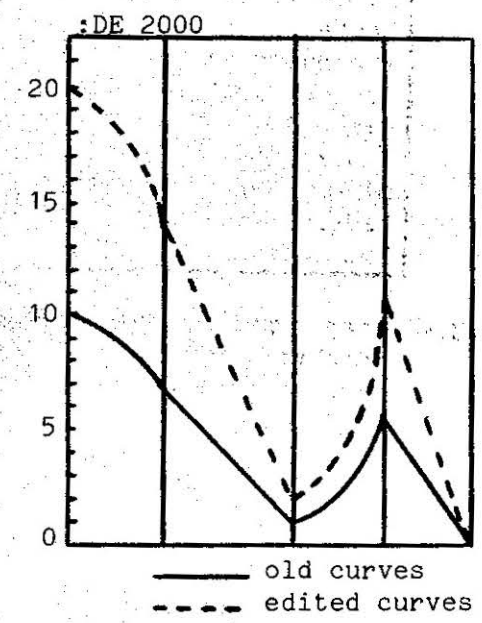
Here n represents a multiplication factor, such that the new value equals the old value times n / 1000.
For example:

```
:DE 2000
```

indicates that density values are to be multiplied by 2000 / 1000 , which means that they are to be doubled. Similarly:

```
:F1 1059
```

indicates that values for F1 are to be multiplied by 1.059, which has the effect of raising them by a semitone. See also the table on the next page.



2) Addition

```
:A1 n
:A2 n
```

Here n is to be added to the old values in the file. Thus:

```
:A1 16
```

adds 16 to values for A1, i.e. they are increased by 4 dB. Similarly:

```
:A2 -10
```

has the effect of lowering A2 values by $2\frac{1}{2}$ dB.

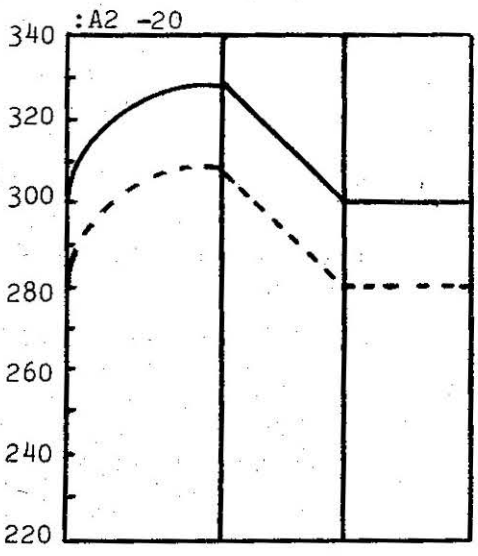


Table of multiplication factors for raising and lowering pitches (unit = 1000)		
Interval	Raise	Lower
Unison	1000	1000
Minor second	1059	944
Major second	1122	891
Minor third	1189	841
Major third	1260	794
Perfect fourth	1335	749
Aug. fourth	1414	707
Perfect fifth	1498	667
Minor sixth	1587	630
Major sixth	1682	595
Minor seventh	1782	561
Major seventh	1888	530
Octave	2000	500
Minor 9th	2119	472
Major 9th	2245	445
Minor 10th	2378	420
Major 10th	2520	397
Perfect 11th	2670	375
Augmented 11th	2828	354
Perfect 12th	2997	334
Minor 13th	3175	315
Major 13th	3363	297
Minor 14th	3563	281
Major 14th	3775	265
Two octaves	4000	250

3) New values

```
:FG n1 n2 (1 ≤ n1 ≤ n2 ≤ 40)
:WF n     (0 ≤ n ≤ 7)
:GL n     (0 ≤ n ≤ 8192)
:ND n     (0 ≤ n ≤ 131000)
```

The values given with these parameters are new values which are to replace the old ones. For example:

```
:WF 4
```

means that wave-form no.4 is used instead of the old value.

N.B.

Definitions given with multiplication and addition factors may result in values outside the legal limits being calculated. If this happens, the program automatically puts the parameter to the legal minimum or maximum.

For example, after the command:

```
:F1 2000
```

(raising all values by an octave) frequencies of 8000 Hz or more are edited to 15999 Hz exactly.

4.3.4 When one of the parameter-commands above is given, its value is checked and then stored; the actual editing takes place when an extra carriage return is done immediately after the program's ":". For example:

```
>ED 2                          edit BLOCK file 2

:DU 500                        halve segment durations

:GL 2000                       put GL to 2000

:                              carriage return to com-
                               plete editing
2     BIN OK                   editing done & all well

>
```

In this case all the segments are changed. Note however that no changes are made until the extra carriage return is done; so the sequence:

```
:DU 500

:DU 800
```

is quite legitimate: 800 replaces 500 as multiplication factor.

4.3.5 It is also possible to edit individual segments, by writing:

```
:SE n                    ( = SEGMENT)
```

where n is a positive whole number denoting which segment is to be altered. The "SE" command can be given at any time before the extra carriage return is done; thus:

```
>ED 2
:SE 5                    alter segment 5 only
:DU 200
:A1 8
:                        carriage return to per-
                         form editing
:                        program waits for more
                         commands
```

In this example the first four segments remain unchanged; segment 5 is altered (:DU 200 and :A1 8) and the program waits for more instructions. In other words "SE" followed at some stage by an extra carriage return moves a "pointer" to the named segment, performs editing there, and then leaves the pointer immediately after the named segment.

EDIT always counts segments from the beginning of the file; so the number given in any further "SE" command must be higher than those given previously. Thus:

```
>ED 2
:SE 5                    in segment 5 ...
:DE 4000                 multiply density by 4
:                        carriage return to per-
                         form this
:SE 1                    pointer is between seg-
                         ments 5 and 6: it can't
ILLEGAL VALUE            go backwards
:SE 6
:ND 50                   50 ms on ND for seg 6
:                        do it
:DE 500                  density x 0.5 for rest
                         of file (seg 7 onwards)
:                        do it
2    BIN OK
```

If the file does not contain as many segments as are defined after "SE", the file is closed and a message is output:

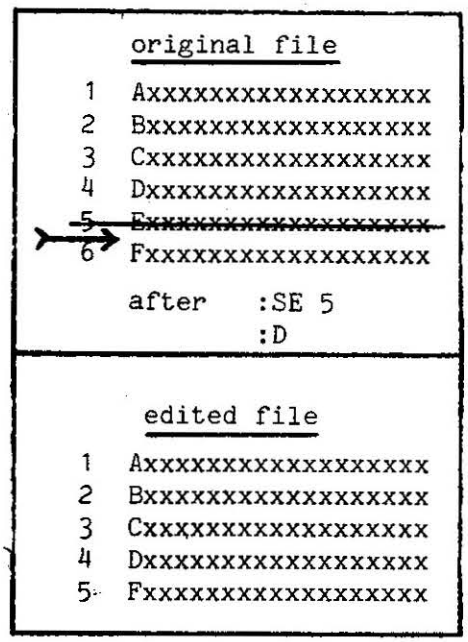
```
SEG   n NOT FOUND
```

4.3.6 Single segments can be deleted with:

```
:D                    ( = DELETE)
```

SE must be defined before "D" can function. For example:

```
>ED 10

:SE 5

:D

:SE 10

:D

:

10    BIN OK
```



If SE is not defined, the program prints an error message, and the submonitor waits for further commands:

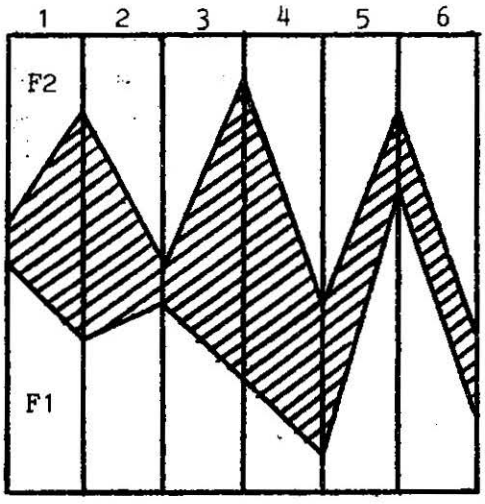
```
'D' ONLY AFTER 'SE'
:
```

After "D" the pointer is positioned between the deleted segment and the next one; note that following segments retain their original numbering until the file is closed.

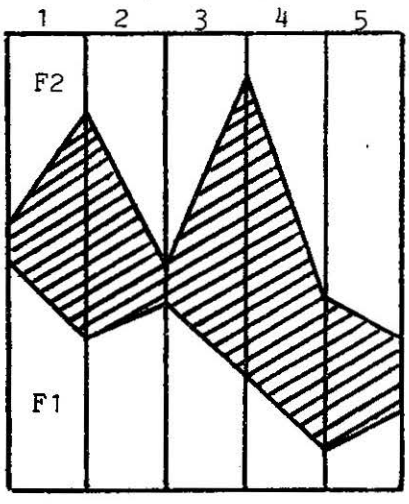
N.B.

For the dynamic parameters, a segment contains information about the final value reached by the curve and the shape of the curve; so when "D" is used the effect is usually a compromise between the deleted segment and the one following it. In the following figure, for example, the diagram on the left represents the frequency curve in an unedited BLOCK-file. The diagram on the right shows what the curve looks like after the EDIT commands:

```
:SE 5
:D
```



original file



edited file

4.3.7 BLOCK files can be combined with:

:G n (= GET)

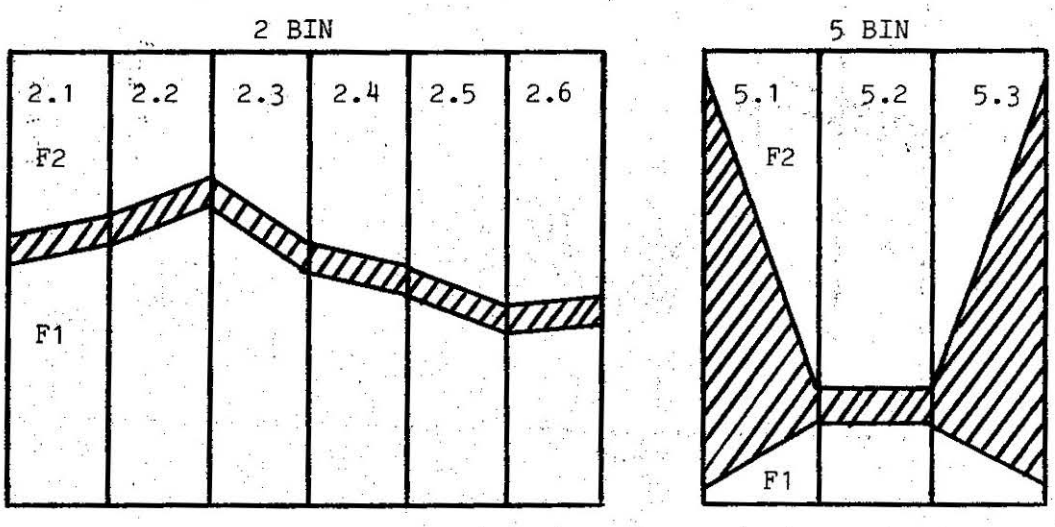
where n is the number of a BLOCK file which is to be combined with the current file. Thus:

```
>ED 2                    open BLOCK file "2 BIN"
                         for editing
:G 5                     read in "5 BIN"
:DU 2000                 the pointer is now after
                         last seg of "5" & before
:GL 8192                 first seg of "2"; chan-
                         ges affect "2 BIN" only
:                        perform editing
2     BIN OK
>
```

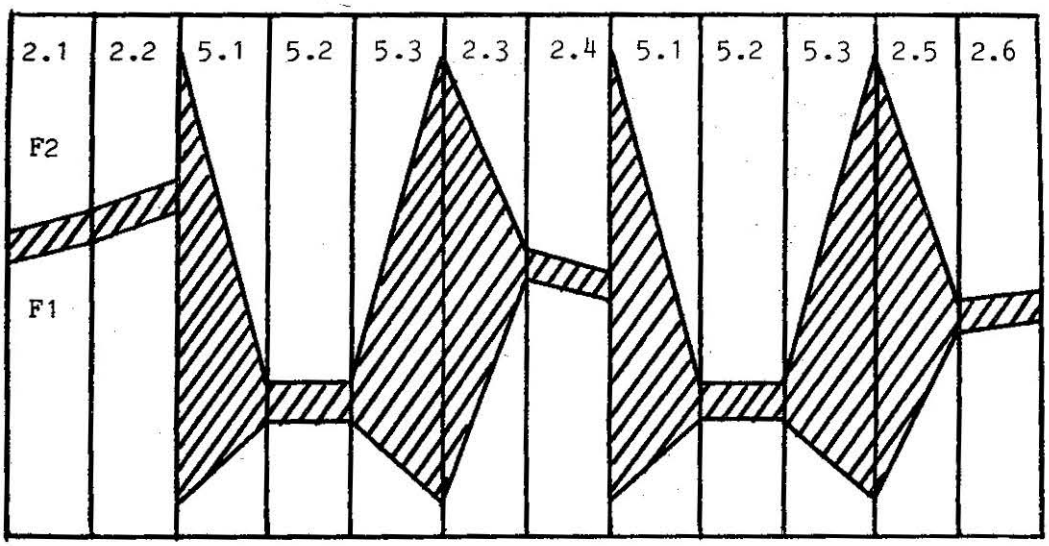
The file read in with "G" is copied without alteration into the place where the pointer is; it is thus possible to place an outside BLOCK anywhere within the current BLOCK. For example:

```
>ED 2                    open "2 BIN" for editing
:SE 2
:                        move the pointer after
                         seg 2
:G 5                     read in "5 BIN"
:SE 4
:                        move pointer after seg-
                         ment 4
:G 5                     read in "5 BIN" again
:etc
```

Suppose that the frequency curves in the original BLOCKS "2 BIN" and "5 BIN" looked like this:



In the combined file the frequency curves will be as follows:



It will be noticed that segments 2.3 and 2.5, which come immediately after the inserted BLOCK, are adjusted automatically to achieve a smooth transition; however, the first segment of the inserted BLOCK, 5.1, is <u>not</u> normally adjusted.

A smooth transition can be obtained here with

```
:G n1 n2
```

where n1 is the number of the BLOCK;
n2 is any non-zero whole number.

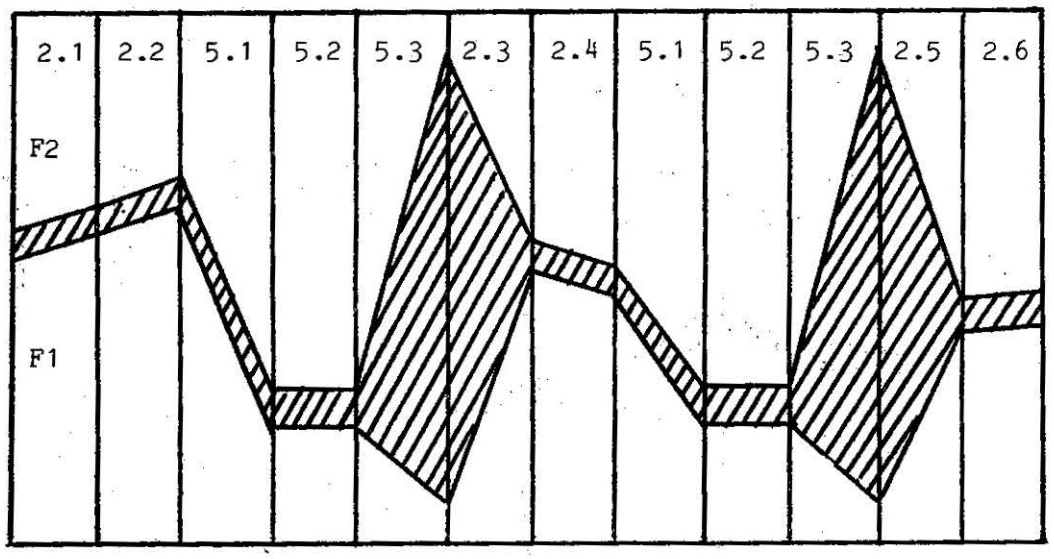
Written in this format, the command has the effect of ignoring the curve start-values at the beginning of segment 1. For example:

```
:G 5 1
```

instead of

```
:G 5
```

in the sequence of instructions on the previous page would give the following result:



4.3.8 If at any time the user should regret the alterations made with EDIT, one of the commands:

```
:AB                    ( = ABORT)
```

or

```
:[EOT]
```

will restore the BLOCK to its pre-edit form. This can be done, of course, only if the file is still open for editing. For example:

```
>ED 2                          open BLOCK "2" for editing
:SE 1
:                              point after segment 1
:G 10                          get BLOCK "10 BIN"
:AB                            it should have been "9"
                               so abort from this edi-
ORIGINAL FILE KEPT             iting and ...
>ED 2                          start again
:etc
```

4.3.9 Summary of the ED-submonitor commands

PARAMETERS			OTHER
multiplication	addition	new values	OPERATIONS
DU	A1	FG	SE
DE	A2	WF	D
F1		GL	G
F2		ND	AB
			[EOT]

4.3.10 A word of warning!

It is vital that the user distinguish between the following commands:

```
(1)  >GL 2000          (2)  >ED
     >etc.                  :GL 2000
                            :etc.
```

(1) Here a change is made to one of the parameters in the BLOCK table. The next time "PL" is done a new BLOCK file will be created, containing this parameter value.

(2) Here the most recently created BLOCK file is altered, but the BLOCK table remains unchanged. So the next "PL" will not cause a new file to be created; it will instead play the old one in its edited version.

This means that the next time a new BLOCK file is created it will still have the old value for GL, not the one given in the EDIT command.

4.4 Horizontal / sequential mixing

KP (KEEP) is used to mix BLOCKS sequentially.

4.4.1 Format:

```
>KP  n1  n2
```

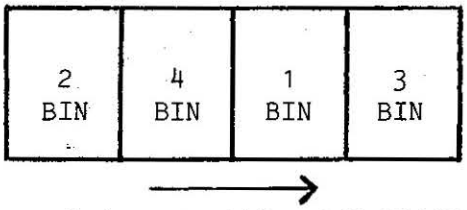
where n1 is the number of the BLOCK which is to be "kept" (mixed); if n1 is zero, negative or undefined, the most recently created BLOCK file is kept;

n2 is an optional "transition-switch"; if it is non-zero there will be a smooth transition into this BLOCK (in other words the start-values of this BLOCK will be ignored - cf §4.3.7 :G n1 n2); if it is zero or undefined, this BLOCK will be mixed without alteration.

KP copies the contents of "n1 BIN" to a special-purpose BLOCK called "999 BIN". Successive KEEP commands copy the named files to "999 BIN" in the order in which the commands are given. For example, after these commands:

```
>KP 2
>KP 4
>KP 1
>KP 3
```

file 999 BIN looks like this:



The KEEP-file is open (new BLOCKs can be copied on to it) until "999" is referenced with one of the commands:

```
>PL 999      >PP 999
>PF 999      >WR 999
>ED 999      >TM 999
>MX 999
```

```
>KP 999
is ILLEGAL!
```

These commands go through the following operations:

1) Close "999 BIN" and rename it, in such a way that the first KEEP-file becomes "1000 BIN", the second becomes "1001 BIN", and so on; this leaves "999" free to be used with any further KEEP files.
2) Perform the desired task on the renamed file.
3) Output message to show what number this KEEP-file has received.

For example:

```
>KP                          KEEP the latest BLOCK
>FG -1                       redefine generators
RANDOM LIMITS:1 40
>PL                          create & play a new BLOCK
>KP                          KEEP this new one
>PL 2                        play BLOCK "2"
>KP 2                        and KEEP it
>PL 999                      play the KEEP file
1000  BIN OK                 it gets name "1000 BIN"
>
```

Note that files created with "KP" are identical in format to all BLOCK files; they can therefore be manipulated in exactly the same ways. Thus:

```
>PL 1000
>ED 1000
>KP 1000
>etc
```

are all legal commands.

4.4.2 The numbering of KEEP-files can be directed with:

```
>FP 1 n                ( = FILE-POINTER)
```

where n is a positive whole number defining the name which is to be given to the next KEEP-file created. See also §6.3.5.

4.5 Vertical / parallel mixing

BLOCK files can be mixed vertically with MX (MIX) and other related commands.

4.5.1 The user creates a MIX-FILE by giving the numbers and start-times of the compositional BLOCKS that are to be in it. Format:

```
>MX[n]
```

where n is a whole number specifying the BLOCK that is to be mixed. If n is undefined, negative or zero, the most recently created BLOCK is mixed.

The program asks:

```
START TIME:
```

and waits for the user to write a floating-point number defining the time in seconds at which this BLOCK is to start, relative to the beginning of the MIX-FILE. Write, for example:

```
START TIME:1.5
```

if this BLOCK is to start 1½ seconds after the beginning of the MIX-FILE. Note that if no decimal point is written, the whole number is interpreted as seconds. Legal formats:

```
0.5  }
 .5  }   ½ second
.50  }
1.0  }
1.   }   1 second
1    }
```

When the operation is complete, the program writes:

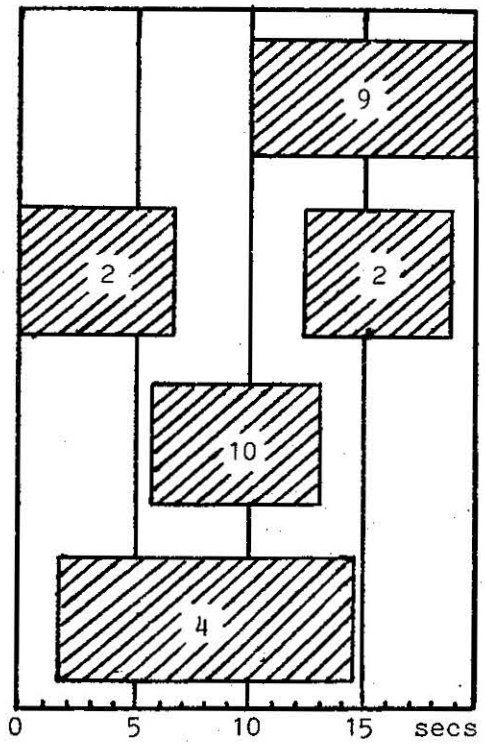
```
OBJECT NR n        MIX TIME f
```

where this is the n^{th} BLOCK to have been put on to the MIX-FILE, and the total duration of the MIX-FILE is "f" seconds. There is no limit to the number of BLOCKS that can be combined in a MIX-FILE.

[EOT] after "START TIME:" returns control to the program monitor without performing the mix.

Here is an example of a score for mixing compositional BLOCKS, and the instructions necessary to define the MIX-FILE.

```
>MX 2
START TIME:0
OBJECT NR 1  MIX TIME  6.5
>MX 4
START TIME:1.6
OBJECT NR 2  MIX TIME 14.7
>MX 10
START TIME:5.5
OBJECT NR 3  MIX TIME 14.7
>MX 2
START TIME:12.25
OBJECT NR 4  MIX TIME 18.75
>MX 9
START TIME:10.0
OBJECT NR 5  MIX TIME 20.0
>etc.
```



Note that BLOCKS need not be mixed in the order in which they start in the MIX-FILE. The order of mixing is important only when overlapping BLOCKS use the same generators, in which case the last-mixed BLOCK is valid.

4.5.2 MOVING-SOUND files, containing information about quadraphonic sound location and movements, can be created and mixed on to a MIX-FILE with:

```
>MS                          ( = MOVING SOUND)
```

The program asks:

1) START TIME:

and waits for a floating-point number defining the time in seconds at which this series of movements is to start.

EOT returns control to the program monitor without a moving-sound file being created.

2) MS#,DIRECT LEV,REVERB LEV:

Write here three whole numbers, separated by single spaces.

MS#	1 or 2	1: uses the four channel outputs; 2: uses the channel distributor.
DIRECT LEV	≥ 0	reference level for direct sound - unit = $\frac{1}{4}$dB
REVERB LEV	≥ 0	reference level for reverberation ($\frac{1}{4}$dB).

If any of the values given is illegal, the program writes:

ILLEGAL VALUE

and asks again:

MS#,DIRECT LEV,REVERB LEV:

EOT returns control to the program monitor without a moving-sound file being created.

3) DURATION,RADIUS,SHAPE,ANGLE,SHAPE:

Write here five whole numbers describing one spiral/circular movement of sound. RADIUS and ANGLE define the point where the sound will be located after "DURATION" milliseconds.

DURATION is the number of milliseconds the whole move takes; if it is zero or negative, the move will be instantaneous.

RADIUS 0 is the centre of the room, 1000 the distance from the centre to any one of the loudspeakers.

SHAPE is the curve-form (-9 to 9) describing the rate of change of the radius.

ANGLE 0° / 360° is midway between loudspeakers 1 and 2.

SHAPE is the curve-form (-9 to 9) describing the rate of change of the angle.

If the values given for SHAPE are outside the legal limits (-9 to 9) the program writes:

ILLEGAL VALUE

and asks again:

DURATION,RADIUS,SHAPE,ANGLE,SHAPE:

For more detailed information see the documentation of moving-sound in EMSETT (MOVSRA); but note that IMPAC does not simulate a Doppler effect.

* * *

The values given in answer to question (3) constitute one segment of a MOVING-SOUND file; the file can consist of any number of segments, and questions for new segments will be asked until end of input is signalled with [EOT]. The segments follow on directly from one another in the order in which they are defined.

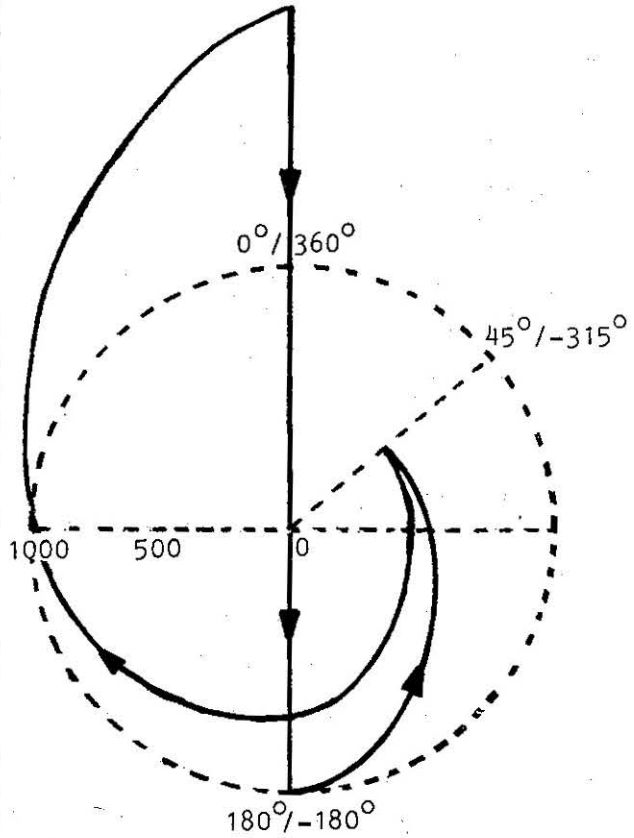
When [EOT] is pressed:

1) The MOVING - SOUND file is closed and given a number, such that the first MOVING-SOUND file created during a run is called "1 MVS", the second is "2 MVS", and so on.
The numbering of these files can be controlled with:

>FP 3 n

where n is the number to be given to the next MOVING-SOUND file created.

2) The MOVING - SOUND file is mixed automatically on to a MIX-FILE, and the program prints the OBJECT NR and MIX TIME.



The diagram on the previous page shows the way in which the MIX-FILE in §4.5.1 is to move.

The sound starts in the centre of the room - it is automatically there at the beginning of a run. Then it:

1) moves out to a point on the circumference between speakers 3 & 4;
2) spirals anticlockwise to: radius 500/angle 45°;
3) spirals clockwise to: radius 2000/angle 360°;
4) moves in a straight line into the centre.

Suppose that the MX instructions in §4.5.1 have already been given; and that each segment is to last 5 seconds. Define the movements as follows:

```
>MS

START TIME:0

MS#,DIRECT LEV,REVERB LEV:1 400 360

DURATION,RADIUS,SHAPE,ANGLE,SHAPE:5000 -1000 -2 0 0

DURATION,RADIUS,SHAPE,ANGLE,SHAPE:5000 500 0 -315 -1

DURATION,RADIUS,SHAPE,ANGLE,SHAPE:5000 2000 -4 360 1

DURATION,RADIUS,SHAPE,ANGLE,SHAPE:5000 0 -3 0 0

DURATION,RADIUS,SHAPE,ANGLE,SHAPE: [EOT]

1     MVS OK

OBJECT NR 6  MIX TIME 20.0

>etc.
```

N.B.

a) The user must himself make sure that suitable connections are made on magnetic tape for moving-sound to function properly.

- For two moving-sounds (FGs 1 to 12 on MS#2, and FGs 13 to 24 on MS#1) use:

  ```
  >CN 4
  ```

- For one moving-sound (FGs 1 to 24 on MS#1) use:

  ```
  >CN 1
  ```

Remember that the MT switch must be on for connections to be written to magnetic tape. (See §§2.5.1 ff.)

b) Sampling-time for moving-sound can be defined separately from the normal music sampling-time. Format:

```
>SA n1 n2
```

where n1 is a whole number from 1 to 65536 giving the sampling-time in milliseconds;
n2 is any non-zero number.

The time defined here should be greater than the music sampling-time; if it is not, it will be ignored.

4.5.3 By calling previously defined MOVING-SOUND files, the user can repeat movement-patterns exactly. Format:

```
>MS n
```

where n is a positive whole number specifying the file name (1 MVS, 2 MVS, etc.). The series of movements defined on the previous page, for instance, can be remixed at a later stage with:

```
>MS 1
```

The program asks, as usual:

```
START TIME:
```

and, when the mix is complete, prints the OBJECT NR and MIX TIME. The same pattern of movements is thus mixed on to the MIX-FILE, but starting at a new time. If MOVING-SOUND files with the same MS# should overlap, the file that is mixed last is valid.

4.5.4 The current MIX-FILE can be saved with:

```
>SM                          ( = SAVE MIX-FILE)
```

Files saved in this way are numbered automatically, such that the first MIX-FILE saved becomes "1 MIX", the second is "2 MIX", and so on. If there is no current MIX-FILE, the program writes:

```
NOTHING ON MIX DISK
```

and control reverts to the program monitor.
The numbering can be directed with:

```
>FP 2 n
```

where n is the number to be given to the next MIX-FILE saved.

N.B.
When SM has been done, there are two identical files on the disk - the saved file ("n MIX") and the internal MIX-FILE. Future MX commands do not, therefore, start from scratch, but continue to mix BLOCKS on to this internal MIX-FILE.

4.5.5 When a MIX-FILE has been saved, it can be mixed again on to the current MIX-FILE with:

```
>MM n                        ( = MIX MIX-FILE)
```

where n is a positive whole number specifying a file that has been saved with SM.

Suppose for example that BLOCKS have been mixed as in §4.5.1, and that exactly the same sequence is to be repeated immediately. Write:

```
>SM

1     MIX OK

>MM 1

START TIME:20.0

OBJECT NR 6  MIX TIME 40.0
```

Remember that the current MIX-FILE remains on the disk even after "SM"; so only one "MM" is required, even though the mixed BLOCK is to appear twice.

4.5.6 With the command:

```
>AP n                    ( = APPEND)
```

it is possible to append a previously saved MIX-FILE, "n MIX", on to the end of the internal MIX-FILE. It is thus the equivalent of:

```
>MM n

START TIME:end of current MIX-FILE
```

If there is no file "n MIX" on DK «USE», the program writes:

```
n     MIX NOT FOUND
```

and control reverts to the program monitor.

4.5.7 The internal MIX-FILE can be deleted with:

```
>DM                      ( = DELETE MIX-FILE)
```

There are two occasions when this operation may be necessary:

1) when an error has been made in mixing. Suppose the following commands are given:

```
>MX 2

START TIME:10.0

OBJECT NR  1  MIX TIME  25.0
```

and it is then realized that it was BLOCK "3" that should have been mixed, or that START TIME should have been "1.0". The only way to repair the damage is to delete the internal MIX-FILE and start again.

2) when SM has been done, and a new independent MIX-FILE is to be started. Thus:

```
>SM

1     MIX OK

>DM                                  delete internal MIX-FILE
                                     (1 MIX is still on disk)
>MX 4                                this is written on to a
                                     new,clean MIX-FILE
START TIME:etc.
```

4.5.8 The duration of the current internal MIX-FILE can be obtained with:

```
>TX
```

4.5.9 MIX-FILES cannot be played directly in the studio; but they can be written to magnetic tape to be played later with the program A2 - cf §2.5.4. Format:

```
>PM
```

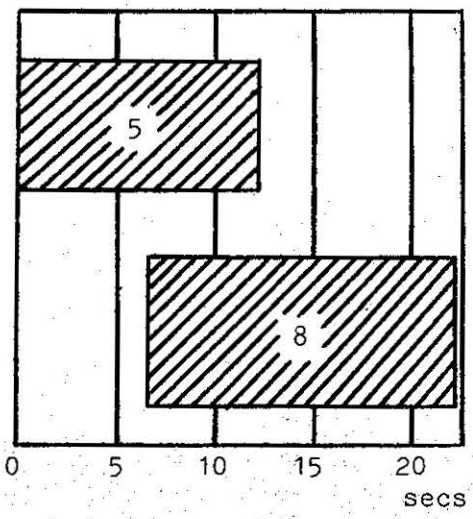
which reads from the currently active internal MIX-FILE and writes music information on magnetic tape. If there is no internal MIX-FILE, the program writes:

```
NOTHING ON MIX DISK
```

and control reverts to the program monitor.

The MT-switch need not be on when PM is started, but connections must have been written to magnetic tape at some stage; see §2.5.3.

Suppose that two BLOCKS which have already been defined, "5 BIN" and "8 BIN", are to be mixed and transferred to magnetic tape:




```
>MX 5
START TIME:0
OBJECT NR 1 MIX TIME 12.0
>MX 8
START TIME:6.5
OBJECT NR 2 MIX TIME 22.0
>MT                          turn on MT-switch
>CN                          and write connections to
                             magnetic tape
>PM                          PLAY-MIX, and wait while
                             the MIX-FILE is written
>etc.                        to magnetic tape
```

N.B.

At the end of the above operation, the internal MIX-FILE is deleted, so that further MX commands start a new MIX-FILE; if the same mix-sequence is to be used again "SM" (SAVE-MIX) must be done before PM.

4.5.10 Summary of the MIX commands

MX (MIX)	SM (SAVE MIX)	TX (TIME OF MIX-FILE)
MS (MOVING-SOUND)	DM (DELETE MIX)	
PM (PLAY MIX)	MM (MIX MIX-FILE)	
	AP (APPEND MIX-FILE)	

4.5.11 WARNING!

In view of the proposed hardware developments in the EMS studio - analogue multi-track tape-recorder and mixer - the MIX subprogram in IMPAC has not been altered since its first conception; it is therefore in many ways primitive. Note the following:

1) The FM generators cannot be used. There is no error message.

2) Only the parameters of the composition-program are valid; the command "SP" (SET PARAMETER - see §4.1.12) has no function here. There is therefore:

- no vibrato (VS = 0, VD = 0);
- no attack time (AT = 0);
- no modulation index or modulation frequency;
- moving-sound and reverberation are controlled with MS files.

3) The execution of PM requires a large amount of memory space: it uses the same space as the various tables and stores of the real-time program. The user who has been working with the real-time program is therefore advised to save these stores with "PT" (see §5.9.2) before doing "PM".

4.6 Miscellaneous commands

4.6.1 >TM[n] (= TIME)

where n is the number of a BLOCK-file.

1) If n is undefined, negative or zero, the program prints the duration of the music that was last played in the studio.

2) If n is a positive whole number, the program prints the duration of BLOCK-file "n BIN".

4.6.2 >IN (= INITIALIZE)

returns the composition program to its state at the start of the run.

1) All BLOCK-files, from "1 BIN" to the latest defined BLOCK-file inclusive, are deleted, except those that have been created with KP; the file-pointer (set with >FP 0 n) is put to 1. MIX-files and MOVING-SOUND files are not deleted.

2) If the KEEP-file 999 is active, it is closed and numbered in the normal fashion.

3) All parameters in the BLOCK table are put to default values:

SE 50	DE and DS 1	FG 25 40
DU 131000	F1 and LF 16	ND 10000
	F2 and HF 3000	GL 1000
	A1 and LA 280	WF 0
SA 5	A2 and HA 320	

4) The MT-switch is turned off (but no stop-mark is written).

5) The tables and stores used in the real-time program are cleared and put to default values.

6) The Tektronix screen is cleared, as are all studio registers.

7) The program writes:

IMPAC Vnn

where nn specifies which version of IMPAC this is.

5 The Real-time Program

5.1 Introduction

In the real-time program the user controls the music parameters while the music is being played; he can connect the parameters to various analogue or programmed devices, or choose to control them from a BLOCK file.

5.1.1 The real-time program is entered with:

```
>PL                    ( = PLAY)
```

Control is returned to the program monitor when

1) the SPACE key is pressed; or
2) the end of the BLOCK file being played is reached.

5.1.2 The normal working procedure at the beginning of a run is:

```
>CN                    make studio connections
                       (see §2.4.1)
>RV n1 n2              set reverberation time
                       and level (see §2.4.3)
>PL
```

The program reads information about parameters from the default file "0 BIN"; or, if a BLOCK number has been given in the "PL" command, a file with this number is read from instead.

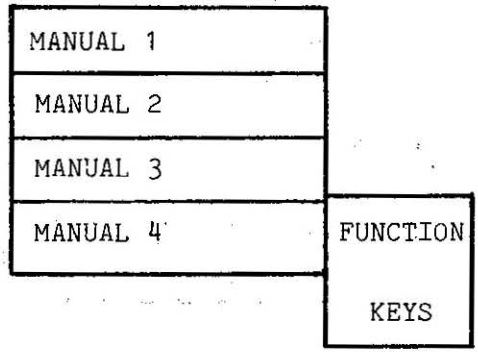
"0 BIN" contains the same values as the default values in the BLOCK-table (see §4.6.2). This file will play for 1 hr 49 min 10 sec, unless the SPACE key is pressed.

SE	50	DU	131000
F1	16	F2	3000
A1	280	A2	320
DE	0		
FG	25 40	WF	0
GL	1000	ND	10000

5.1.3 Instructions are given by pressing the keys on the Tektronix in certain special ways; the characters printed on the keys are of no significance!

Instructions are of five types:

- Parameters
- Devices
- Multiple-key functions
- Single-key functions
- Keyboard



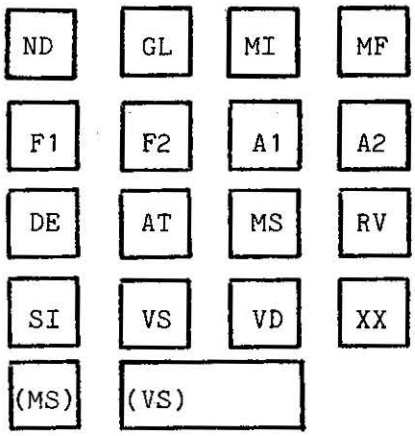
A full plan of the Tektronix as it is used in the real-time program is to be found in §§ 6.7 and 6.8.

Note that groups of keys will be referred to as MANUALS 1 to 4 and FUNCTION KEYS, as in the diagram above.

5.2 The parameters and devices

There are sixteen parameters, represented by the sixteen function keys on the right-hand side of the Tektronix.

5.2.1 The music parameters are: ND, GL, MI, MF, F1, F2, A1, A2, DE, AT, MS, RV, VS and VD; these are described fully in ch 3. SI and XX are discussed later.



To reference a parameter, press the key associated with it; when this is done, the current value of the parameter is written on the screen. Thus:

10000.000

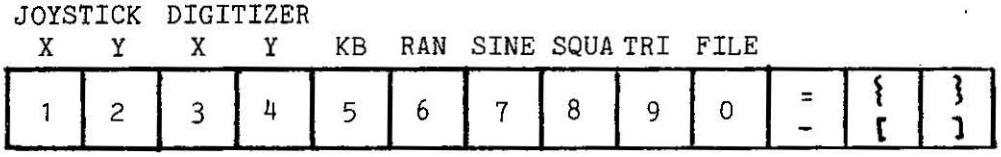
5.2.2 The devices used to control the parameters are:

1) JOYSTICK X-axis
2) JOYSTICK Y-axis
3) DIGITIZER X-axis
4) DIGITIZER Y-axis
5) KEYBOARD (MANUAL 3)
6) RANDOM GENERATOR
7) SINE-WAVE GENERATOR
8) SQUARE-WAVE GENERATOR
9) TRIANGULAR WAVE GENERATOR
0) FILE

1 - 5 are controlled manually.
6 - 9 are software devices. See §§5.6ff for control facilities.
0 is not, strictly speaking, a device - it represents the information on the BLOCK file being played.

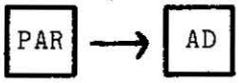
For the sake of convenience, all the devices - analogue and programmed - are referred to as ANALOGUE DEVICES (AD).

The devices are represented on the Tektronix by the corresponding numbers in Manual 1.

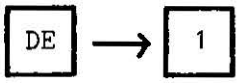


(KB = KEYBOARD)

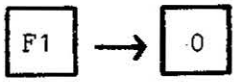
5.2.3 A parameter is connected to a device by pressing the parameter key followed by the device key.



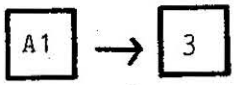
If, for example, DE is to be controlled by the JOYSTICK X-axis, press:

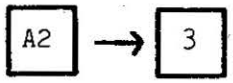


If F1 is to be controlled by information in the BLOCK file, press:



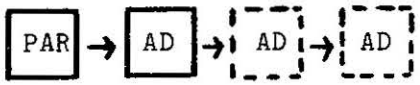
There is no limit to the number of parameters which may be controlled by a given device at any one time. Thus the sequence:



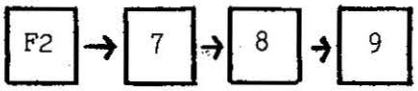


means that the amplitude limits A1 and A2 are controlled in parallel by the digitizer X-axis.

5.2.4 Parameters can also be controlled by combinations of devices:

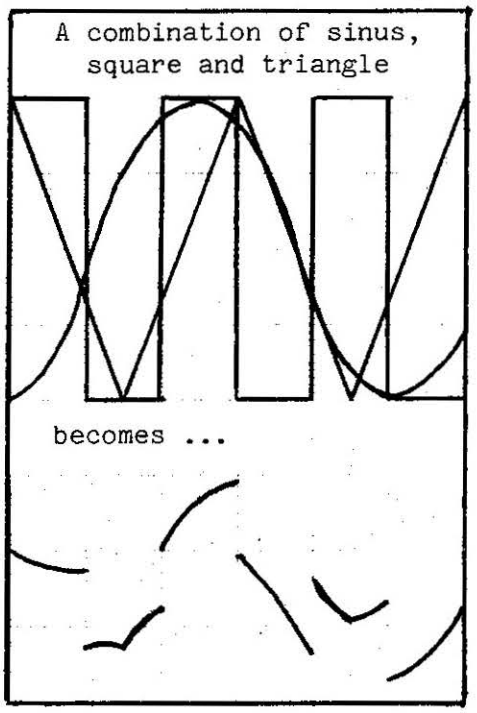


Control is then divided in equal proportions between these devices; so if F2 is connected thus:

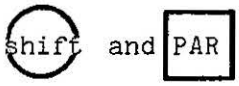


each of the devices contributes 1/3 towards the result.

The exception to this multiple-connection facility is device 0 (FILE), which cannot be combined with any other device; as soon as the FILE key is pressed, the relevant parameter is controlled from file only.



5.2.5 It is often desirable to find a suitable value for a parameter with one of the analogue devices, and then freeze the parameter there; i.e. leave it at that value while the device is freed to control other parameters. This is done by pressing:



at the same time. The parameter can be unfrozen at a later stage by repeating the action - it functions therefore as an on/off switch.

When a parameter is frozen, its value will not change until
a) it is unfrozen; or
b) it is connected to a device.

For example, to find a suitable value for MI with JOYSTICK Y-axis:

1) [MI] → [2]

2) Move the JOYSTICK until the right value is found.

3) shift and [MI]

to freeze the parameter.

4) When the parameter is to be changed again:

shift and [MI]

5) To control MI with another device, e.g. the random generator, press:

[MI] → [6]

5.2.6 Exact values can be put on the parameters by giving the command:

>SP

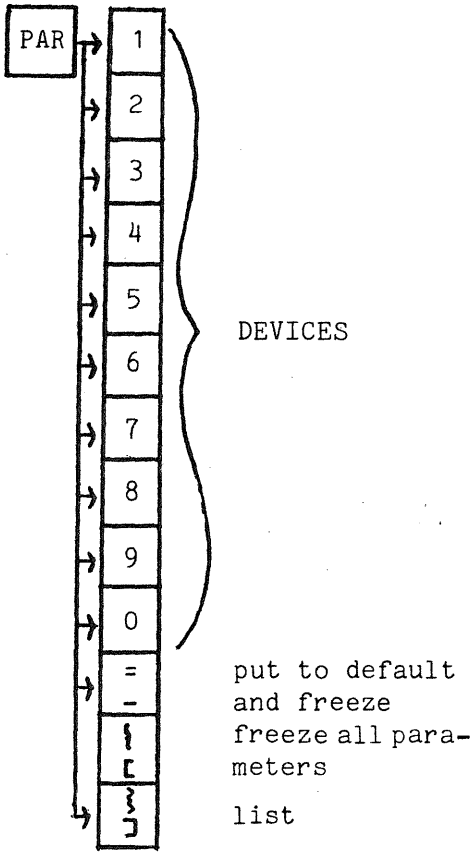
from the program monitor (see §4.1.12 for format and other details). Note that when this is written the parameter in question is automatically frozen.

5.2.7 All the keys in MANUAL 1 except [ ], which freezes all parameters, have varying functions depending on which key is pressed immediately before them. Keys 1 to 9 and 0, if preceded by a parameter key, make connections to the analogue devices; their functions with other keys are discussed later.

After a parameter key:
[= -] puts one parameter to its default value and freezes it;

ND	10000	GL	1000
MI	100	MF	1000
F1	16	F2	3000
A1	280	A2	320
DE	0	AT	20
MS	500	RV	240
SI	4	VS	100
VD	5	XX	0

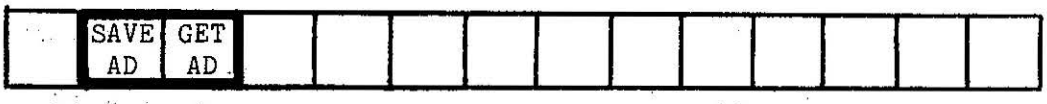
[ ] writes the value of the latest parameter pressed.



5.3 The AD table

The program maintains a table containing information about parameter/device connections and parameter ranges. This is called the AD (ANALOGUE DEVICE) TABLE.

5.3.1 The AD-table is referenced with two keys in MANUAL 2.



For example, the table can be listed by pressing either of these keys followed by the LIST key in MANUAL 1.

Column 1 contains the names of the parameters.

Column 2 contains the numbers of the devices to which the parameters are connected; for example, 789 indicates that the parameter is connected to devices 7, 8 and 9. An asterisk immediately before the device number shows that the parameter is frozen.

Columns 3 and 4 show the limits of the parameter range - from column 3 when the device has a low value (left or bottom) to column 4 when it has a high value (right or top). Note that in the example VD's left limit is higher than the right one; this means that when the random generator produces a low number, VD gets a high value, and vice versa. Vibrato depth is thus made inversely proportional to vibrato speed.

ND	789	100	10000
GL	*2689	0	8192
MI	1	0	1000
MF	2	25	1000
F1	0	16	3000
F2	0	16	3000
A1	57	240	320
A2	57	260	340
DE	*1	0	200
AT	3	0	1000
MS	0	19	20
RV	*0	240	360
SI	0	0	2880
VS	6	150	200
VD	6	20	10
XX	0	0	511

5.3.2 It has already been shown how parameter/device connections can be made in real-time. Connections and parameter ranges can also be altered from the program monitor with the command:

```
>AD n                    ( = ANALOGUE DEVICE)
```

where n is the number of a device (0 to 9) to which parameters are to be connected.

"AD" is a subprogram with its own submonitor; it prints:

```
:
```

and waits for the user to give one of the following commands:

Parameters				Other functions
:ND n1 n2	:GL n1 n2	:MI n1 n2	:MF n1 n2	:L (= LIST)
:F1 n1 n2	:F2 n1 n2	:A1 n1 n2	:A2 n1 n2	:S n (= SAVE)
:DE n1 n2	:AT n1 n2	:MS n1 n2	:RV n1 n2	:G n (= GET)
:SI n1 n2	:VS n1 n2	:VD n1 n2		:AD n (= ANALOGUE DEVICE)

Control reverts to the program monitor when EOT or carriage return is done immediately after ":".

1) PARAMETER COMMANDS

n1 and n2 specify the new parameter range. Thus:

```
:F1 220 440
```

gives "F1" a range between 220 and 440 Hz. At the same time it is connected to the device stated in the AD-command. For example:

```
>AD 5                    we're connecting to device 5
:A2 300 350              A2 with these limits...
:RV 400 200              and reverb from 400 down to 200
:                        empty line to return to program monitor
>etc.
```

MS (moving-sound) is defined in a slightly different way. Here n1 and n2 specify which output channels the sound is to move between; the numbers used are the standard EMS designations.

n1 is the left channel (MS = 0), and n2 is the right channel (MS = 1000). For example, to move the sound between channels 2 and 4, write:

```
:MS 20 22
```

CHANNEL OUTPUTS	CHANNEL DISTRIBUTOR
1: 19	1: 15
2: 20	2: 16
3: 21	3: 17
4: 22	4: 18

Both numbers must come from the same group of amplifiers, 19 to 22 or 15 to 18.

Note the following special cases:

a) Parameter ranges <u>cannot</u> be altered after:

```
>AD 0
```

which connects parameters to BLOCK-file information; this is because ranges have no significance when a file has control.

b) To put all parameter/device connections to zero, write:

```
>AD -1
```

c) To alter parameter ranges without defining parameter/device connections, write:

```
>AD 10
```

and then parameter commands as usual; the connections in the table remain unchanged.

N.B.
All parameters can theoretically be connected to BLOCK-files; however, files have information for ND, GL, F1, F2, A1, A2 and DE only. If the other parameters are connected to "0", they are, in effect, frozen.

2) To list the AD-table, write:

:L

This is also done automatically when the AD-subprogram is entered.

3) To choose a new device for connection to parameters, write:

:AD n

This performs the same function as the program-monitor command "AD". For example:

>AD 6	connect random generator
:MI 0 250	to modulation index
:AD 7	connect sine-wave generator to ...
:MF 500 750	MF-ratio
:AD 10	no device!
:F1 16 100	so change range only
:etc.	

4) The values in the AD-table can be stored with:

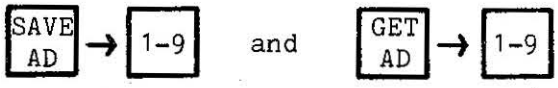
:S n

where n is a whole number from 1 to 9 specifying which of the nine AD-store positions this table is to go into. Values which have been saved in this way can be retrieved with:

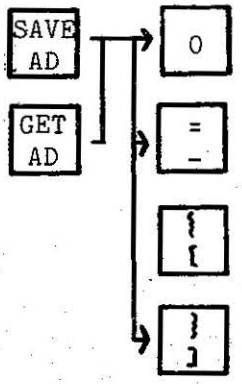
:G n

where n specifies which of the nine AD-stores is to be referenced. Both of these commands cause the Tektronix screen to be cleared and the AD-table to be listed.

5.3.3 In the real-time program the AD-stores are referenced with:



With the other keys in MANUAL 1, SAVE and GET have the following functions:

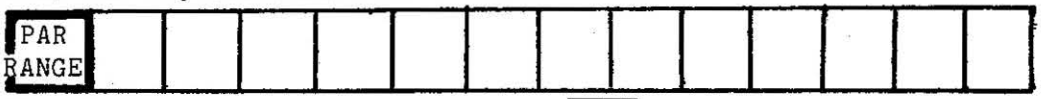


put all connections to zero

complement all connections, i.e. unfreeze all frozen parameters, and vice versa

list the current AD-table

5.3.4 A certain amount of control can be exercised over parameter ranges with the key PAR RANGE in MANUAL 2.

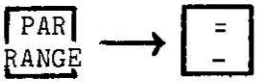


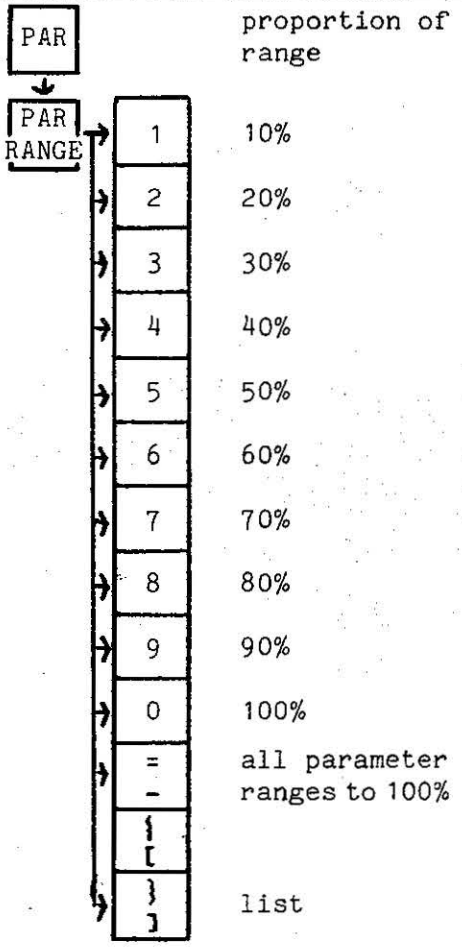
Three keys are pressed in this operation:

1) the parameter whose range is to be modified;
2) PAR RANGE
3) one of the keys in MANUAL 1.

The actual range then becomes a given percentage of the original range in the AD-table. This is done by lowering the upper limit; the lower limit is not changed. For example, suppose that in the AD-table F1 is to lie between 100 and 900 Hz - a range of 800 Hz. After:

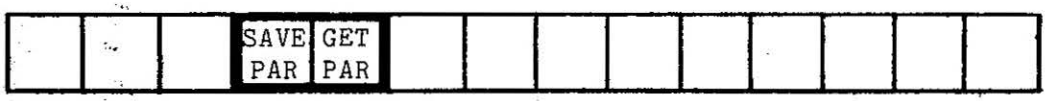
the range becomes 60% of 800, which is 480; so the upper limit is decreased to (480 + the lower limit), which is 580 Hz. But note that the original range limits always appear when the AD-table is listed. All the parameter ranges are restored to their original 100% values with:



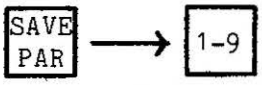


5.4 The Parameter table

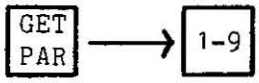
The PAR (parameter) TABLE contains the current values of all 16 parameters. It is updated each sample by any analogue devices and BLOCK-file information that may be controlling parameters; and it is from this table that music is calculated.



5.4.1 There are nine positions in memory reserved for storing the PAR table. These are accessed with:



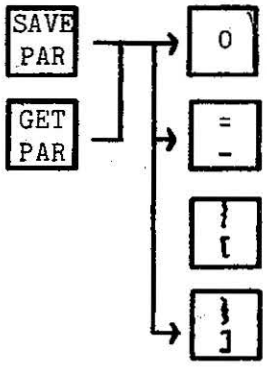
and



which saves the whole of the current PAR table (i.e. the values of all the parameters at the moment the number key is pressed) to one of the positions in the PAR store.

which retrieves values that have been stored with SAVE PAR, and puts them in the PAR table; then it freezes all connections in the AD-table.

The other keys in Manual 1 have the same functions for both SAVE PAR and GET PAR.

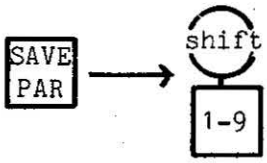


put all parameters to legal minimum and freeze them.

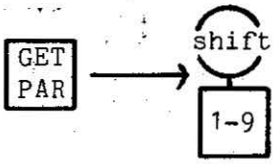
all parameters to default and freeze; see §5.2.7 for default values.

list PAR table.

5.4.2 The AD and PAR stores can be accessed simultaneously with:



and



which stores the current AD-table in position "n" of the AD-store, and the current PAR-table in position "n" of the PAR-store.

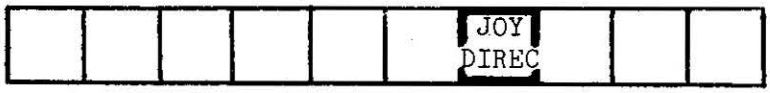
which fetches values from position "n" of the AD-store and position "n" of the PAR-store and puts them in the AD and PAR tables respectively.

5.5 The Joystick Store

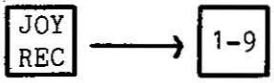
Movements with the Joystick can be stored in the JOYSTICK-store, and then recalled later to reproduce exactly the same, or slightly modified, sequences. Two keys in Manual 2 - JOY REC and JOY PLAY (Joystick Record and Joystick Playback) - are used to start storage and retrieval of Joystick information, while three keys are used to exercise control over playback - JOY DEL (Joystick Delay) and JOY SAMP (Joystick Sampling) in Manual 2, and JOY DIREC (Joystick Direction) in Manual 4.

Manual 2

Manual 4



5.5.1 There are nine memory banks in the Joystick store, each with room for 1023 pairs of coordinates; during recording, one pair of joystick coordinates is stored every sample. To start recording, press:



where the number specifies one of the nine memory banks

Recording is completed:

1) after 1023 samples;

2) after

3) when recording is re-initialized.

4) when playback is started.

5.5.2 To play back Joystick movements, press:

where the number specifies one of the nine memory banks.

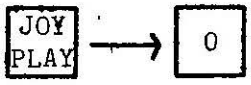
Each sample, one pair of coordinates is fetched from the store, and these coordinates replace information coming from the actual joystick; so parameters connected to the joystick (devices 1 and 2) are controlled by the store information instead.

Playback is completed:

1) when another playback is started;

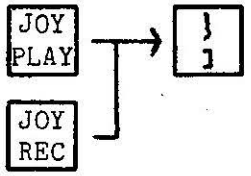
2) when joystick recording is started;

3) after



Otherwise, each memory bank is treated as if it were circular - the last pair of coordinates is followed by the first.

Note also:



writes the number of the current memory bank being played back.

5.5.3 Playback from Joystick store can be manipulated in a number of ways. Normally, a new pair of coordinates is retrieved each sample; with:

where n is a number: 1 - 9 or 0

the user can control the rate at which information is fetched, such that each pair is valid for n + 1 samples. For example:

means that new coordinates are fetched every tenth sample; playback speed is therefore one tenth of the speed during recording.

restores the original speed.

writes the current delay factor.

5.5.4 With the function:

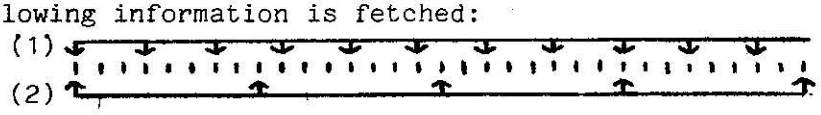
where n is a number: 1 - 9 or 0,

the user controls the playback sampling rate, in such a way that n pairs of coordinates are skipped each time new values are fetched. For example:

(1) 

(2) 

The following information is fetched:



JOY SAMP → 0 restores the original sampling rate.

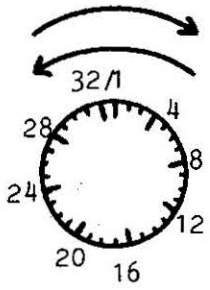
JOY SAMP →] writes the current sampling rate.

The difference between JOY DEL and JOY SAMP is thus that JOY SAMP determines which coordinate-pairs are to be fetched, while JOY DEL determines how long each pair is to be valid.

5.5.5 The direction of playback can be changed with:

JOY DIREC

This is a single-key switch - each time it is pressed, the direction changes. A memory bank that is read backwards is still circular; the last coordinate-pair follows the first.



5.5.6 Summary of the Joystick-store functions.

JOY REC → 1-9 start recording to memory bank "n".

→ 0 stop recording

→ = -

→ [{

→] } write memory-bank number

JOY PLAY → 1-9 start playback from memory bank "n"

→ 0 stop playback

→ = -

→ [{

→] } write memory-bank number

JOY DEL → 1-9 set delay factor n

→ 0 no delay

→ = -

→ [{

→] } write delay factor

JOY SAMP → 1-9 skip n points each sample

→ 0 no skip

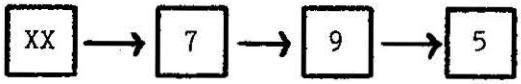
→ = -

→ [{

→] } write sampling factor

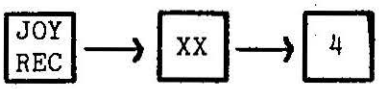
5.5.7 The dummy parameter XX is used to store information in the Joystick store; it is like the other parameters in that it can be controlled by one or more devices, but it has no direct effect on the musical result. Its primary purpose is to construct complex wave-forms and record them in to one of the Joystick memory banks, so that they can be fetched with JOY PLAY and used to control other parameters. For example:

1) Construct a complex wave-form on XX:

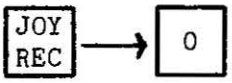


- a combination of sinus and triangular waves and manual keyboard control.

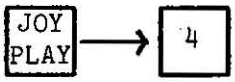
2) Record this wave-form into memory bank 4:



3) Stop recording:



4) Play back memory bank 4:



- parameters that are connected to device 1 (Joystick X-axis) are controlled by the complex wave-form. Parameters connected to device 2 get a slow sawtooth wave-shape, rising from minimum to maximum in 512 samples.

5.6 Controlling the devices

Four of the "analogue devices" - the sinus, triangle, square and random generators - are program-controlled.

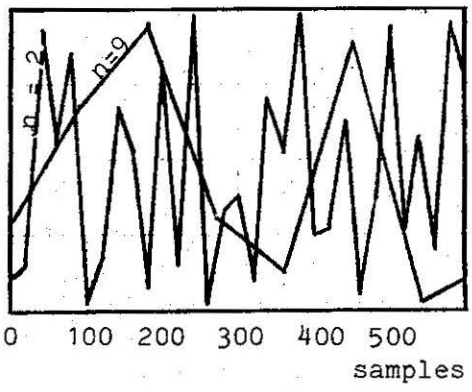
5.6.1 The sine-wave generator is controlled by the parameter SI (SINE-WAVE INCREMENT), which determines the number of values in the sine-wave table that are to be skipped each sample. If SI = 10, the generator takes every tenth value from the table; if SI = 200, it takes every two-hundredth value; if SI = 0, the generator stops, and so on.

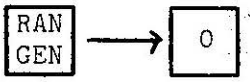
The maximum value for SI is 2880 (see §6.10 for more information about the sine-wave table.

SI can be connected, listed, frozen, etc. in exactly the same way as every other parameter.

5.6.2 The random generator is controlled with:

where n is a number: 1 - 9 or 0, such that a new random number is chosen every (n x 10) samples. For the other samples, values are calculated on a linear-interpolation basis between the random numbers before and after.





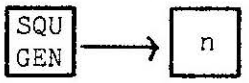
ensures that a new random number is chosen every sample.

5.6.3 The triangular-wave generator is controlled with:

where n is a number: 1 - 9 or 0,

such that the period of the generator is (n + 1) x 20 samples.

5.6.4 The square-wave generator is controlled with:

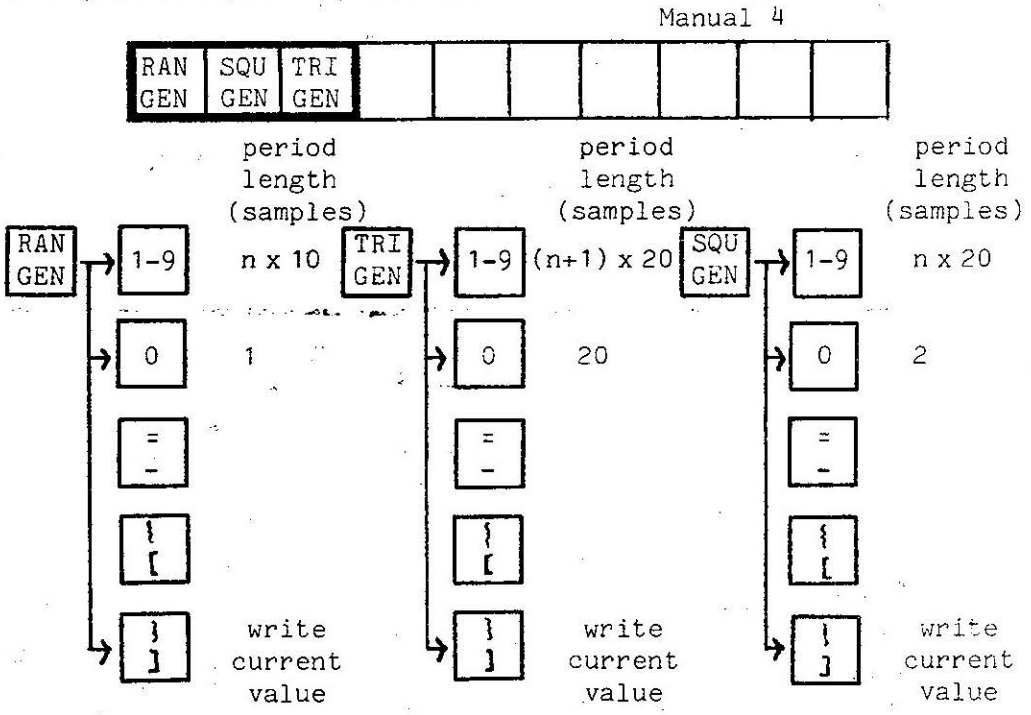


where n is a number: 1 - 9 or 0,

such that the period of the generator is n x 20 samples.

gives the shortest possible period: 2 samples.

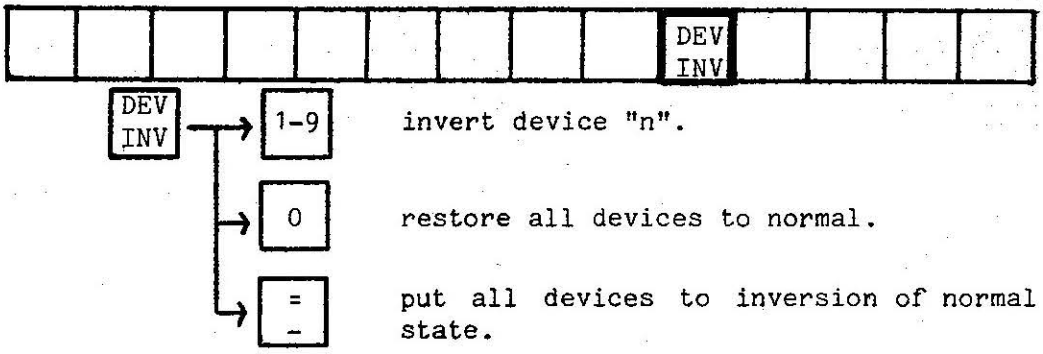
5.6.5 Summary of the periodic devices



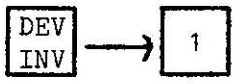
5.6.6 Control of the Joystick, Digitizer and Tektronix Keyboard (device 5) is manual, and should present no problems. Note however that only the region 0 to 1100 of the Digitizer is used; coordinates below or above these limits are adjusted to 0 or 1100 respectively.

5.7 Miscellaneous multiple-key functions

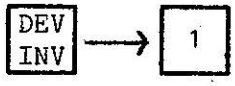
5.7.1 The control exercised by the analogue devices can be inverted with the DEVICE INVERSION key in Manual 2.



For example:



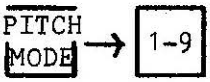
means that the Joystick X-axis will give high values to the left, and low values to the right; values read from the Joystick-store are inverted in the same way. Pressing:



again restores the device to its original state; DEV INV functions therefore as an on / off switch.

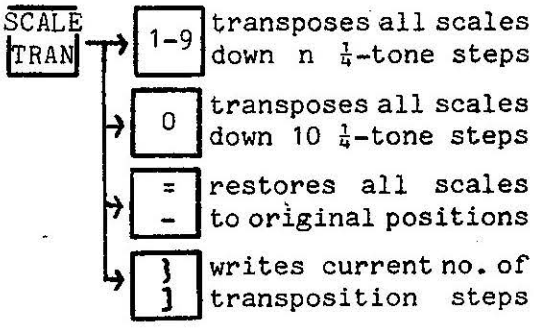
5.7.2 Interval relations and scale-patterns are controlled with PITCH MODE and SCALE TRANSPOSITION in Manual 2.

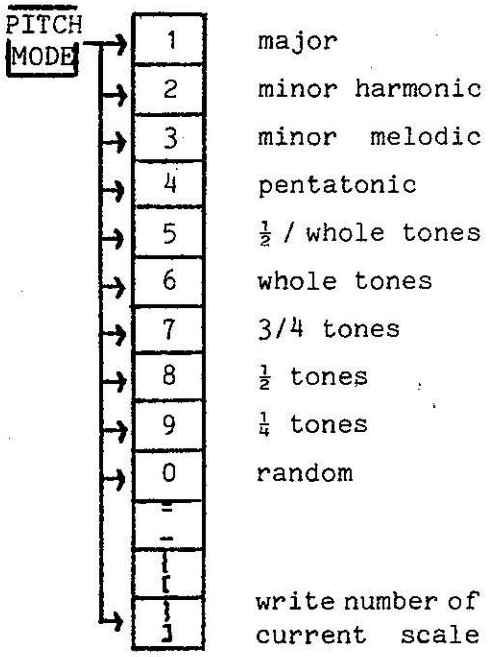
With PITCH MODE the user chooses between different scales; nine are pre-defined and can be accessed with:



Any or all of these scales can be replaced by scales of the user's own choice - see §3.4.5. After:

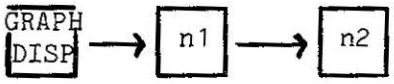
frequencies are chosen on a purely random basis.





5.7.3 The GRAPHIC DISPLAY key in Manual 2 is used to plot parameters and devices.

Format:

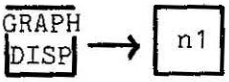


where n1 and n2 are devices (1 - 9) or parameter keys.

1) When n1 is pressed, that parameter or device is plotted on the Y-axis, against time on the X-axis.

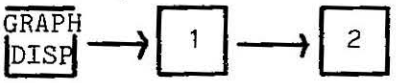
2) When n2 is pressed, the first parameter or device is moved over to the X-axis, and the second is plotted on Y.

N.B.
After stage 1 above, the program will interpret the next pressing of a parameter key as a GRAPH instruction, even if the user intends it to be part of a device connection. If it is desired to plot one parameter or device against time, the user is recommended to press any key in Manual 2 except GRAPHIC DISPLAY immediately after

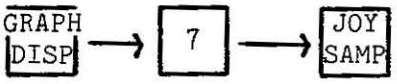


Examples:

a) To plot Joystick-X against Joystick-Y:

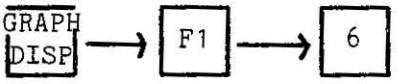


b) To plot the sine-wave generator alone:

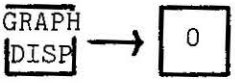


(or any other key in Manual 2)

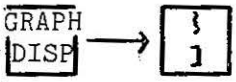
c) To plot F1 against the random generator:



d) To stop GRAPHIC DISPLAY:



e) To see which parameters and devices are being plotted:



This information is written as "X Y": parameters are represented by their two-character mnemonics, devices by the numbers 1 to 9; time on the X-axis is represented by T. The above examples give:

a)	1	2	b)	T	7
c)	F1	6	d)	0	

Graph information is drawn on the screen at a rate of one point per sample, except that the program does not allow text messages and graphs to be output at the same time. Text messages have priority; so graphic display is temporarily disenabled while messages are being written.

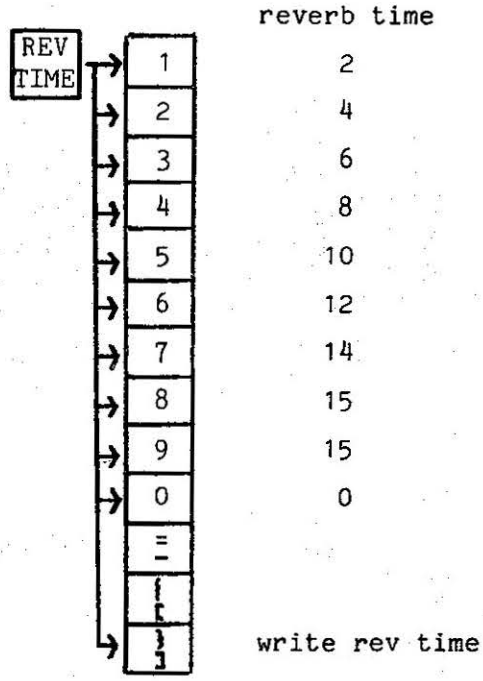
Scaling with GRAPHIC DISPLAY:

- Devices are scaled so that the minimum value is drawn at the lower or left edge of the screen while maximum is drawn at the upper or right edge.
- Parameters that are connected to 0 (file-information) are scaled from zero to legal maximum.
- Parameters connected to devices (1 - 9, frozen or unfrozen) are scaled between the minimum and maximum values in the AD table.

5.7.4 Reverberation time is controlled with:

REV TIME → n

where n is a number (1 - 9 or 0) giving reverberation times as in the diagram.



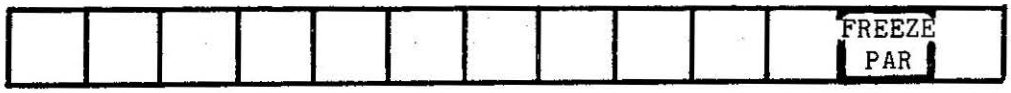
5.8 Single-key functions

There are nine functions which are operated by pressing a single key.

5.8.1 All parameters are frozen with:

FREEZE PAR

Note that this is the only key in Manual 1 with a fixed function; it does not matter which key was pressed before.



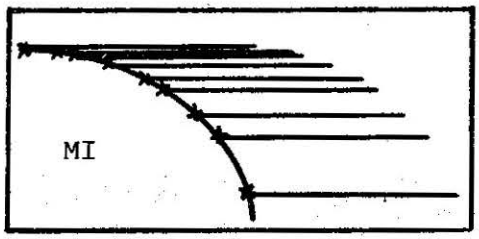
5.8.2 The RIGID/FLUID key is an on/off switch which allows parameters two completely different modes of operation.

In RIGID-mode the parameter values valid at the beginning of a note are kept throughout the duration of the note.

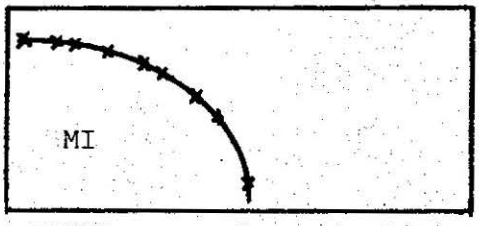
In FLUID-mode, every note follows changes made in the parameters; so all generators that are playing notes at a given moment have exactly the same values for ND, GL, MI, MF, AT, VS, and VD.

N.B.

1) Wave-form is always rigid.
2) At the start of a run, this switch is set to "FLUID".



RIGID - start-values follow the curves, but each note keeps its own value.

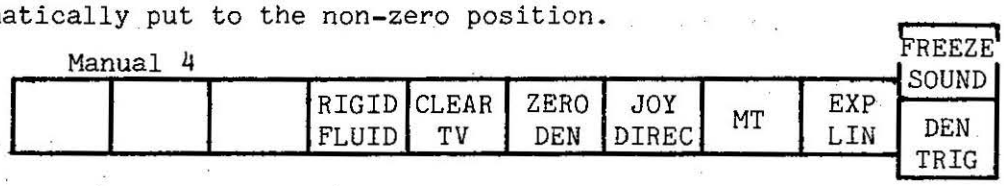


FLUID - new notes, starting at x, follow the MI curve.

5.8.3 The CLEAR SCREEN key clears the screen, stops any message that is in the process of being written, and sets a program flag to ensure that the next message is written at the top of the screen. Graphic display is not affected.

5.8.4 ZERO DENSITY puts density to zero no matter what the actual value on the parameter DE; no new notes can start, though all old notes will complete their envelopes as usual. It is an on / off switch: pressing it a second time returns control to DE.

Every time playing is started in the studio, this switch is automatically put to the non-zero position.



5.8.5 JOY DIREC changes the direction in which data is fetched from the Joystick store. See §5.5.5.

5.8.6 With MT ON/OFF it is possible to enable/disenable storage on magnetic tape. When it is "on", music is written to magnetic tape as well as to the studio; when it is "off", music is played in the studio only.

When the switch is turned on, a check is made first to see if connections have been written to magnetic tape; if they have, then all is well, and music can be written there too. If not, the message

CONNECT MT!

is written on the screen, the music stops, and control reverts to the program monitor. Connections can then be written to tape. (See also §§2.5 ff.)

5.8.7 The switch EXP/LIN controls the distribution of random frequencies between F1 and F2.

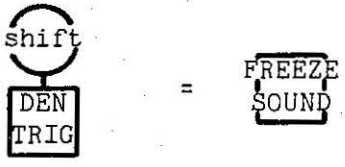
EXP = exponential (logarithmic)
LIN = linear (rectangular)

At the start of a run this switch is set to EXP.

5.8.8 When DENSITY TRIGGER is pressed, all the generators which are being used - those defined with >FG n1 n2 - start a new note simultaneously; in other words, density is put to its maximum legal value for one sample only.

5.8.9 FREEZE SOUND is an on/off switch which is operated by pressing SHIFT and the DENSITY TRIGGER key at the same time. As long as it is on, all studio functions are frozen; no changes are made on any of the generators, reverberation units or amplifiers. However, graph functions, messages, and calculation of devices and parameters continue as normal.

This switch is automatically "off" every time playing is started in the studio.



5.9 Program-monitor commands and real-time

There are a number of program-monitor commands which exercise control over the music played with the real-time program.

5.9.1 Some of these commands have already been discussed:

>SP (SET PARAMETER) sets single values on parameters, and freezes them (see §4.1.12).

>AD (ANALOGUE DEVICE) controls the AD table, with information about parameter/device connections and parameter ranges (see §§5.3 ff).

>SA (SAMPLING TIME) controls the rate at which information is sent to the studio (see §§3.4.2 and 4.1.8).

>ED (EDIT) makes changes to previously created BLOCK-files (see §§4.3 ff); it can also be used to alter the default file 0 BIN, which is played if no definitions have been made with the composition program. In fact, this is the only way to control the number of generators and the wave-forms to be set on them.

Suppose, for example, that at the beginning of a run the user wishes to play music on generators 1 to 12 with wave-form 5; but in all other respects, the default file is to be used:

```
>ED                                  use EDIT to change gen-
:FG 1 12                             erator and wave-form de-
:WF 5                                finitions
:
0     BIN OK
>PL                                  play the result, with
STOPPED IN SEG     1                 real-time control.
>ED                                  now try it with genera-
:FG 25 40                            tors 25 to 40.
:
0     BIN OK
>PL                                  and play this.
```

N.B.

1) The commands:

```
>ED[ 0]
```

and

```
>PL[ 0]
```

always mean "Edit (or Play) the latest BLOCK-file created"; they refer to "0 BIN" only when no definitions for composition-program BLOCKS have been made. However:

```
>FP 0 1
```

restores the file-pointer to its position at the beginning of the run: the next BLOCK-file to be created will be "1 BIN", and therefore the latest file is "0 BIN", which is thus referenced by >ED and >PL.

Similarly, "0 BIN" can be accessed by renaming it:

```
>RN 0 7777          "0 BIN" gets new name
                    "7777 BIN".
>PL 7777            play this
>ED 7777            and edit it.
:etc.
```

2) At the start of a run file "0 BIN" on DK <IMP> is transferred to the user-defined disk-area, destroying any file "0" that may previously have been on this disk. If, therefore, a version of "0 BIN" which has been edited during one run is to be used in the next run too, it must be renamed before the start of the second run. For example:

```
$PIP
>R DK 777 BIN_DK 0 BIN
>etc.
```

>WR (WRITE) is used to inspect BLOCK-files, including default-file "0 BIN". See also §4.2.8.

5.9.2 Some of the operations performed in real-time can be saved for use in future runs with:

```
>PT                 ( = PUT)
```

which creates a file "n IMP" on the user-defined disk area; the first such file created during a run is "1 IMP", the second is "2 IMP", and so on.

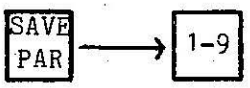
"IMP" files contain:

1) the AD-store, i.e. the nine versions of the AD-table which have been saved with:

or

```
>AD
:S n
```

2) the PAR-store, i.e. the nine versions of the Parameter-table which have been saved with:



3) the Joystick-store, i.e. the nine memory banks where Joystick movements and XX wave-structures have been stored with:

The numbering of these files can be directed with:

```
>FP 4 n
```

where n is a positive whole number specifying the name of the next "IMP" file to be created.

5.9.3 Files created with PT can be accessed with:

>GT n (= GET)

where n is a positive whole number stating which "IMP" file is to be read in. The previous contents of the AD, PAR, and Joystick stores are destroyed and replaced with the information in the file.

If there is no file called "n IMP" on the user-defined disk, the program writes:

n IMP NOT FOUND

and control reverts to the program monitor.

5.10 Combining real-time facilities with the composition program

BLOCKS created with the composition-program can always be modified in real-time by the operations described in this chapter. Such modifications are not permanent: but each time a BLOCK is played, the same real-time control must be exercised if the musical result is to be the same.

5.10.1 Suppose that the following structure is to be realized:

Fixed Values		Dynamic Values	
FG 1 40	GL 1020	ND	keyboard: 100 - 5000
MI 400	MF 667	A1	file: 280 - 330
F1 110	F2 880	A2	file: 300 - 350
AT 5	MS 500	DE	digitizer X: 0 - 100
WF 0	RV 10 360	VS	random: 400 - 600
duration: 2 minutes		VD	random: 8 - 12

Note that both VS and VD are to be set at random; suppose, however, that they are to be controlled by <u>different</u> random functions, so that they do not move in parallel. This can be done by connecting the dummy parameter XX to device 6 (random generator) and recording 1023 random numbers into one of the Joystick-store's memory banks. When music is played, the memory bank containing the random numbers is recalled with the JOY PLAY key, and connected to VD; VS is connected to device 6. The sequence of numbers controlling VD is repeated every 1023 samples, but since this takes at least 30 seconds when forty generators are being used, there is little danger of the composer's becoming aware of the repetitions.

A1 and A2 are to start with low values rising continuously throughout the BLOCK.

Wave-forms on frequency generators will be set by hand, since different values are to be used at the same time.

The following real-time switches are to be set:

RIGID parameter mode
EXPONENTIAL frequency distribution
RANDOM GENERATOR CONTROL: 0 (as fast as possible)
PITCH MODE: 8 (semitone scale)
PARAMETER RANGE: all parameters to full range

The "fixed" parameters GL, F1 and F2 can be set either with "SP" (set parameter) or with values in the file. Here they will be defined in the BLOCK-file.

This structure can be realized with the following sequence of instructions:

>CN	make suitable connections.
>RV 10 360	set reverberation,
>PL	and play the default file - not to make music, but to set real-time switches.
RIGID FLUID	this is FLUID at start of run, so press it to get RIGID-mode.
RAN GEN → 0	fast random generator.
PITCH MODE → 8	semitone-scale.
PAR RANGE → = -	put all parameters to full range
	EXPONENTIAL does not have to be pressed at the start of a run - this is the default mode.
XX → 6	connect XX to random generator.
JOY REC → XX → 1	record XX random numbers into Joystick memory-bank 1, and wait until the message "END STORE" is written.
JOY PLAY → 1	get the recorded values - this will still be valid the next time "PL" is done!
SPACE	end playing
STOPPED IN SEG 1	
>AD -1	put all parameter/device connections to zero.
:AD 5	now set connections and ranges for ND (device 5), DE (3), VS (6) and VD (1).
:ND 100 5000	
:AD 3	
:DE 0 100	
:AD 6	
:VS 400 600	
:AD 1	
:VD 8 12	
:	empty line to exit from "AD".

```
>SE 1                                  now create the BLOCK-file.
>DU 120000                             fixed values first.
>FG 1 24
>GL 1020
>WF 0
>F1 110
>F2 880
>LA 280                                and now the dynamic values A1 and
                                       A2.
>A1
:330 0
>HA 300
>A2
:350 0
>SP 3 400          (MI)                put fixed values on the parameters
>SP 4 667          (MF)                which are not controlled from file
>SP 10 5           (AT)
>SP 11 500         (MS)
                                       prepare to control ND with key-
>PL                                     board and DE with digitizer and
                                       then play the music.
etc.
```

5.10.2 Note the following points when combining real-time functions and BLOCK-files created with the composition program:

1) Make sure that any parameters which are to be controlled from a BLOCK-file are connected to zero in the AD-table.

2) All functions, switches and connections valid when PLAY is completed are also valid for the next PLAY. The only exceptions are FREEZE SOUND and ZERO DENSITY, which are <u>off</u> when PLAY begins.

3) The real-time and MIX facilities are completely incompatible. None of the operations performed in real-time is saved on a MIX-file, and indeed many parameters which can be controlled in real-time do not exist at all in MIX. The user who intends to work with MIX is therefore advised to write as follows at the beginning of the run; this ensures that when a BLOCK is listened to with "PL", it sounds exactly as it will when written on to a MIX-file. (See also §4.5.1.)

```
>SP 10                    put ATTACK TIME to zero.
AT     20
:0
>SP 14                    put VIBRATO SPEED to zero.
VS    100
:0
>SP 15                    likewise VIBRATO DEPTH
VD      5
:0
>etc.
```

6 Summary

6.1	Connections and other formalities	>CN[n]	CONNECT "n CON"		
		>RV n1 n2	REVERBERATION time & level		
		>CL	CLEAR STUDIO		
		>MT[-1]	MAGNETIC TAPE 0=on/-1=off		
		>CT	CLOSE TAPE (stop-mark)		
		>EX	EXIT from IMPAC		
		>IN	INITIALIZE		
6.2	BLOCK definition				
6.2.1	Structural parameters	>SE n	SEGMENTS	1	50
		>DU	DURATION input as for dynamic parameters	10	131000
6.2.2	Dynamic parameters	>DE	DENSITY	0	2000
		>F1	FREQUENCY 1	0	15999
		>F2	FREQUENCY 2	0	15999
		>A1	AMPLITUDE 1	0	400
		>A2	AMPLITUDE 2	0	400
	Input format: 1) >DE n (positive) same value in all segments 2) >DE[0] (zero) individual values 3) >DE -n automatic		-1 Random RANDOM LIMITS: SHAPE LIMITS: -2 Sinus -3 Triangle -4 Sawtooth } MIN,MAX,PERIOD,SHAPE: -5 Square -6 Curve		
6.2.3	Curve-start parameters	>DS n	DENSITY START	0	2000
		>LF n	LOW FREQUENCY	0	15999
		>HF n	HIGH FREQUENCY	0	15999
		>LA n	LOW AMPLITUDE	0	400
		>HA n	HIGH AMPLITUDE	0	400
6.2.4	Static parameters	>FG n1 n2	FREQUENCY GENERATORS	1	40
		>WF n	WAVE-FORM	0	7
		>GL n	GLISSANDO	0	8192
		>ND n	NOTE-DURATION	10	131000
	Input format: 1) >WF n same value throughout 2) >WF -n automatic				
6.2.5	Non-file parameters	>SA n	SAMPLING TIME	1	65536
		>SC n	SCALE - submonitor - [EOT] completes input	1	9
6.2.6	BLOCK-table functions	>L[n]	LIST BLOCK-table or named parameter "n"	0	17
		>CS n	CHANGE SEGMENT [EOT] keeps old value	1	50

6.3 BLOCK-file manipulations

6.3.1 Playing and creating BLOCK-files

```
>PL[ n]    PLAY "n BIN"
>PP[ n]    PLAY PART of "n BIN"
    SEGMENTS:n1 n2
>F         FILE : create new file
>PF[ n]    PART FILE : create new file from
    SEGMENTS:n1 n2.    part of existing one
>RN n1 n2 RENAME : n1 old name/n2 new name
>WR[ n]    WRITE "n BIN"
>TM[ n]    TIME: write duration of "n BIN"
```

6.3.2 Internal EDITOR Submonitor

```
>ED[ n]    EDIT BLOCK-file "n BIN"
```

- Parameter commands

:DU n	multiplication factor	0	131071
:DE n	multiplication factor	0	131071
:F1 n	multiplication factor	0	8192
:F2 n	multiplication factor	0	8192
:A1 n	addition factor	-400	400
:A2 n	addition factor	-400	400
:FG n1 n2	new value	1	40
:WF n	new value	0	7
:GL n	new value	0	8192
:ND n	new value	10	131000

- Other operations

```
:       empty line = perform editing
:SE n   SEGMENT
:D      DELETE one segment
:G n1 n2  GET file "n1 BIN"
          n2 ≠ 0 : smooth transition
:AB     ABORT - ORIGINAL FILE KEPT
: EOT   ABORT         - " -
```

6.3.3 Horizontal mixing

```
>KP[n1 [n2]]KEEP "n1 BIN"
            n2 ≠ 0 : smooth transition
```

6.3.4 Vertical mixing

```
>MX[n]     MIX "n BIN"
    START TIME:     floating-point number
>MS[n]     MOVING-SOUND: mix "n MVS" to inter-
           nal mix-file, or create new MVS file
    START TIME:
    MS#,DIRECT LEV,REVERB LEV:
    DURATION,RADIUS,SHAPE,ANGLE,SHAPE:
    End input with [EOT]
>PM        PLAY MIX
>SM        SAVE MIX
>DM        DELETE MIX
>MM n      MIX MIX-FILE "n MIX"
    START TIME:
>AP n      APPEND mix-file "n MIX"
    START TIME:
>TX        TIME OF MIX-FILE
```

6.3.5 Numbering of files

```
>FP n1 n2 FILE POINTER                  default
    n1 ≤ 0      BLOCK files              1 BIN
       = 1      KEEP files            1000 BIN
       = 2      MIX files                1 MIX
       = 3      MOVING-SOUND files       1 MVS
       ≥ 4      REAL-TIME STORE files    1 IMP
    n2 is number of next file to be created.
```

6.4 Monitor commands affecting real-time control

6.4.1 AD-table submonitor

- Parameter ranges

```
>AD n      ANALOGUE DEVICE connect to device n

    :ND n1 n2                                  10  131000
    :GL n1 n2                                   0    8192
    :MI n1 n2                                   0    4095
    :MF n1 n2                                   0    2000
    :F1 n1 n2                                   0   15999
    :F2 n1 n2                                   0   15999
    :A1 n1 n2                                   0     400
    :A2 n1 n2                                   0     400
    :DE n1 n2                                   0    2000
    :AT n1 n2                                   0  131000
    :MS n1 n2                  15     18 or 19      22
    :RV n1 n2                                   0     400
    :SI n1 n2                                   0    2880
    :VS n1 n2                                   0    2880
    :VD n1 n2                                   0    1000
```

- Other operations

```
    :L        LIST AD-table
    :S n      SAVE table to AD-store  1       9
    :G n      GET table from AD-store 1       9
    :AD n     ANALOGUE DEVICE
              1-9: connect to device and define
                   range
              0:   connect to BLOCK-file (no
                   range definition)
              -1:  put all parameter/device
                   connections to zero
              10:  alter ranges only (no con-
                   nections)
    :         empty line returns to program
              monitor
```

6.4.2 Fixed values on real-time parameters

```
>SP n1[ n2]SET PARAMETER "n1" with value "n2"

   1) ND  2) GL  3) MI  4) MF  5) F1  6) F2
   7) A1  8) A2  9) DE 10) AT 11) MS 12) RV
  13) SI 14) VS 15) VD 16) XX

   n2 must be legal value as in AD above
   MS: 0 to 1000          XX 0 to 511
   If n2 is undefined, negative or zero,
   current value printed - define new value
   or [EOT] to keep old one
```

6.4.3 Real-time store save and get

```
>PT       PUT current AD-store, PAR-store and
          Joystick-store on to a file "IMP".
>GT n     GET file "n IMP", created with PT,
          into real-time stores.
```

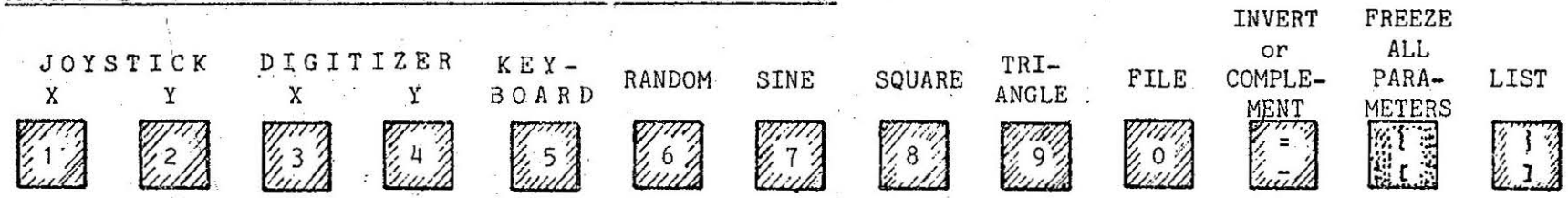
6.5 Memory inspect and modify

```
>RM n     READ MEMORY - 32 words, starting at
          octal address "n", are printed
          :    carriage return = continue
          :[EOT] = return to program monitor
>WM n1 n2 WRITE MEMORY - octal value n2 is put
          into word at octal address n1.
```

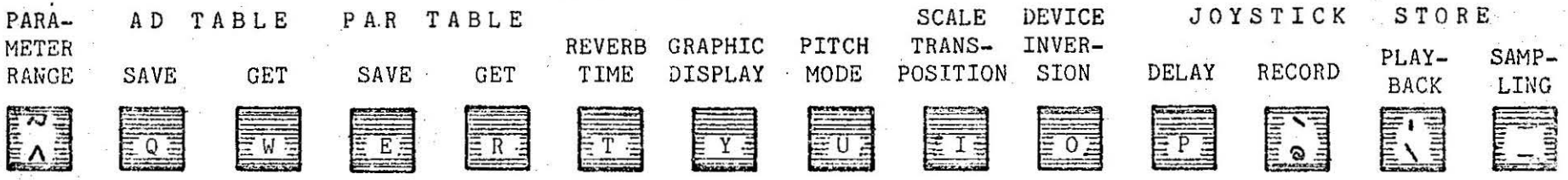
6.6 Multiple-key functions in real-time

Man	1-9	0	= -	}]	Comments
PAR	func	connect to device	connect to file	default & freeze	write current value	shift + PAR =(un)freeze
PAR RANGE	2	(n x 10)% range	100% range	all ranges 100%	write range	PAR key first
SAVE AD	2	save to AD store	all connections to zero	complement all connections	list AD table	
GET AD	2	get from AD store				
SAVE PAR	2	save to PAR-store	all parameters to minimum & freeze	all parameters to default & freeze	list PAR table	1-9 – shift PAR & AD tables together
GET PAR	2	get from PAR-store				
REV TIME	2	lowest of (n x 2) & 15	zero	n/a	write rev time	
GRAPH DISP	2	plot devices and parameters	off	n/a	list names of plotted devs & pars	followed by one or two keys
PITCH MODE	2	n^{th} scale	random	n/a	write scale no.	define with >SC
SCALE TRANS	2	transpose n $\frac{1}{4}$-tone steps	transpose 10 $\frac{1}{4}$-tone steps	all scales original position	write no. of transp. steps	
DEV INV	2	invert device n	all devs normal	all devs inverted	n/a	
JOY DEL	2	delay n samples	no delay	n/a	write delay	
JOY REC	2	record to JOY-store	stop recording or playing	n/a	write memory bank number	JOY REC → XX → DEVICE
JOY PLAY	2	get from JOY-store				
JOY SAMP	2	skip n values	no skip	n/a	write sampling rate	
RAN GEN	4	n x 10 samples	1 sample	n/a	write period	
SQU GEN	4	n x 20 samples	2 samples	n/a	write period	
TRI GEN	4	(n + 1) x 20 samples	20 samples	n/a	write period	

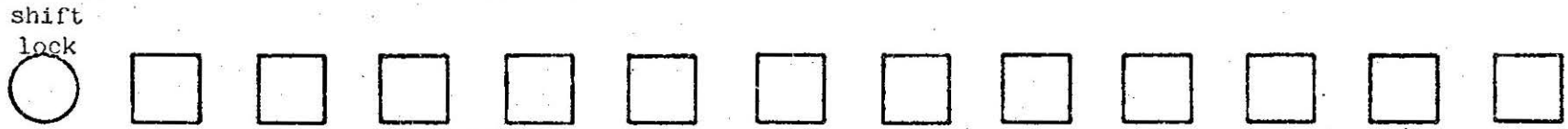
MANUAL 1: DEVICES and FUNCTION CONTROL



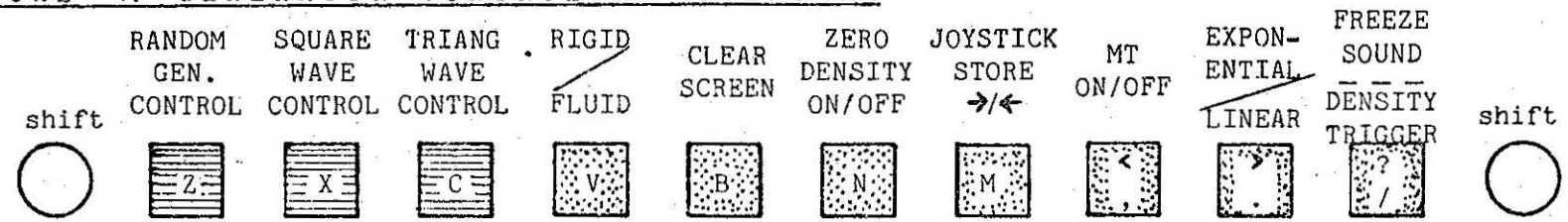
MANUAL 2: CONTROLLABLE FUNCTIONS



MANUAL 3: KEYBOARD (DEVICE 5)



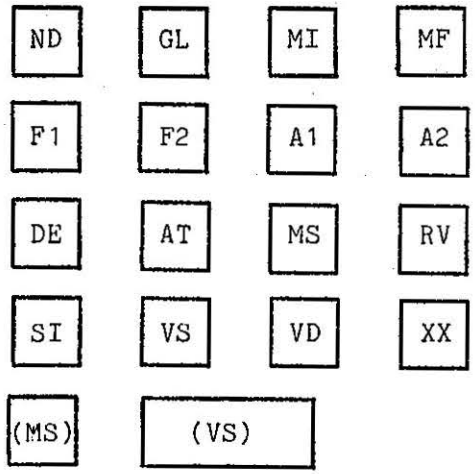
MANUAL 4: GENERATOR CONTROL and SWITCHES



STOP PLAYING

6.6 Multiple-key functions in real-time

Man	1-9	0	= -	] 1	Comments
PAR	func	connect to device	connect to file	default & freeze	write current value	shift + PAR =(un)freeze
PAR RANGE	2	(n x 10)% range	100% range	all ranges 100%	write range	PAR key first
SAVE AD	2	save to AD store	all connections to zero	complement all connections	list AD table	
GET AD	2	get from AD store				
SAVE PAR	2	save to PAR-store	all parameters to minimum & freeze	all parameters to default & freeze	list PAR table	1-9 + shift PAR & AD tables together
GET PAR	2	get from PAR-store				
REV TIME	2	lowest of (n x 2) & 15	zero	n/a	write rev time	
GRAPH DISP	2	plot devices and parameters	off	n/a	list names of plotted devs & pars	followed by one or two keys
PITCH MODE	2	n^{th} scale	random	n/a	write scale no.	define with >SC
SCALE TRANS	2	transpose n ½-tone steps	transpose 10 ½-tone steps	all scales original position	write no. of transp. steps	
DEV INV	2	invert device n	all devs normal	all devs inverted	n/a	
JOY DEL	2	delay n samples	no delay	n/a	write delay	
JOY REC	2	record to JOY-store	stop recording or playing	n/a	write memory bank number	JOY REC → XX → DEVICE
JOY PLAY	2	get from JOY-store				
JOY SAMP	2	skip n values	no skip	n/a	write sampling rate	
RAN GEN	4	n x 10 samples	1 sample	n/a	write period	
SQU GEN	4	n x 20 samples	2 samples	n/a	write period	
TRI GEN	4	(n + 1) x 20 samples	20 samples	n/a	write period	



6.9 Survey of IMPAC files

6.9.1 BIN files are those which contain the information for user-defined blocks.

A BIN file consists of any number of binary records, each containing eleven data words:

word	contents
1 - 2	a single-precision floating-point number defining the start time in seconds of this activity - e.g. 1.5 denotes that this activity is to start 1.5" after the beginning of the block.
3	activity number - 40, 41, or 48.
4 - 11	information - type depends on activity number.

time act information

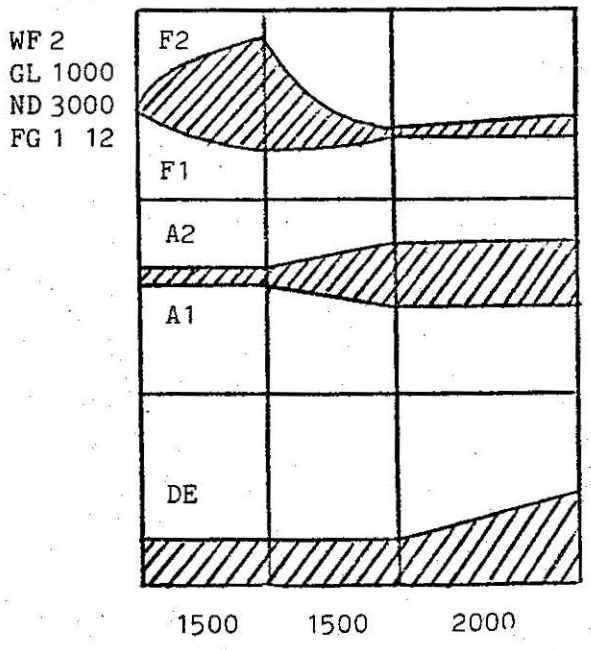
1 2	3	4	5	6	7	8	9	10	11	
TIME	48	LF	HF	LA	HA	DS	0	0	0	start values
TIME	41	F1 to shape		F2 to shape		A1 to shape		A2 to shape		curve information
TIME	40	DE to shape		DU	FG1	FG2	WF	GL	ND	parameters & music

Activities 41 and 48 provide information for activity 40.
Activity 40 causes music to be calculated and played in the studio. Each segment contains therefore one record of activity 40, optionally preceded by one record of activity 48 and/or one record of activity 41.

Within each segment, curves for primary parameters are calculated FROM either previous activity 41 or activity 48, TO current activity 41.

Here is an example of an IMPAC score and the instructions required to describe it:

```
>SE 3
>FG 1 12
>WF 2
>GL 1000
>ND 3000
>LF 400
>HF 440
>LA 300
>HA 320
>DS 10
>DU
:1500 1500 2000
>F1
:220 -1 296 -2 296 0
>F2
:700 2 315 -4 392 0
>A1
:300 0 280 0 280 0
>A2
:320 0 340 0 340 0
>DE
:10 0 10 0 20 0
```



The BIN file created when this is played looks like this:

0.0	48	400	440	300	320	10	0	0	0
0.0	41	220	-1	700	2	300	0	320	0
0.0	40	10	0	1500	1	12	2	1000	3000
1.5	41	296	-2	315	-4	280	0	340	0
1.5	40	10	0	1500	1	12	2	1000	3000
3.0	41	296	0	392	0	280	0	340	0
3.0	40	20	0	2000	1	12	2	1000	3000

N.B.
When BIN files are inspected with the WR command, the segment number is written at the beginning of each record; this is only for the convenience of the user - the segment number is not contained in the file.

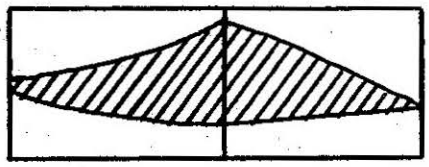
6.9.2 It is possible for the user proficient in FORTRAN to create IMPAC block files externally. The following points should be noted:
1) IMPAC reads BIN files with FORTRAN unformatted READ statements.

```
DIMENSION IARRAY(9)
...
READ(12,END=999)TIME,IARRAY
```

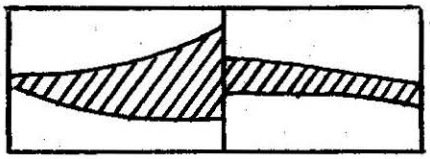
No checks are made to see if
a) the file is in the correct mode (alphanumeric, binary, etc.)
b) the records are of the correct length
c) the parameter values are legal.
If the records are not of exactly the type required by IMPAC, the results will be unpredictable.

2) Activity 48 is not restricted to the beginning of a block; it can be used whenever a primary-parameter curve is to start at a point different from the one reached at the end of the previous activity 40. For example:

Here we need activities 48, 41 and 40 for the first segment, but only 41 and 40 for the second, since it starts with the values reached at the end of the first.



In this example, both segments require activities 48, 41 and 40, as the starting-point of segment 2 is not the end point of segment 1.



3) When music is played with the commands PL or PP, the start-time given in each record is ignored; each segment follows on as soon as the previous segment has been played. Only in conjunction with the MX facility are start-times of importance.

4) The last record in a BIN file must be Activity 40 if the file is to function properly with MX and related commands.

6.9.3 CON files contain data for studio connections, disconnections, and amplifier intensities; they are executed with the command:

>CN n

where n is the number of the file to be performed. The five predefined CON files are discussed in §2.4.1.

A CON file consists of any number of binary records, each containing twenty-five data words.

word	contents
1 - 2	not used
3 - 25	function numbers (3, 4 or 5) and data; a function number indicates that all following data is of that type, until another function number is given.

function	data
3	amplifier intensities; format in each word is AAIII, where AA is amplifier number (as in EMSDEV and DIP1) and III is the intensity in $\frac{1}{4}$-dB. For example: 20400: 100 dB on channel 2. 1360: 90 dB on reverberation unit 1.
4	studio connections; format in each word is FFTT, where FF is the number of the device which is to be connected to TT. For example: 7172: FG3 connected to FG6. 219: reverberation unit 2 connected to channel 1.
5	studio disconnections; format and numbering as for connections.
<3	there is no more data in this record.

For example, the following record connects four groups of generators to channels 1, 2, 3 and 4 and reverberation units 1, 2, 3 and 4 respectively, and puts 100 dB on all of the channel output amplifiers:

0, 0, 3, 19400, 20400, 21400, 22400, 4, 7172, 7219, 7201, 7374, 7420, 7402, 7576, 7621, 7612, 7977, 8077, 7778, 8178, 8278, 8378, 7822, 7813

6.9.4 MIX files and MVS files consist of an unlimited number of binary records, each containing twenty-five data words defining one segment of music or moving-sound.

word	contents
1 - 2	single-precision floating-point number defining the start-time in seconds of this segment.
3	activity number: 0, 1, or 2.
4 - 25	data; type depends on activity number:

1) activity 0 and 1 - music segment

4: duration ms	5: FG 1	6: FG 2	7: not used
8: wave-form	9: glissando	10: note-length	11: F1 from
12: F1 to	13: F1 shape	14: F2 from	15: F2 to
16: F2 shape	17: A1 from	18: A1 to	19: A1 shape
20: A2 from	21: A2 to	22: A2 shape	23: DE from
24: DE to	25: DE shape		

act 0: this segment has user-defined start-values: DS LF HF LA HA.
act 1: no user-defined start-values.

2) activity 2 - moving-sound segment

```
 4: duration ms  5: MS  (1 or 2)
 6,7: reverberation level, real ( = 10.0 ** (REVLEV/40.0-10.0) )
 8: radius from  9: radius to   10: radius shape
11: angle from  12: angle to    13: angle shape
14,15: direct level, real ( = 10.0 ** (DIRLEV/40.0-10.0) )
16-25: not used
```

6.9.5 IMP files consist of one logical binary record containing 10,224 data words; they are written and read with FORTRAN unformatted I/O statements.

```
      INTEGER PARSTO,ADSTO,JOYSTO
      COMMON PARSTO(288),ADSTO(720),JOYSTO(9216)
      ....
C     PT - CREATE IMP FILE
C
      WRITE(12)PARSTO,ADSTO,JOYSTO
      ....
C     GT - READ IN IMP FILE
C
      READ(12)PARSTO,ADSTO,JOYSTO
      ....
```

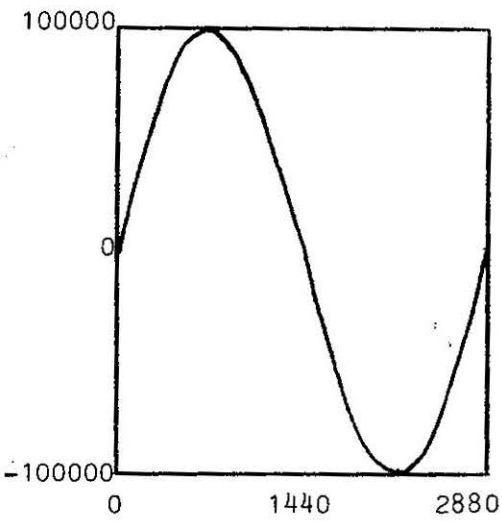
PARSTO has nine banks of thirty-two words each for storing the PAR table.

ADSTO has nine banks of eighty words each for storing the AD table.

JOYSTO has nine banks of 1024 words each for storing Joystick movements; the first word in each bank contains the number of coordinate pairs which have been recorded into the bank, minus 1023.

6.10 The sine-wave table

The real-time program's sine-wave generator and the vibrato function on frequency generators are calculated by looking up values in the sine-wave table. This consists of 2880 integers, between -100000 and 100000, which describe one period of a sinus oscillation.



6.10.1 The sine-wave generator is merely a pointer moving through this table, its speed governed by the parameter SI; the higher the value for SI, the faster the pointer moves.

6.10.2 For calculation of vibrato, there is one pointer for each of the frequency generators; these pointers move independently of one another, their speed governed by VS.

N.B.
1) If a pointer moves with a speed of zero or 2880 it remains, in effect, still - the oscillation comes to a halt.
2) Values between 1440 and 2880 are mirror images of those between zero and 1440; so a speed of 2879 is the same as a speed of 1.

6.11 Special uses of the studio registers

The studio registers for the noise generator and amplitude modulators are used for special purposes.

6.11.1 The noise-generator register shows the number of the most recently created BLOCK-file.

6.11.2 Amplitude-modulator 1 displays the sampling time in milliseconds. This is not the time defined by the user with "SA", but the actual time taken for calculations. See also §3.4.2.

6.11.3 Amplitude-modulator 2 displays the status of three switches:

1) on = RIGID
   off = FLUID
2) on = MT-switch on
   off = MT-switch off
3) on = exponential frequency distribution
   off = linear distribution