INTRODUCTION TO THE USE OF EMS-1

1. General

1.1 EMS-1 is a device orientated language, which means that the composers have to translate their musical conceptions into device terms. Other possible types of languages are for instance notation orientated languages, where the composer is building a bridge between his musical conceptions and a notation system; and a composition technical orientated language, where the bridge is built from the musical conception to rules of composition. The guide line in preparation of EMS-1 has been done on the basis of the principles of the console.

2. Devices

2.1 Names

In EMS-1 abbreviations for the name of devices are used. The rule of abbreviation is that if the device name consists of one word, the three first letters of the word are used, if it consists of several words, the first letter in each word is used.
Example: channel = CHA, frequency generator = FG.

Below is a complete list of devices:

AM	AMPLITUDE MODULATOR
AMP	AMPLIFIER
AT	ANALOGUE TAPE
CD	CHANNEL DISTRIBUTOR
CHA	CHANNEL
FF	FREQUENCY FILTER
FG	FREQUENCY GENERATOR
FS	FREQUENCY SHIFTER
NG	NOISE GENERATOR
REV	REVERBERATOR
RM	RING MODULATOR

2.2 The device parameters

The device parameters are written in the following way. The device name is followed by a left parenthesis, followed by the parameters separated by commas, followed by a right parenthesis. The parameters

in the following example are written with names, not with values:
FG(NUMBER,FREQUENCY,LEVEL,WAVEFORM).
The device name plus the device parameters are called a term.

In practice the value of the parameters is used and the same example would look:
FG(1,44Ø,1ØØ,2).

Example (all devices):

ABBREVIATION	COMPLETE EMS-I TERM
AM	AM(NO,ENTRY,LEVEL)
AMP	AMP(NO,LEVEL)
AT	AT(NO,LEVEL)
CD	CD(NO,CHANNEL,LEVEL)
CHA	CHA(NO,LEVEL)
FF	FF(NO,FILTERCHANNEL,LEVEL)
FG	FG(NO,FREQUENCY,LEVEL,WAVEFORM)
FS	FS
NG	NG(NO,LEVEL,COLOUR1,COLOUR2)
REV	REV(NO,REVERBERATION TIME,LEVEL)
RM	RM(NO,ENTRY,LEVEL)

Examples with the value of the parameters indicated:
FG(1,44Ø,1ØØ,2)
FF(2,2Ø,11Ø)
RM(1,A,1ØØ)
RM(1,B,1ØØ)
REV(2,6,1ØØ)
CHA(4,8Ø)

2.3 Abbreviations

Observe that not all parameters need be written out! In cases when some parameter is not written out, it keeps its value from the last time the parameter was set.

Example: FG(1,44Ø,1ØØ,2)
.
.
(new time indication)
.
.
.
FG(1,,2Ø)

The example shows that the frequency and the waveform are kept, but the level is changed from 100 to 20 dB.

2.4.1 Connections

The devices can be connected to device chains by using the connection sign

>

and the connection can be broken by the disconnection sign

#

2.4.2 Connection of devices and device chains

In the example A below frequency generator 1 is connected to output channel 1, so that we can listen to the tone of the frequency generator, after time has been set. In example B frequency generator 4 and frequency generator 2 are connected to ring modulator 1, which is connected to output channel 2, and we can listen to the result after time has been set. All device chains should be ended by a semi-colon.

A) FG(1,440,100,2)>CHA(1,80);

B) FG(4,300,100,2)>RM(1,A,100)>CHA(2,80);
FG(2,350,100,3)>RM(1,B);

C) FG(1,440,100,2)#CHA(1,80);

See appendix 1 for a list of all connections allowed.

2.4.3 Different devices can be collected in a group of devices by uniting the device terms with the sign

&

It is important to distinguish between a group of devices and a chain of devices and/or device groups. A group of devices are for instance a number of frequency generators or a number of filter channels. Below are some examples of device chains, made up by groups of devices and single devices:

FG(1,440,100,2)&FG(2,660,100,2)>RM(1,A,100)>CHA(1,80)&CHA(2,80);
NG(1,100)>FF(1,4,110)&FF(1,10,110)>RM(1,B);

A group of devices, connected to another group of devices or a single device, means that all devices in the first group are connected to all devices in the second group, or to the single device.

Example: FG(1)&FG(4)&FG9&REV(1);
FG(1,100)&FG(2,200)&FG(3,300)&FG(4,400)>CHA(1,100);

The last line means that the frequency generators 1, 2, 3, and 4 are given a frequency each and are then connected to channel 1, given the level 100 dB.

2.4.4 Groups of frequency generators and filter channels can be collected with the sign ←

A condition is that no device between the lowest and the highest number is to be excluded. The meaning of
FF(1,5←12,1ØØ)
is for instance that the channels 5 to 12 in the frequency filter no 1 shall be set to 100 dB.
Example: FG(1←6)
means that the frequency generators 1 to 6 shall be connected somewhere keeping their old values.
Example:
FG(1)&FG(7←12,,8Ø,2)&FF(1,1←3,1ØØ)>CHA(2)&CHA(3)&CHA(4,98);

This way of writing leads to frequency generator 1 and also no 2 and no 3, all keeping their old parameter values, and the frequency generators 7, 8, 9, 10, 11, and 12, all with their old frequencies, with the new level 80 dB and waveform 2 and filter 1, are connected to the channels 2, 3, and 4. Channel 1 and 2 in filter 1 are also set to 100 dB and output channel 4 to 98 dB.

Exercise 1:

Make one or several groups consisting of frequency generator groups 3, 6, 9, and 12 and the reverator 1. Connect what is possible to channel 1 via filter 1 or else directly to channel 1. Let the units keep the values they have been given earlier.

Exercise 2:

Make a suitable group, consisting of 5 or 6 frequency generators. Let these be amplitude modulated to 90% (=99dB) by the frequency 100 Hz in AM(2). Connect the resulting signal to channel 1. The A-entry of an amplitude modulator modulates the B-entry.

(solutions see page 6.)

3. Time

We make a distinction between TIME and DURATION so that TIME says when something occurs and DURATION (DUR) indicates the duration of the occasion, for instance the duration of a tone.

3.1 General

There are three different ways of describing duration on the console. First, the order TIDMÄTA (measure time), which means that the studio is made to sound during the time the TIDMÄTA-button is pressed, i e a duration.

The other way is the order LYSSNA KONTINUERLIGT (listen continuously), meaning that the studio is made to sound when the button for the order LYSSNA KONTINUERLIGT is pressed, and is made to stop sounding when the button for the order STOPPA KONTINUERLIGT LYSSNANDE (stop listening continuously) is pressed.

The third way to describe time at the console is the order TIDLYSSNA (time listen), functioning so that the time desired for the studio to sound is set in the index register, whereupon the button for the order TIDLYSSNA is pressed. The studio then sounds during the time set in the index register.

Time- and duration descriptions in EMS-1 do not coincide with the orders at the console. At the console the composer orientates himself in the music structure, or rather the digital tape, via record numbers. In EMS-1 the composer orientates himself in the music structure via different time- and duration conceptions.

3.2 Global time

The expression Global Time (GT) refers to the running composition time. For a piece of 10 minutes' length Global time zero stands for the beginning of the piece, Global time 5 minutes for the middle of the piece, and Global time 10 minutes for the end of the piece.

In EMS-1 beginning time and end time, expressed in Global time, is automatically written on the commentary medium after each completed block or PART - see point 7.18 and the examples on page . The actual GT can be calculated by taking the time of the end of the preceeding block and add to it the beginning time for the sound currently being worked on plus the current Local time, i e where the composer is at the time being.

3.3 Local time

The time between each single tone has on the other hand its counterpart in EMS-1 and is called Local time, in short LT(MS) or LT(SEC,MS). The millisecond parameter must not contain a number greater than 99 999, and the second parameter not a number greater than 8 333.

The LT-term can more correctly be said to indicate when the connection settings and device settings which follows before the next LT-term are to be performed.

Examples of LT-terms:

```
LT(0)  FG(1,440,100,2)>CHA(1,80);
LT(2000)FG(1,660,100,3)>CHA(1,80);
LT(52,230)
LT(52,0)
LT(52,)
```

It is no demand that LT-terms come in time order.

```
LT(900)FG(1,200)
LT(0)FG(1,100,100)>CHA(1,100);
```

How to solve exercise 1:

```
FG3>FG6;
FG9>FG12;
FG6&FG12>FF(1);
FF(1)&REV(1)>CHA(1);
```

How to solve exercise 2:

```
FG15>FG18>AM(2,B,100)>CHA(1,100);
FG(24,100,99)>AM(2,A);
```

3.4 Duration

While LT is a time where something is going to happen, for instance a frequency generator shall start to sound, the term duration, shortened as D(MS), indicates how long a certain sound level is to last.

Example:

```
LT(20,200)FG(1,440,100,2)>CHA(1,80)>D(2000);
```

Exercise 3:

FG(1) and FG(4) are set, but not connected. FG(1) shall start sounding after 31 seconds and 120 ms, counted from the beginning of the block, and sound during 23168 ms. FG(4) shall start sounding after 23 seconds and sound during 20000 ms. Both sound in channel 3.

Write the part of the program that handles this. (Solution page 8.)

While the term LT is valid for all devices until the next LT, the term D is valid only for the device or group of devices that it is connected to.

The use of a new LT-term is made either to introduce changes in the device parameters - a new tone, a new intensity - or especially to introduce 0-values into the level parameter, which means that the device from now on will no longer sound. (About the D-term in connection with envelopes and glissandi, read point 4.4 and 4.5.)

3.5 TIDMÄTA (time measure) is a term which does not exist in EMS-I, but only at the console.

3.6 TIDLYSSNA (time listen) exists also only at the console. A close counter-part is however the order PLAY (see point 7.8).

3.7 LYSSNA KONTINUERLIGT (listen continuously) exists only at the console, but can naturally also be reached via EMS-I by the order PLAY, under the condition that what the composer wants to listen to has already been given a very long duration.

4. Envelopes

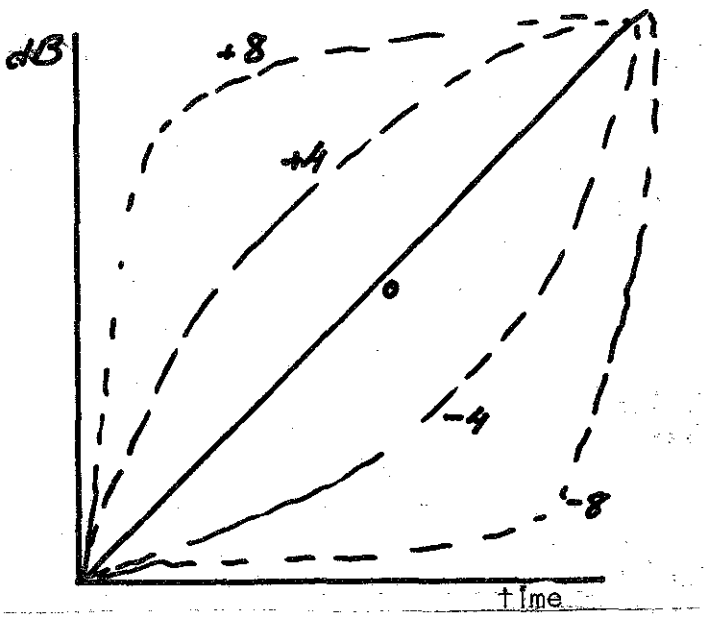
4.1 The envelope, i.e. the changes of intensity, at the console, is planned to be performed with the help of attenuators. The counter-part in EMS-I is the term ENVELOP, in short ENV.

4.2 The complete term with parameters is written:
ENV(STARTLEVEL,ENDLEVEL,DUR,CURVETYPE,TIMESTEP)
meaning that a level can be changed to an end level during a specified duration in milliseconds, and following a decided curveshape with a certain time-step between the changes. The time-step is the difference in ms between two on each other following changes. The time-step must not be greater than the envelope-time.

4.3 Curveshapes
We have 19 different curveshapes, built up in three classes.
The first class consists of one unit, a linear change from one value to another value.
The second class consists of 9 units, all convex, and the third class

of 9 units, all concave. The linear class is in EMS-I described by the number Ø. The 9 convex units are described by the numbers I-9, where I is the least convex and 9 the most convex. The 9 concave units are described by the number -I to -9, where -I is the least and -9 the most concave.

In a case where the intensity changes from lower to stronger, convex can also be described as a crescendo, where the greatest change of intensity takes place in the beginning of the tone. In a case where the intensity changes from a higher value to a lower, the choice of a convex curve means that the greatest intensity change will come at the end. In the concave cases the conditions are contrary.
The curves can graphically be illustrated in the following way:



How to solve exercise 3:

LT(3I,I2Ø)
FG(I)>D(23I68)>CHA(3,IØØ);
LT(23,)
FG(4)>D(20000)>CHA(3,IØØ);

Commentary: The LT-terms need not come in time order. Observe that the D-term must come before the channel-term to reach the effect desirec. If we had written:
FG(I)>CHA(3,IØØ)>D(23I68):
resp.
FG(4)>CHA(3,IØØ)>D(20000);
and had the same starting-times as above, it would have meant that

both sounds had stopped sounding after 23 seconds + 20000 ms (= 43 s.) instead of stopping at the times 54 s 288 ms resp 43 s.

4.4 In time respect the envelope term is a special case inasmuch as the time is included as one of the parameters. Because of this, two interpretations were possible during the design of the language:

1) shall a tone, connected to an envelope, and thereby given an intensity going from one level to another, continue to sound after the end level has been reached, or
2) shall it be considered that when the end level has been reached, this tone shall stop sounding?

In EMS-1 we have chosen the solution to consider that the tone continues to sound and that the composer must give an order if he wants it not to sound after reaching the end level; this order is

Z

If the composer wants the tone to sound only during the time it takes to make the envelope, write as follows:

ENV(Ø,1ØØ,5ØØ,2,2Ø)>Z;

meaning that the tone to which the envelope is connected, starts with the intensity of 0 dB, grows to 100 dB during a duration of 500 ms, following curvetype 2, whereupon the level returns to 0 dB, and the tone consequently is no longer heard. The number 20 indicates the time-step between the changes (the sampling speed).

4.5 The D-term can also be applied to an envelope term so that the sound-source sounds during the time specified in the envelope term plus the time in the D-term, to which the envelope term is connected.

Example: ENV(Ø,1ØØ,5ØØ,2,2Ø)>D(1ØØØ);

meaning, as in the previous example, that the intensity grows from 0 to 100 dB during a time of 500 ms, but that in this case it continues to sound at the end-level during another 1000 ms, i.e. the time of the D-term.

4.6 A chain of envelopes can be created by using the connection sign between the envelope terms.

Example: ENV(Ø,1ØØ,5ØØ,2Ø)>ENV(1ØØ,1ØØ,1Ø)>ENV(1ØØ,Ø,5ØØ,2,1Ø)

4.7 Envelope terms and envelope chains can be connected to devices or groups of devices by connection sign.

Example:

```
LT(Ø)FG(1,1ØØØ,1ØØ,2)>ENV(Ø,1ØØ,5ØØ,2,1Ø)>ENV(1ØØ,1ØØ,1ØØØ)>
                      ENV(1ØØ,5Ø,5ØØ,2,1Ø)>Z;
                      FG(1)>CHA(1,1ØØ)&CHA(2,1ØØ);
```

The term ENV(1ØØ,1ØØ,1ØØØ) is a special case of the envelope term, where the starting value and the end value are alike.

4.8 ESTEP and GSTEP

The composer can specify how often the computer shall set a new value during the generation of envelopes and glissandi. If no value is indicated the computer will set a new value every ten ms. If the composer wants the computer to set another value, more frequently or less, this is done with the term ESTEP(MS) regarding the envelope, and the term GSTEP(MS) regarding glissandi.

Example: ESTEP(1)

meaning that the computer will set a new level-value every ms;

GSTEP(2Ø)

meaning that the computer will set a new frequency value every 20 ms.

If the composer wishes to change the time-step only during one envelope term, the desired time-step is written as a parameter in the envelope term. Thereafter, the value set with ESTEP resp. GSTEP is valid again. For technical reasons an envelope or a glissando must not contain more than 2047 time-steps.

5. Glissando

5.1 The parameters of the glissando term

Glissando is written with a term much alike the envelope term, i.e. GLIS, followed by starting frequency, end frequency, duration in ms, curveform, and time-step.

Example:

```
GLIS(FREQUENCY,FREQUENCY,DUR,CURVEFORM,GSTEP)     (Regarding GSTEP see 5.3
GLIS(44Ø,8ØØ,1ØØØØ,2,5)
```

5.2 Curveshapes

The curveshapes of the glissando term are the same as those of the envelope term.

5.3 GSTEP

GSTEP is the counter-part for the glissando term of ESTEP for the envelope term, i.e. to indicate in ms how often the computer shall give a new value - make a sample. If the values are too sparsely set, there is a risk to get a scale instead of a glissando. If the values are too densely set, unnecessary information is used.

5.4 An envelope term is related to the closest preceeding device term. The starting time in an envelope term is the latest indicated local time or in the case of envelope chains, the end time of the closest preceeding envelope term (if no device term stands between, parting the envelope chain into two chains). The same is valid for the GLIS-term.

Observe that envelope times and glissando times ignore the existence of one another, and consequently can be woven into each other. Envelope terms and envelope chains can be mixed with glissando terms and glissando chains.

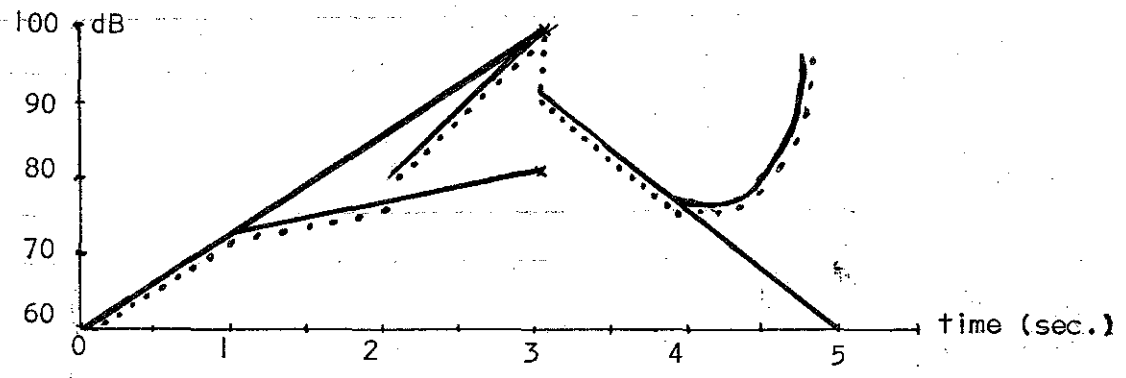
Example:

```
FG(1<-3)&FG(7<-9,,,2)>
ENV(50,100,500)>GLIS(220,440,200,2)>
GLIS(440,880,1000)>ENV(100,90,1200,1,10);
CHA(1,100)>ENV(80,100,2000,1)>D(10000);
```

5.5 At overlapping envelopes the last defined is valid.

Example:

We have given a frequency generator level the following envelopes:



```
LT(Ø,Ø)FG(1)>ENV(6Ø,1ØØ,3ØØØ);
LT(1,Ø)FG(1)>ENV(73,80,2ØØØ);
LT(2,Ø)FG(1)>ENV(8Ø,1ØØ,1ØØØ);
LT(3,Ø)FG(1)>ENV(75,1ØØ,1ØØØ,-3);
```

The resulting envelope is dotted in the figure on page 11.

6. The connection matrix of the frequency generators

At the console, the 24 frequency generators have their own connection matrix. The connection matrix is designed to have 24 entries, the same number as the number of frequency generators, but only 8 exits. The frequency generators are thus connected in groups of three so that the frequency generators 1, 2, and 3 have one exit called FG3, 4, 5, and 6 have exit FG6, etc. These 8 connection points are called FG3, FG6, FG9, FG12, FG15, FG18, FG21, and FG24.

6.1 Creation of a sound

You want a sound consisting of a basic tone with a frequency of 200 Hz, an intensity of 70 dB, and with harmonic overtones, being 1.5, 2 resp. 3 times higher than the basic tone, each with a 3 dB lower intensity than the closest preceeding tone. Each tone shall be fed into a channel of its own (each corner of the listening room). The waveshape shall be sine, i.e. number 0 or number 1. The sound is to sound during 30 seconds. How is this achieved with EMS-1?

Solution:

```
1    PART(ONE)
2    LT(Ø)
3    FG(1,2ØØ,7Ø,Ø)>CHA(1,1ØØ);
4    FG(4,3ØØ,67,Ø)>CHA(2,1ØØ);
5    FG(7,4ØØ,64,Ø)>CHA(3,1ØØ);
6    FG(1Ø,6ØØ,61,Ø)>CHA(4,1ØØ)>
7    D(30000);
8    PLAY
```

Commentary to the solution:

Line 1. For technical reasons all EMS1-programs consist of blocks. One block contains at the most about 14.000 changes. The composer must indicate where the block starts and where it ends. The beginning

of a block is indicated by the word PART(NUMBER) and the end with END. Each PART must have a name. This is called (ONE).

Line 2. LT(Ø) - Local time - tells when in the sound object something is to happen. At the time 0 the nearest following lines (3-7) are executed.

Line 3. Frequency generator 1 is here set to the frequency 200 Hz, the level 70 dB, and sine wave. The frequency generators 1, 2, and 3, which are always connected, are further connected to output channel 1 and its out-level is set to 100 dB, meaning that you will not have any relative amplification, i.e. the sound level at listening will be 70 dB.

Lines 4-6. To be able to connect the frequency generators further to different places they must be chosen so that they are not permanently connected to each other.

Line 7. D(3ØØØØ) is an envelope term. It is connected to channel 4, which thereby is made to keep its level during 30 seconds (30000 ms) and then stop. As no other time specification exists, everything stops after 30 seconds.

Line 8. To test-listen to the result, use the order PLAY. Technically the order results in the object being sorted, i.e. all LT-terms are arranged in the right order - translated to EMS-code, recorded on the magnetic tape, and re-created in the studio.

6.2 To erase errors the principle in EMS-1 is to rewrite. If you have written for example:

LT(21ØØ) FG(4,44Ø,92)>CHA(1,1ØØ);
LT(54ØØ) FG(6,356,7Ø,3)>CHA(4);

and want to change 92 into 80:

First, find out after which LT-term the fault is, then in which device, and finally in which parameter.

If you are writing a text, the LT-term with the error is found by writing LT(21ØØ). Consequently, it is possible to correct errors anytime anywhere in the existing text. After having found the LT-term, the device is found by writing its name - in this case the frequency generator no 4 - followed by the parameters of the device, including the corrected parameter. Write:

LT(21ØØ)FG(4,44Ø,8Ø) or LT(21ØØ)FG(4,,8Ø)

and the error is corrected.

6.3 CLEAR

At the console the order NOLLSTÄLL (set to zero) can be used to make sure that no old settings are left when the creation of a new sound is starting. The corresponding command in EMS-I is CLEAR.

SUMMING UP OF ELEMENTS IN EMS-I

Device terms:

AM(NO,ENTRY,LEVEL)
CD(NO,CHANNEL,LEVEL)
CHA(NO,LEVEL)
FF(NO,FILTERCHANNEL,LEVEL)
FG(NO,FREQUENCY,LEVEL,WAVESHAPE)
NG(NO,LEVEL,COLOUR1,COLOUR2)
RD(NO,CHANNEL,LEVEL)
REV(NO,REVTIM,LEVEL)
RM(NO,ENTRY,LEVEL)

Studio points:

FS
FG3,FG6,FG9,FG12,FG15,FG18,FG21,FG24

Commands:

CLEAR

TEMP-commands:

PLAY

Local time:

LT(MS),LT(SEC,MS) Observe that LT(1,5) means the same as LT(1,ØØ5), not LT(1,5ØØ).

Envelope terms:

ENV(LEVEL,LEVEL,MS,CURVESHAPE,STEP)
Z
D(MS)

Glissando terms:

GLIS(FREQUENCY,FREQUENCY,CURVESHAPE,STEP)

Step terms:

ESTEP(MS)

GSTEP(MS)

EXIT stops the run and starts EMS-I for a new program.

>, #, & stand between elements in connection chains, envelope chains, and glissando chains. If the chain continues over several lines, it shall be divided so that one of these signs is the last on the line.

; ought to end each chain.

← can be used in FG- and FF-terms.

'COMMENTARY' shall be used relatively often. Otherwise you will very soon forget what a program is about.

Examples of error printouts

In the following there are some examples of error printouts. The first example is commented here.

\ erases the preceeding sign. Commentaries can be inserted between single apostrophes ('...'). It is a good habit to write in the commentary what the program is about as a help for the memory.

```
:PART(ONE)
:LT(Ø)
:FG(1,23Ø#2Ø)>CHA(1,1ØØ);
 FG(1,23Ø#)
Ø2  Ø11       ILLEGAL DELIMITER
!FG(1,23Ø,2Ø)
!
:'ERROR NOW CORRECTED'
```

The lines 1 and 2 are correct, but in the third line there is an error. The composer is told so by the computer after he has written the third line and made a carriage return. The error printout of the computer shows the composer both the symbol where the error is, and the symbol which thereafter is correct, and with which the composer shall end the corrections.
On a new line the code for the error is written with two numbers and the description of the error 'ILLEGAL DELIMITER'. The error in this case is that the symbol for disconnection has been written as a punctuation mark in a frequency generator term. The composer corrects the error by rewriting the full FG-term, when the computer has told him that it is ready to receive corrections by writing an exclamation sign in the beginning of a new line. After the correction the composer makes carriage return, and the computer writes a colon in the beginning of the following line, thereby informing that it is ready to receive more EMS-1 text, which in this case is placed before the term CHA(1,1ØØ).

```
                    EMS1 V1.3
                    NAME OF INPUTFILE?

                  INTERACTIVE MODE? YES OR NO?                    F2
YES


:PART(FELU)
:LT(0)
:FG(1,100#80,5)
 FG(1,100#     )
02   011          ILLEGAL DELIMITER
!FG(1,100,80,5)
!
:FG(7,A,75)
 FG(7,A,   )
02   012          ILLEGAL PARAMETER
!FG(7,300,75)
!
:'THE SYMBOL A IS RESERVED TO DENOTE AN ENTRY IN AM- OR RMTERMS'
:

                     INTERACTIVE MODE? YES OR NO?
YES

:FG(1,2,3,4,5)
 FG(          )
01   020          PART COMMAND FIRST THING IN A BLOCK!
!
:PART(2)
:FG(1,2,3,4,5)
 FG(1,2,3,4, )
02   014          ILLEGAL NUMBER OF PARAMETERS
!FG(1,2,3,4)
!
:FG(25,100)
 FG(25,   )
02   121          ILLEGAL DEVICE NUMBER
!FG(24,100)
!
:FG(16,16000)
 FG(16,16000)
02   022          ILLEGAL FREQUENCY GENERATOR FREQUENCY
!FG(16,15999(
 FG(16,15999(
02   011          ILLEGAL DELIMITER
!FG(16,15999)
!
:FG(8,,,8)
 FG(8,,,8)
02   023          ILLEGAL FREQUENCY GENERATOR WAVEFORM
!FG(8,,,7)
!
:FG(12,,120.25)
 FG(12,,120.25)
02   024          ILLEGAL FREQUENCY GENERATOR INTENSITY
!
:FG(12,,120)
```

```
:FF(1,29,121)
 FF(1,29,   )
02     121            ILLEGAL DEVICE NUMBER
!FF(1,28,121)
 FF(1,28,121)
02     033            ILLEGAL FREQUENCY FILTER INTENSITY
!FF(1,28,120)
!
:AMP(3)
 AMP(3)
02     121            ILLEGAL DEVICE NUMBER
!AMP(2)
!
:FG(24)>CHA(1);
               ;
02     051          ILLEGAL (DIS-) CONNECTION
!FG(24)>FG24>CHA(1);
!
:REV(2,16,90)
 REV(2,16,   )
02     062            ILLEGAL REVERBATION TIME
!REV(2,15,90)
!
:REV(2,0,130)
 REV(2,0,130)
02     063            ILLEGAL REVERBATION INTENSITY
!REV(2,0,120)
!
:D()
 D()
02     070            TIME MISSING IN D-TERM
!D(100)
!
:D(0)
 D(0)
02     071            ILLEGAL TIME
!D(1)
 D(1)
02     245          ENV OR GLISTIME LESS THAN STEP
!ESTEP(1)
!D(1)
!
:NG(1,90,BLUE)
 NG(1,90,BLUE
04     020            SYMBOL NOT DEFINED
!NG(1,90,A)
 NG(1,90,A)
02     081            ILLEGAL NOISE COLOUR
!NG(1,90,PINK)
!
:NG(1,150)
 NG(1,150)
02     082            ILLEGAL NOISE INTENSITY
!NG(1,120)
!
:AM(3,B,100)
 AM(3,      )
02     121            ILLEGAL DEVICE NUMBER
!AM(2,PINK)
 AM(2,PINK)
02     092            ILLEGAL AMPLITUDE MODULATOR ENTRY
!AM(2,B,170)
 AM(2,B,170)
02     093            ILLEGAL AMPLITUD MODULATOR INTENSITY
!
```

```
:RM(1,PINK,90)
 RM(1,PINK,
02    102            ILLEGAL RING MODULATOR ENTRY
!RM(1,B,140)
 RM(1,B,140)
02    103            ILLEGAL RING MODULATOR INTENSITY
!RM(1,B,120)
!
:CD(1,4,150)
 CD(1,4,150)
02    112            ILLEGAL CHANNEL DISTRIBUTOR INTENSITY
!
:AT(3,-10)
 AT(3,-10)
02    150           ILLEGAL ANALOG TAPE INTENSITY
!AT(3,10)
!
:CHA(1,1000)
 CHA(1,1000)
02    142            ILLEGAL CHANNEL INTENSITY
!CHA(1,100)
!
:ESTEP(0)
 ESTEP(0)
02    201           ILLEGAL ESTEP VALUE
!ESTEP(10)
!
:FG(1)>ENV(100,120);
       >ENV(100,120);
02    241           ILLEGAL ENV SYNTAX
!FG(1)>ENV(100,80,1000)
!
:>ENV(100,130,1000)
 >ENV(100,130,     )
02    242            ILLEGAL ENV AMPLITUDE
!>ENV(100,120,1000)
!
:>ENV(80,110,0)
 >ENV(80,110,0)
02    245            ENV OR GLISTIME LESS THAN STEP
!
:>ENV(110,106,10,10)
 >ENV(110,106,10,10)
02    247            ILLEGAL ENVELOPE TYPE
!>ENV(110,106,10,9)
!
:ENV(80,96,1000)
 ENV(            )
02    251            NO DEVICE TO ENV GIVEN
!
:LT(-5,170)
 LT(-5,    )
02    261            ILLEGAL LOCAL TIME VALUE
!LT(5,170)
!
:GSTEP(-4)
 GSTEP(-4)
02    271           ILLEGAL GSTEP VALUE
!GSTEP(4)
!
:FF(1,28)>GLIS(440,880,1000);
          >GLIS(              )
02    301            NO DEVICE TO GLISSANDO GIVEN
!FG(1,,82)>GLIS(440,880,1000);
!
```

```
:FG(7)>GLIS(1);
      >GLIS(1);
02    302          ILLEGAL GLISSYNTAX
!FG(7)>GLIS(100,300,2100)
!
:FG(8)>GLIS(300,100,0)
      >GLIS(300,100,0)
02    245          ENV OR GLISTIME LESS THAN STEP
!>GLIS(300,100,300,-10)
 >GLIS(300,100,300,-10)
02    304          ILLEGAL GLIS TYPE
!>GLIS(300,100,300,-9)
!
:>GLIS(0,16000,3000,-4)
 >GLIS(0,16000,        )
02    305          ILLEGAL GLIS FREQUENCY
!>GLIS(0,15999,3000,-4)
!
```

7. Working procedures

7.1 General

In the following one of several possible working procedures is described. It is based on the assumption that a composer creates his music structure by letting, as a basis, a group of single tones compose a sound-object, and that he further creates his structure with the help of such sound-objects. The single tones we call sounds, a group of sounds an object, and a group of objects a PART.

With this technique the composer can build up a structure in the form of a series of objects and PARTS. He can also work in two steps, i.e. first build up a conditional series of objects, and, later on, place these objects in the right order with the help of their names.

7.2 TEMP, ACC, and AUX

The console works in such a way that a sound is stored in a register, which in its turn is transferred to and stored on a digital tape. During PLAY the information passes a buffer memory, before it is restored in the register, and recreates the sound. What in EMS-2 best corresponds to the register of the console, is called TEMP, ACC, and AUX. EMS-2 has, in other words, three different storages where the console has one. This can be used so that a sound is built up in TEMP and transferred to ACC (the command is MIX or APP). A new sound can be built up in TEMP, and the two sounds be mixed and brought back to ACC. The composer also has access to a test-mixing, where he can listen to the result and maybe change the latest sound. This differs from the final mixing, where the result cannot be divided again. He must, of course, make sure that the object is mixable, i.e. that one object does not use a device, which is also used in the other one. When one PART is completed, you write END, and the contents in ACC is transferred to a magnetic tape.

7.3 Transmission of a sound in TEMP to ACC

A sound in TEMP is transferred to ACC via the command MIX(START). Technically, the command causes a sorting of the sound in TEMP, i.e. the sound is transferred to ACC. TEMP is prepared to have a new sound put in. (START) indicates the time reference for the sound in TEMP, when TEMP is sorted together with a possible previous content in ACC. When the sounds in TEMP and ACC are mixed, they consequently need not start at the same time. ACC always starts at LT(Ø), while TEMP for instance can be shifted 20 ms by writing MIX(2Ø).

7.4 Test-mixing

When the composer wants to test-listen to a mixing he uses the command TRY(START). If he is satisfied with the result and wants to make a definite mixing, he uses MIX(START). Otherwise, he can make TRY with another starting time or use CLEAR.

7.5 CLEAR and CLEAR(MIX)

The command CLEAR causes an erasure in TEMP of the old settings. In the same way ACC can be erased by the order CLEAR(MIX). Both TEMP and ACC can be erased at the same time with the order CLEAR(ALL). However, CLEAR does not change any settings already existing in ACC, nor CLEAR(MIX) anything remaining from a previous block. At MIX, CLEAR occurs too. At END, CLEAR(MIX) occurs.
Observe that PART(NUMBER) is not erased, being no setting.

7.6 The setting to zero of TEMP at MIX can be neutralized if KEEP follows immediately. The same is valid for ACC if KEEP(MIX) follows immediately after END. This can be explained by saying that after a CLEAR you start again in TEMP and write over the lines, which used to be there, but after KEEP you continue to write where you ended. If, for instance, you want to sort in the same object at different times, you can write MIX(time 1); KEEP (to regain the object); MIX(time 2), etc.

7.7 The console order SPELA (play) is in EMS-2 equivalent to PLAY, PLAY(MIX), PLAY(ALL), PLAY(ONE), and PLAY(ONE,TWO).

7.8 The command PLAY means that the sound built up in TEMP is played.

7.9 The command PLAY(MIX) means that the sound stored in ACC is played.

7.10 The command PLAY(ALL) means that all ready PARTS are played. Thereafter ACC is played.

7.11 The command PLAY(PART 1, PART 2) means that the part of the composition, which starts with PART 1 and ends with PART 2, is played. PLAY(PART 1) means that the part PART 1 is played.

7.12 If you want to see the sound built up in TEMP as a sound-object with its own object-name, you give the sound this object-name by the order SAVE(NAME), for instance SAVE(ADAM). The corresponding text is then stored on the DEC-tape.

7.13 If, instead of creating a new sound in TEMP, you want to recall a previous sound, which has been given an object-name and thus been transformed into an object, use the command CALL(NAME), for instance CALL(ADAM). If you, furthermore, want the text of the object ADAM written out on the commentary medium, this is made by TOP(NAME) followed by LOOK(NUMBER), where NUMBER is an integer, the number of lines you want to have a look at.

7.14 An object is added to an object by the command APP, meaning that the object stored in TEMP is appended after the object stored in ACC. If, for instance, you have first called the object ADAM3 to TEMP and then stored ADAM3 in ACC with the order MIX(START), you can let ADAM3 be followed by the object ADAM4 by calling ADAM4 to TEMP, followed by the command APP:

TOP(ADAM3)	TOP(ADAM4)
LOOK(3Ø)	LOOK(5Ø)
CALL(ADAM3)	CALL(ADAM4)
MIX(Ø)	APP

7.15 If you are uncertain if it sounds good to let ADAM3 be followed by ADAM4, you can first test-listen to the object succession by using the command TRAPP (short for TRY and APP), whereupon you can definitely decide with the order APP.

7.16 To better understand the different commands used in order to get orientated in an already stored material, for example the object in your own object library, we introduce pointer, a computer technical

conception. A pointer points to what is to be used in a coming operation. The pointer can be made to point at the first line in a file named NAME, and this file contains - or is - the object. This is done by the command TOP(NAME). The composer can then do something with the object with this name, for instance read in lines from it with an IN-command. He then receives these lines from the file and does not have to write what is written there on the commentary medium. Within the object the pointer can point out the different lines, and the composer can then via commands do something with the line the pointer points at, for instance move the pointer to the next line by a SKIP-command. After TOP(ADAM), where the pointer is placed in the beginning of the object ADAM, the pointer can be moved to point at line 51 with the help of SKIP(50).

At this point we will also introduce another technical conception. Saying that an object is in TEMP means that the object is stored on a disk, which we from now on will call the TEMP-disk. Consequently, an object is not automatically in TEMP because it is written out on the teletype. If the composer only wants to have a look at what he has written in the object, it is enough to write it on the teletype, and the object needs not be in TEMP.

To have a look at an object use the command TOP(NAME), followed by LOOK(NUMBER). The number of lines of the object asked for, are then written out on the teletype. The first line written out is also the first line of the object. If, however, the first 20 lines are not to be written out, but only the ten lines from 21 to 30, write:

```
TOP(NAME)
SKIP(2Ø)
LOOK(1Ø)
```

or

```
TOP(NAME)
SKIP(3Ø)
LOOK(1Ø)
SKIP(2)
LOOK(5)
etc. etc.
```

The difference between only looking at the object and to also have

it in TEMP, is that you cannot transfer to ACC an object which is not in TEMP. To be able to perform commands like MIX, the object must be in TEMP. We place an object in TEMP with the command CALL(NAME). The whole object is then read in to the TEMP-disk. If we do not need to have the full object in TEMP, we can use the orders IN(NUMBER) and SKIP(NUMBER). The order SKIP is consequently used both for excluding something in an object on the teletype and for excluding something on the TEMP-disk. On the other hand, the command LOOK has been replaced by the command IN, for reading into the TEMP-file. We can consequently write: CALL(ADAM) if we want to have the whole object on the TEMP-disk. If we only want to have the first ten lines, we write:

```
TOP(ADAM)
IN(1Ø)
```

If we first want to look at an object in order to possibly later on read in only parts of the object on the TEMP-disk, we can, for instance, write like this:

```
TOP(ADAM)
LOOK(1ØØØ)
TOP(ADAM)
SKIP(1)
IN(2Ø)
SKIP(3Ø)
IN(27)
```

The lines 2 to 21 plus the lines 52 to 78 have then been read into the TEMP-disk and can then be made into an object with its own name with the order SAVE(NAME).

Even if it is necessary that the object is stored in TEMP to be able to transfer it to ACC, it is not necessary that the object is in TEMP to be able to correct errors in it. The following error correction procedure can be used.

The first line in an object is wrong. We call the object to the display by TOP, followed by a LOOK. We observe that the first line is wrong. With the order SKIP(1) we eliminate the first line and write in the correct one. Thereafter we use the command IN and thereby read the object to TEMP.

```
TOP(ADAM)
LOOK(5)
SKIP(1)
CLEAR
LT(1,44Ø,8Ø,2)
IN(56)
```

The fifth line is the correct text. The command CLEAR is made if we want to store the corrected version of the object in the library again, but avoid to get any old terms, possibly already in TEMP.

A last example:
To correct an error in an object in the object library, for instance use waveshape 4 instead of 2 in generator no 5 starting to sound at LT(2ØØØ) and written in line 14 in the object ADAM3, and make this correction on the TEMP-disk, write:

```
1.  TOP(ADAM3)
2.  LOOK(2Ø)
3.  IN(13)
4.  SKIP(1)
5.  LT(2ØØØ)FG(5,263,83,4)
6.  IN(27)
```

Commentary:

Line 2. LOOK writes out the text of the object on the display.

Line 3. The number in the parenthesis tells how many lines we want to have read in.

Line 4. Skip the erroneous line.

Line 5. Enter the correct text.

Line 6. Read in the rest of the text of ADAM3.

Always choose a number rather on the large side so that you make sure that all lines can be included.

7.17 If we have made such important changes that we want to give the object a new name, we use the commands:

```
SAVE(EVA)
ERASE(ADAM)
```

meaning that the new name is EVA and that ADAM is erased on the DT.(Be careful in using ERASE, as it erases the whole object ADAM. You must be positive that you will not need it again.)

7.18 PART(NUMBER) and END

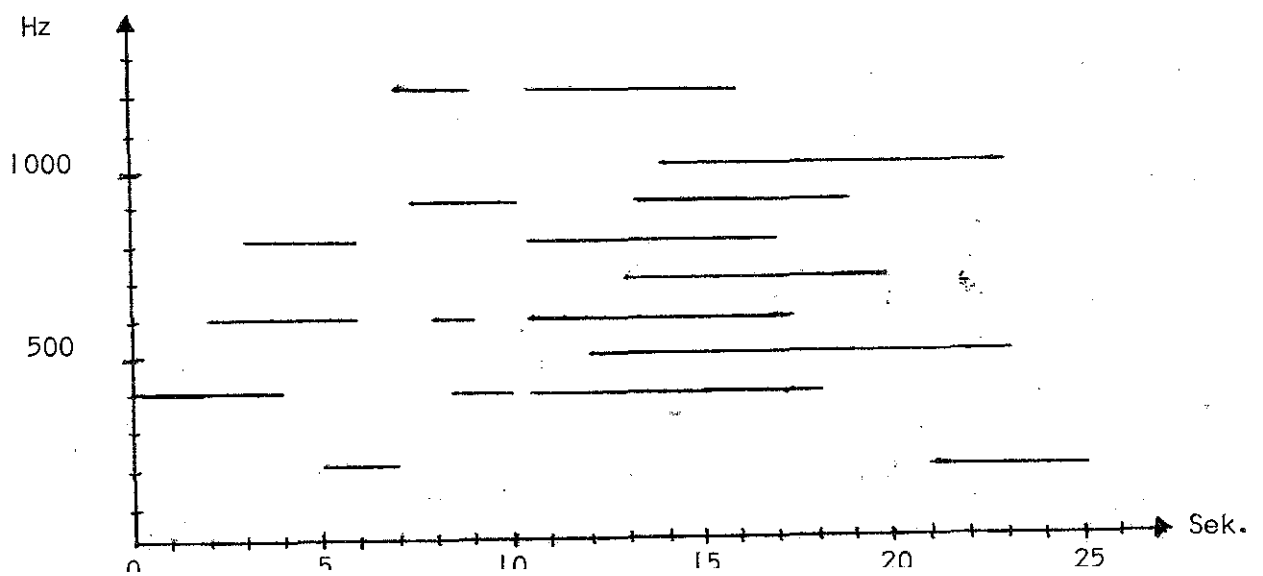
The composer can join his objects into parts of a composition, here called PART(NUMBER). For technical reasons the composer must always start the creation of the first sound with the command PART(NUMBER), for instance PART(ONE). Also for technical reasons, he must not use more than about 14000 changes within each PART of the composition. A PART is ended with the command END. There are no technical obstacles to be generous with the number of PARTS, the composition is divided into. After END, the part is on magnetic tape under the name given after PART, and in such a form that it can be played in the studio immediately. The next PART comes immediately after END, and keeps all the values which the preceeding PART has left.

7.19 The command STDTIM, standard time, allows a variation of the tempo in which the composition is to be played. The reference for the tempo is written as STDTIM(1ØØØ), meaning that the composition is played in the tempos decided by indication of the Local time terms. The number 1000 stands here for 1000 ms, and the term means that programmed 1000 ms shall be performed during 1000 ms, i.e. without any modifications. If we write STDTIM(2ØØØ) the tempo has been doubled, as 2000 ms now is to be played within 1000 ms. The corresponding command to play the composition with half speed is naturally STDTIM(5ØØ). STDTIM can consequently be considered as a metronome tempo, more precisely the number of "beats with the length of a formal millisecond" during one second real time. STDTIM is of importance only to the play-commands and is valid for all performances until it is rechanged.

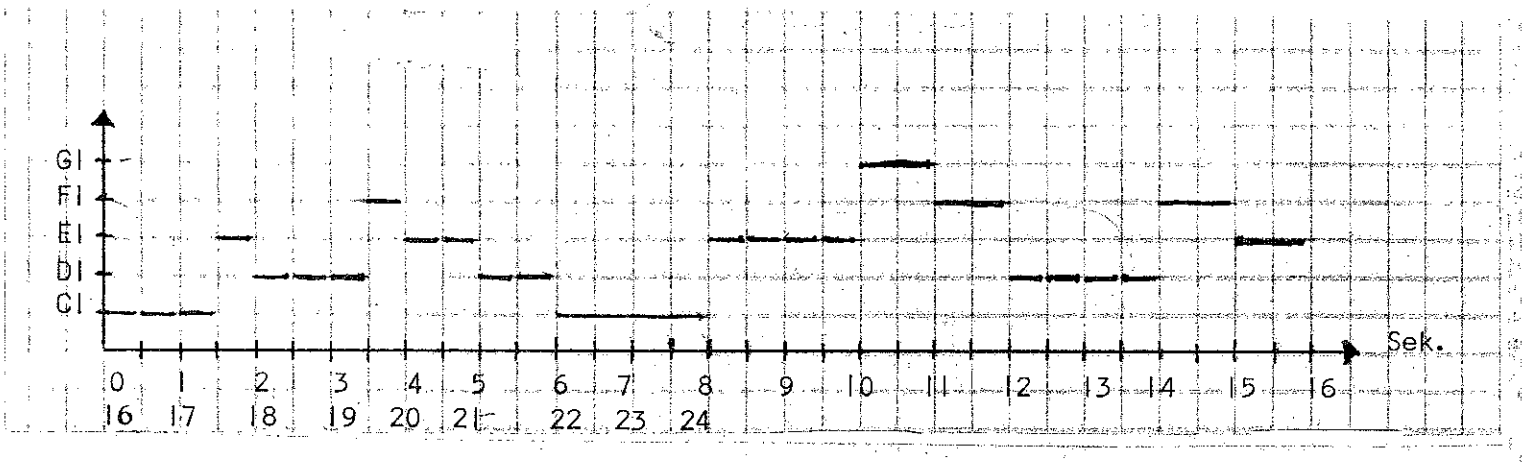
Exercise 4:

Realize the two short compositions described in the following diagrams:

A.



B.



How to solve exercise 4.
(diagram A)

```
PART(TWO)
LT(Ø)
FG3&FG6&FG9&FG12>CHA(1,1ØØ);
FG(1,1ØØ)FG(2,2ØØ)FG(3,3ØØ)FG(4,4ØØ)FG(5,5ØØ);
FG(6,6ØØ)FG(7,7ØØ)FG(8,8ØØ)FG(9,9ØØ)FG(1Ø,1ØØØ);
FG(11,11ØØ)FG(12,12ØØ);
LT(Ø)  FG(4,,8Ø)>D(4ØØØ);
LT(2,) FG(6,,8Ø)>D(4ØØØ);
LT(3,) FG(8,,8Ø)>D(3ØØØ);
LT(5,) FG(2,,8Ø)>D(2ØØØ);
LT(7,) FG(12,,8Ø)>D(2ØØØ);
LT(7,5ØØ)    FG(9,,8Ø)>D(25ØØ);
LT(8,) FG(6,,8Ø)>D(1ØØØ);
LT(8,5ØØ)    FG(4,,8Ø)>D(15ØØ);
LT(1Ø,5ØØ)   FG(4,,8Ø)>D(75ØØ);
             FG(6,,8Ø)>D(7ØØØ);
             FG(8,,8Ø)>D(65ØØ);
             FG(12,,8Ø)>D(55ØØ);
LT(12,)      FG(5,,8Ø)>D(11ØØØ);
LT(13,)      FG(7,,8Ø)>D(7ØØØ);
LT(13,5ØØ)   FG(9,,8Ø)>D(55ØØ);
```

cont.

```
LT(14,)   FG(1Ø,,8Ø)>D(9ØØØ);
LT(21,)   FG(2,25Ø,8Ø)>D(4ØØØ);
MIX
END
```

8. Macros and variables

8.1 Macros

To realize the diagram B the following is done.
C1, D1, etc. stand for tones in the one marked octave:

```
PART(ONE)
LT(Ø)     FG3>CHA(1,1ØØ);
FG(1,262,8Ø,3)>D(4ØØ);
LT(5ØØ)   FG(1,,8Ø)>D(4ØØ);
LT(1,5ØØ)  FG(3,33Ø,8Ø)>D(4ØØ);
LT(2,)    FG(2,294,8Ø,3)>D(4ØØ);
LT(2,5ØØ)  FG(2,,8Ø)>D(4ØØ);
```

After only three seconds, we remark that this is getting laborious Is there no other way?

There are few frequencies and few durations. For each of these repeated sounds, it is possible to define so called macros for the corresponding terms or group of terms. With the help of macros the sequence would look as below.

In the lines 2-6 we give the names C1, D1, etc. to the five FG-terms that need to be used. In line 7-9 names are given to the three D-terms needed. In the following C1 can be written instead of the full term, etc. and a more readable program with less labour is achieved.

```
1    PART(ONE)
2    C1="FG(1,262,8Ø,3)";
3    D1="FG(2,294,8Ø,3)";
4    E1="FG(3,33Ø,8Ø,3)";
5    F1="FG(4,349,8Ø,3)";
6    G1="FG(5,392,8Ø,3)";
7    DUR19="D(19ØØ)";
8    DUR9="D(9ØØ)";
9    DUR4="D(4ØØ)";
```

cont.

```
LT(Ø)FG(I<-6)>CHA(I,IØØ);
CI>DUR4;
LT(5ØØ)CI>DUR4;
LT(I,)CI>DUR4;
LT(I,5ØØ)EI>DUR4;
LT(2,)DI>DUR4;
LT(2,5ØØ)DI>DUR4;
LT(3,)DI>DUR4;
LT(3,5ØØ)FI>DUR4;
LT(4,)EI>DUR4;
LT(4,5ØØ)EI>DUR4;
LT(5,)DI>DUR4;
LT(5,5ØØ)DI>DUR4;
LT(6,)CI>DURI9
SAVE(ADAM)
CLEAR
LT(Ø)FG(I>6)>CHA(I,IØØ);
EI>DUR4;
LT(,5ØØ)EI>DUR4;
LT(I,)EI>DUR4;
LT(I,5ØØ)EI>DUR4;
LT(2,)GI>DUR9;
LT(3,)FI>DUR9;
LT(4,)DI>DUR4;
LT(4,5ØØ)DI>DUR4;
LT(5,)DI>DUR4;
LT(5,5ØØ)DI>DUR4;
LT(6,)FI>DUR9;
LT(7,)EI>DUR9;
SAVE(BERTIL)
CLEAR
CALL(ADAM)
MIX
CALL(BERTIL)
MIX(8ØØØ)
CALL(ADAM)
MIX(I6ØØØ)
PLAY(MIX)
END
```

Here we define the two objects ADAM and BERTIL. We then make a CLEAR to set the TEMP-file to zero as initial condition. The symbol table is not concerned by CLEAR and the macro definitions remain therefore.
By CALL(ADAM) the sound object ADAM is read.
By MIX, ADAM is transferred to ACC. The effect is also the same as after CLEAR, and after CALL(BERTIL) the sound object BERTIL is the only remaining object in TEMP. By MIX(8ØØØ) BERTIL is inserted 8 seconds into ACC. The sound object ADAM returns after 16 seconds from the beginning of ACC.

8.2 A macro is defined in the following way:
macro-name== "a text which can be terms or group of terms";
The name shall consist of maximum six letters and/or numbers, out of which the first must be a letter.

Exercise 5:

We want to create a tonica-dominant-subdominant sequence (T-D-T-S) and want it to sound with three different levels, 70, 80, and 90 dB. A frequency table is in appendix 3.

How to solve exercise 5.

```
PART(TWO)
FREMAT="FG(1,262,,3);FG(2,33Ø,,3);FG(3,392,,3)";
FREMAD="FG(3,392,,3);FG(2,494,3);FG(1,587,,3);"
FREMAS="FG(1,262,,3);FG(2,2Ø8,,3);FG(3,175,,3);"
NIV1="FG(1←3,,7Ø)";
NIV2="FG(1←3,,8Ø)";
NIV3="FG(1←3,,9Ø)";
NIV="NIV1";
SEKVMA="LT(Ø);NIV;FREMAT;LT(2,5ØØ)FREMAD;
LT(4,)FREMAT;LT(6,)FREMAS;FG(1←3)>D(15ØØ);"
LT(Ø)FG3>CHA(1,1ØØ);
SEKVMA
MIX(Ø)
NIV="NIV2"
SEKVMA
MIX(75ØØ)
```

cont.

```
NIV="NIV3"
SEKVMA
MIX(15ØØØ)
PLAY(MIX)
END
```

Observe that NIV and SEKVMA are macros, which in their turn call other macros. When SEKVMA is called, it calls in its turn NIV, which calls NIV1, which sets the level to 70 dB in the first three frequency generators. The control is returned to SEKVMA, which then calls FREMAT, and so on. TEMP is sorted over to ACC by MIX(Ø). After this TEMP is CLEARed, but the macro definitions still remain in the symbol table. The macro NIV is redefined and executes NIV2, next time it is used by SEKVMA. By MIX(75ØØ) the result from SEKVMA is sorted, i.e. a series of chords, now with the level 80 dB into ACK. The third chord sequence is realized in the same way with NIV3 and MIX(15ØØØ).

8.3 OLD

In a case where the starting value of an envelope- or glissando term is the same as that of the previous term, it is possible to write OLD instead. If you want to shift the value of a preceeding term, for instance with 10 upwards or downwards, write OLD+1Ø resp OLD-1Ø.
Example:

```
LT(Ø)
FG(1,1ØØ,Ø)>ENV(OLD,1ØØ,5ØØ,2)>ENV(OLD,OLD,1ØØ)>
            ENV(OLD,5Ø,5ØØ,2)>Z;
LT(18ØØ)FG(1,OLD-5Ø,1ØØ)>ENV(OLD+1Ø,Ø,6ØØ,3);
```

Observe that OLD is no variable and therefore must be used solely in ENV-, GLIS-, and device-terms.

8.4 Constants

In EMS-2 there is a possibility to signify numerical values with a symbol instead of stating them explicitly, i.e. give a name to the numerical value. This is done in the following way:

```
KALLE=44Ø;
ANDERS=23
AI=44Ø
```

cont.

```
MF=7Ø;F=8Ø;FFM=9Ø;
AØ=22Ø;FØ=175;C1=262;
E1=33Ø;G1=392;H1=494;
D2=582;
```

The solution to the exercise above can then be written like this:

```
FREMAT="FG(1,C1,,3)FG(2,F1,,3)FG(3,G1,,3)";
FREMAD="FG(3,G1,,3)FG(2,H1,,3)FG(1,D2,,3)";
FREMAS="FG(1,C1,,3)FG(2,AØ,,3)FG(3,FØ,,3)";
NIV1="FG(1←3,,MF)";
NIV2="FG(1←3,,F)";
NIV3="FG(1←3,,FFM)";
```

8.5 Variables

It is possible, if desired, to give numerical values to names and use the name in the place of the figures.

```
AB=1ØØ;
FG(1,AB);
AB=2ØØ;
FG(1,AB);
```

If you write AB==17 the variable AB is protected from further changes. This possibility to change values is especially useful in connection with macros. This can be illustrated by realizing the example in 6.1 (EMS-1) in a more simple way:

```
PART(ONE)
CLEAR
LT(Ø)
M1="FG(NR,FREQ,NIV,Ø CHA(NO,1ØØ);";
NR=1;FREQ=2ØØ;NIV=7Ø;NO=1;
M1;
NR=4;FREQ=3ØØ;NIV=67;NO=2;
M1;
NR=7;FREQ=4ØØ;NIV=64;NO=3;
M1;
NR=1Ø;FREQ=6ØØ;NIV=61;NO=4;
M1;
LT(3Ø,)CHA(1)>Z;
```

cont.

```
SAVE(ADAM)
MIX(Ø)
PLAY(MIX)
END;
```

Exercise 6:

Realize the chord example in the same way by using 3 macros instead of 8.

How to solve exercise 6:

```
PART(TWO)
FREMAA="FG(1,FREQ1,,3)FG(2,FREQ2,,3)FG(3,FREQ3,,3)";
NIVAA="FG(1>3,,DB)>CHA(1,1ØØ)>D(75ØØ)";
SEKVMA="LT(Ø)NIVAA;FREQ1=262;FREQ2=33Ø;
FREQ3=392;FREMAA;
LT(2,5ØØ)FREQ2=294;FREQ1=587;FREMAA;
LT(4,)FREQ1=262;FREQ2=33Ø;FREMAA;
LT(6,)FREQ2=2Ø8;FREQ3=175;FREMAA;"
LT(Ø)
DB=7Ø;SEKVMA;
MIX(Ø)
DB=8Ø;SEKVMA;
MIX(75ØØ)
DB=9Ø;SEKVMA;
MIX(15ØØØ)
PLAY(MIX)
END
```

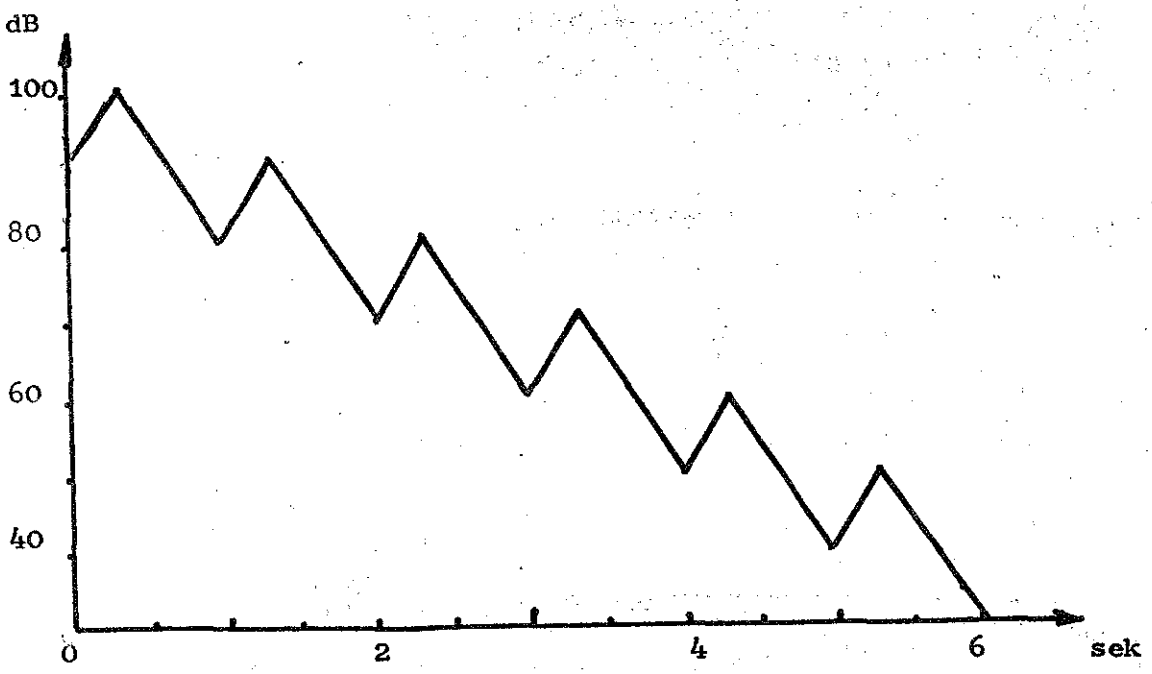
The so called macro expansion of SEKVMA looks as follows:

```
LT(Ø)
FG(1<3,,DB)>CHA(1,1ØØ)>D(75ØØ);
FREQ1=262;FREQ2=33Ø;  etc.,
```

i.e. all that the used macros have done. By MIX(75ØØ) all will therefore be sorted in again starting at 7.5 seconds, but we will first give the variable DB a new value, 80. In the same way the chord sequency is sorted in at 15 seconds with the level 90. Another variation is that the duration is set on a CHA-term in the level-macro.

Exercise 7:

Realize the example below. It is not allowed to use other variables than OLD:

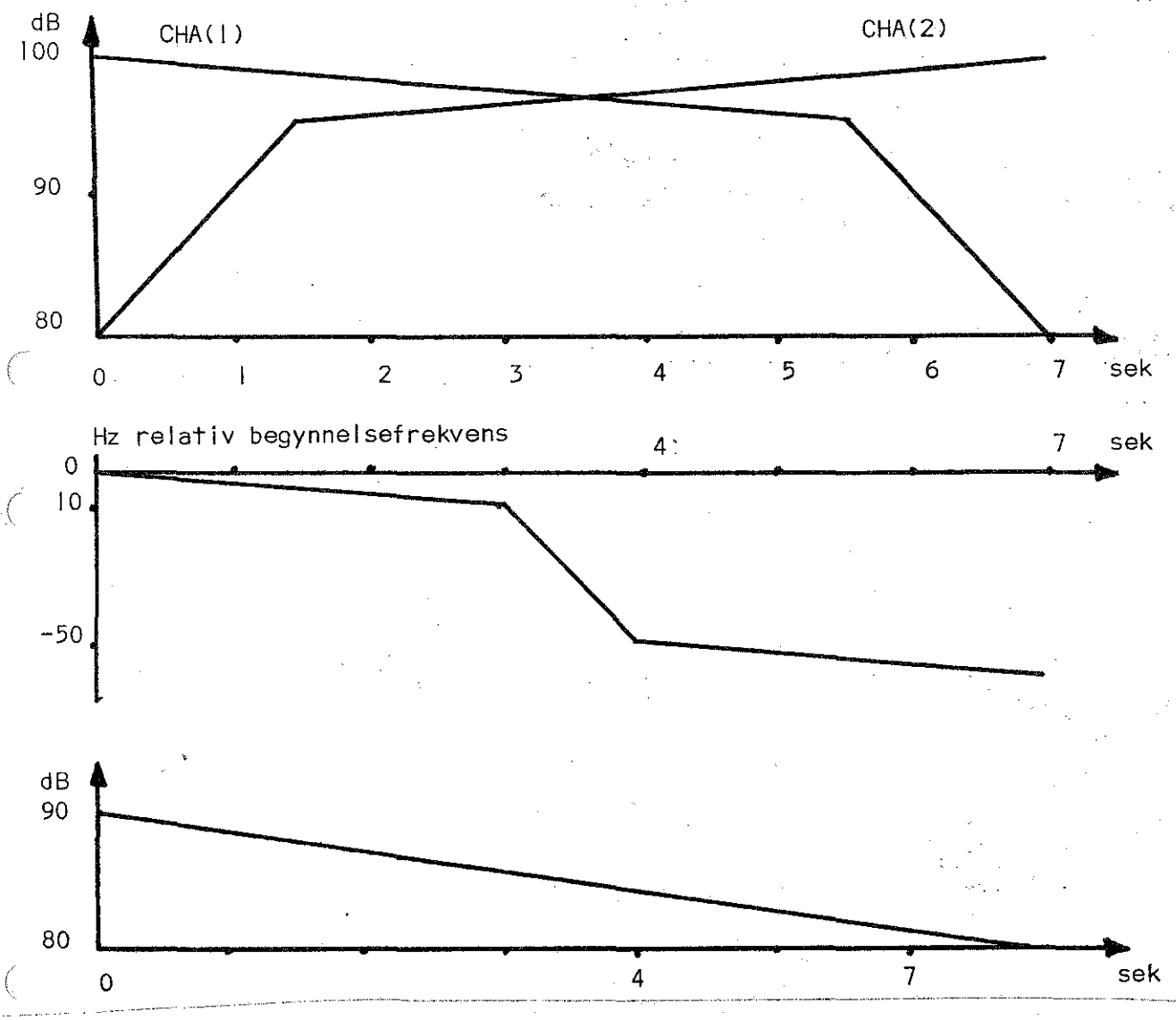


How to solve exercise 7:

```
PART(THREE)
CLEAR
MI="ENV(OLD,OLD+1Ø,3ØØ)>ENV(OLD,OLD-2Ø,7ØØ)";
LT(Ø)
FG(1,1ØØ,9Ø,6)>MI>MI>MI>MI>MI>MI>CHA(1,1ØØ);
SAVE(DESCND)
MIX
PLAY(MIX)
END
```

Exercise 8:

Realize the example below:



Channel 1 and channel 2 shall vary between 8Ø and 1ØØ dB according to the first figure. The frequency generators are given the following starting frequencies: FG(1) 400 Hz

FG(2) 590 Hz

FG(3) 780 Hz

Each of these three frequencies are deduced with 60 Hz according to the second figure. The level of the three frequencies goes from 90 to 80 dB during the seven seconds the sound lasts.

```
How to solve exercise 8:
PART(FOUR)
CLEAR
LT(Ø)
FG(1,4ØØ)&FG(2,59Ø)&FG(3,78Ø)>
ENV(9Ø,8Ø,7ØØØ)>
GLIS(OLD,OLD-1Ø,3ØØØ)>
GLIS(OLD,OLD-4Ø,1ØØØ)>
GLIS(OLD,OLD-1Ø,3ØØØ);
FG3>CHA(1)>ENV(1ØØ,96,55ØØ)>ENV(96,8Ø,15ØØ);
FG3>CHA(2)>ENV(8Ø,96,15ØØ)>ENV(96,1ØØ,55ØØ);
SAVE(SWOCHE)
MIX
PLAY(MIX)
END
```

SUMMARY OF ELEMENTS IN EMS-2

Device terms:

```
AM(NO,ENTRY,LEVEL)
RD(NO,CHANNEL,LEVEL)
CD(NO,CHANNEL,LEVEL)
CHA(NO,LEVEL)
FF(NO,FILTERCHANNEL,LEVEL)
FG(NO,FREQUENCY,LEVEL,WAVEFORM)
NG(NO,LEVEL,COLOUR1,COLOUR2)
REV(NO,REVTIM,LEVEL)
RM(NO,ENTRY,LEVEL)
```

Studio points:

```
FS
FG3,FG6,FG9,FG12,FG15,FG18,FG21,FG24
```

Commands:

```
APP
CLEAR, CLEAR(MIX), CLEAR(ALL)
DELETE(SYMBOL)
KEEP, KEEP(MIX), KEEP(ALL)
MIX(TIME)
STDTIM(MS)
```

TEMP commands:

CALL(FILENAME)

ERASE(FILENAME)

IN(LINENUMBER)

LOOK(LINENUMBER)

PLAY, PLAY(MIX), PLAY(PARTNAME), PLAY(PART1,PART2)

REPL(FILENMAE) (functions as ERASE followed by SAVE)

SAVE(FILENAME)

SKIP(LINENUMBER)

TOP(FILENAME)

TRAPP

TRY, TRY(MS), TRY(SEC,MS)

Allow a line of its own to each command.

Local time:

LT(MS), LT(SEC,MS)

Observe that LT(1,5) means the same as LT(1,005), not LT(1,500).

Envelope terms:

ENV(LEVEL,LEVEL,MS,CURVESHAPE,STEP)

Z functions as ENV(OLD,0,10,0,10)

D(MS) functions as ENV(OLD,OLD,MS)>Z

Glissando terms:

GLIS(FREQUENCY,FREQUENCY,MS,CURVESHAPE,STEP)

Step terms:

ESTEP(MS)

GSTEP(MS)

EXIT interrupts the run and starts up EMS-1 for a new program.

>, #, & stand between elements in connection chains, envelope chains and glissando chains. If the chain continues over several lines it shall be divided so that one of these signs is the last one on the line.

; ought to end each chain

← can be used in FG- and FF-terms

'COMMENTARY'

It is advisable to use commentaries often. It is easy to forget very soon what a program is about.

MACRO-NAME="...TEXT..."

Macro definition. It is not allowed to write anything else on the same line after the end-sign.

MEX — Can be included in the macro text (MACRO-EXIT).

Macros can call each other up to 100 levels. No macro must include CALL.

↑ — Arithmetic operation for exponentiation.

EMS-3

8.5 Arithmetic operations

Besides the possibility to give a number to a symbol, it is also possible to give a symbol a value, as a result of an arithmetic expression.

Example:

A1=2Ø; A2=23; A3=5;

A4=A2-A1+A3; A4=A4-5;

gives the result A4=3.

This operation is especially useful together with macros. A variable can then be increased or decreased with a certain quantity each time a certain macro is called.

We return to the example in 6.1 (EMS-1)

```
1    PART(ONE)
2    CLEAR
3    M1="FG(NR,FREQ,NIV,Ø)>CHA(NO,1ØØ)>D(3ØØØØ);
4    NR=NR+3;NO=NO+1;NIV=NIV-3;
5    LT(Ø)NR=1;NIV=7Ø;NO=1;FREQ=2ØØ;
6    M1
7    FREQ=3ØØ;M1;
8    FREQ=4ØØ;M1;
9    FREQ=6ØØ;M1;
1Ø   SAVE(ADAM)
11   MIX(Ø)
12   PLAY(MIX)
13   END
```

Line 3-4: The macro is defined as in 8.5 but with the addition of NR=NR+3; NO=NO+1; NIV=NIV-3;

This addition gives the result that each time the macro is used, the value of NR is increased with 3, NO with 1 and NIV is decreased with 3, giving these variables the values which shall be in the FG-expression next time the macro is used.

Line 5: Here NR=1; NIV=7Ø; FREQ=2ØØ;

These values are used in the macro at the call on line 6 and are

then automatically changed to 4,2 resp 67.

Line 7: FREQ is set to 300 and used together with the values NR=4, NO=2, and NIV=67 in the macro. NR, NO and NIV are thereby changed to resp. 7,3,64, etc.

The time in EMS-I is indicated with the time from the start of the object, i.e. absolute time. In some cases it would be better with relative time, i.e. to be able to indicate how long a certain time indication shall last without having to calculate the time for the next occasion.
We will now show how to use relative time indication with the help of variables which automatically are incremented in a macro, and thereby also with the help of a macro on a higher level, can represent a whole sequence of occasions to be executed at different times.

8.6 Examples of automatic updating in a macro

Suppose that the sound-object is not longer than 99999 ms. Local time can then be indicated in only milliseconds. Suppose further that we want a macro which together with the variables TICK and TACK work in the following way: In the beginning of the sound-object TICK is set to zero. The object is supposed to start with a LT(Ø). TICK is then increased little by little and indicates absolute time within the sound-object.

```
PART(ONE)
LT(Ø)
'definition of the macro RELTID'
TICK=Ø
'other macro definitions, operations, etc. to be executed at the
time zero'
TACK=the time until next occasion
RELTID
other macrocalls and operations to be executed at this time:
TACK=the time until next occasion
RELTID
ETC
```

Exercise 9:

Describe the macro definition for RELTID.

How to solve exercise 9:

The macro RELTID looks as follows:

RELTID="TICK=TICK+TACK; LT(TICK)"

Naturally, it is possible to mix the use of relative time and absolute time, or set TICK to zero and start describing a parallel sequence.

Exercise 10:

Use the macro RELTID in the chord example in 8.2. The tonica is at odd occasions 2.5 seconds (i.e. first, third, fifth, ... time) and at even occasions 2 seconds. The dominant is as the subdominant 1.5 seconds.

```
PART(TWO);
.
.
.    as before
.
.
SEKVMA="RELTID;NIV;FREMAT;
TACK=25ØØ;RELTID;FREMAD;
TACK=15ØØ;RELTID;FREMAT;
TACK=2ØØØ;RELTID;FREMAS;
TICK=Ø;TACK=Ø;SEKVMA
NIV="NIV2"
TACK=15ØØ;SEKVMA
NIV="NIV3"
TACK=15ØØ;SEKVMA
FG(1←3)>D(15ØØ);
MIX;
PLAY(MIX)
END;
```

TACK=...;RELTID can also be included in the macro definitions:

.

.

.

.

cont.

```
.
FREMAT="FG(1,262,,3)FG(2,330,,3)
FG(3,392,,3);RELTID"
FREMAD="FG(3,392)FG(2,494)FG(1,587);TACK=1500;RELTID"
FREMAS="FG(1,262)FG&2,208)FG(3,175);TACK=1500;RELTID"
SEKVMA="NIV;TACK=2500;FREMAT;FREMAD;TACK=2000
     FREMAT;
     FREMAS"
.
.
.
```

FREMAT must be given the value of TACK, as two different values are used. FREMAD and FREMAS contain in themselves the value 1500 of TACK.

Example:

The chord example in 8.2 can be re-formulated in the following way:

We want to have different appearance of the chord tones, so that the lower tones are attenuated somewhat faster than the higher ones, We therefore introduce three envelope macros.

```
ENVMA1="ENV(0,90,200,-1)>ENV(90,80,100,2)>ENV(80,50,2700);";
ENVMA2="ENV(0,90,200,-1)>ENV(90,82,100,2)>ENV(82,56,2700);";
ENVMA3="ENV(0,90,200,-1)>ENV(90,84,100,2)>ENV(84,62,2700);";
FREMAD="FG(3,G1,,3)>ENVMA1;
        FG(2,H1,,3)>ENVMA2;
        FG(1,D2,,3)>ENVMA3;
        TACK=1500;RELTID";
etc.
```

9. Conditionals

Suppose that you want to repeat the following macro sequence 25 times: SEKVMA="FREMAT;FREMAD;FREMAT;FREMAS";

You can write: SEKVMA;SEKVMA;SEKVMA;.....SEKVMA;SEKVMA;

1 2 3 24 25

This way of working is laborious and is also not working because TICK will be greater than 99999 ms in the macro RELTID after the 9th repetition of SEKVMA.

How to better and simpler realize the macro sequence? We want to indicate in a simpler way that SEKVMA shall be executed 25 times, and then we must also start using both arguments of LT (seconds, milliseconds).

This problem can be solved with the help of another macro level, a variable used as counter and a new conception now introduced: In EMS-3 there are so called conditionals, with the help of which the value of a variable can be made to generate the continued course, for instance after the value of the variable has been operated with an arithmetic expression.

The arithmetic conditionals in EMS-3 are:
IFPOS(VARIABLE)
IFZER(VARIABLE)
IFNEG(VARIABLE)

VARIABLE is (according to previous information) a symbol of at the most six alphanumerical characters, of which the first is a letter. IFPOS(VARIABLE) means that what is between the right parentheses,), and the nearest <-symbol is executed when the variable is greater than zero. The same is valid for IFZER(...), where the corresponding string is executed when the variable within the parentheses has the value zero, and also for IFNEG(VARIABLE), where the string is executed only when the variable is less than zero.
Example:
NR=1;NO=Ø;
a)IFPOS(NR)NR=NR-1;NO=NO+1;<
b)IFPOS(NR)NR=NR-1;NO=NO+1;<

In a) NR=1, i.e. greater than zero, meaning that the string is executed and that NR is =Ø and NO is 1.
In b) NO=Ø, i.e. not greater than zero, meaning that the string is not executed and the program continues after the <-symbol.

If we want NO to be =1ØØØ for NR=Ø and only for NR=Ø, we can write:
IFZER(NR)NO=1ØØØ;<
The general expression for conditionals is:

STRING TO BE EXECUTED IF
CONDITIONAL(VARIABLE) ;<
CONDITION FULFILLED

end symbol

The program continues after the end symbol if the conditional is not to be executed.

We return to the initial example:

RELTID: To avoid that the millisecond part gets too great, we can test its value and decrease it with 1000 if the millisecond part is greater than 1000, at the same time as the second part is increased with 1.

```
1)   RELTID="TICK=TICK+TACK-1000;
2)          !FPOS(TICK)STICK=STICK+1;<
3)          !FZER(TICK)STICK=STICK+1;<
4)          !FNEG(TICK)TICK=TICK+1ØØØ;<
5)          LT(STICK,TICK)";
```

Line 1: TICK is set to the same as TICK+TACK (TACK is the time until next occurence. This is as before but also 1ØØØ (ms) is then subtracted.

Line 2-3: If the result still is greater than or equal to zero, i.e. TICK was greater than 1000 before the subtraction was executed, the second part STICK (which is supposed to already have a value) is increased by 1 (second) and consequently the millisecond part can be kept down by using the second-part.

Line 4: If TICK+TACK was less than 1000 the result after the subtraction by 1000 becomes negative and we can (and must!) restore the old value by adding 1000 to TICK.

The main part of the example consisted of repeating the macro SEKVMA 25 times. This can now be solved in the following simple way: we introduce a macro on a higher lvel and a counter, RAEKN.

```
.
.
.
RAEKN=25;TICK=Ø;STICK=Ø;
T25="!FPOS(RAEKN)RAEKN)RAEKN=RAEKN-1;SEKVMA;
     T25;<"
.
.
.
```

This means that if RAEKN is greater than zero, RAEKN is decreased by 1. Thereafter SEKVMA is executed and the procedure is repeated

until RAEKN is zero.

If a variable has not ever been used, it can not have acquired any value, and is therefore denoted undefined. There are two conditionals which examine whether a symbol is defined or not. These are:
IFUND(SYMBOL)MEASURES;≺if undefined ... ..
IFDEF(SYMBOL)MEASURES;≺if defined ... ..

If it is considered practical not to have to define the variables of a macro before it is called, it is possible to use the above conditionals A practical use of this gives RELTID its final appearance:

```
RELTID=IFUND(STICK)STICK=Ø;≺
       IFUND(TICK)TICK=Ø;≺
       IFUND(TACK)TACK=Ø;≺
       TICK=TICK+TACK-1ØØØ;
       IFPOS(TICK)STICK=STICK+1;≺
       IFZER(TICK)STICK=STICK+1;≺
       IFNEG(TICK)TICK=TICK+1ØØØ;≺
       LT(STICK,TICK)"
```

Now it is possible to replace LT(Ø) by RELTID, for if the variables STICK,TICK and TACK are undefined when RELTID is called, they are defined and set to zero and the result is consequently the time argument Ø in the LT-term.
If by any reason you want a variable (or a macro) to be excluded from the table (symbol-table), where all used names and corresponding values are stored, there is a possibility to do so. The command for this is DELETE(NAME).
If you want to use the macro RELTID in the IFUND-alternative, and in several sound-objects, or for parallel courses, you probably want to have STICK and TICK set to zero. One way of doing this is to make them undefined so that RELTID automatically defines them again and sets the value to 0. The command is DELETE(STICK,TICK);

10. A few other commands

Commands of the type WRITE(VARIAB) can be used to examine for instance the course of complicated macros.
Example:

```
                EMS1 V1.4
                NAME OF INPUT FILE?
                INTERACTIVE MODE? YES OR NO?
YES
:TOP(ACVIBF)
:LOOK(88)
ACCVIB="LT(TID)FG(1)>GLIS(434,446,GTID)>GLIS(446,434,GTID);
WRITE(TID,GTID);
SLASK=GTID/2Ø;
GTID=GTID-SLASK;
TID=TID+GTID;
TEST=GTID-5Ø;
IFPOS(TEST)ACCVIB<;"
Ø5    ØØ2       END OF FILE ON ADDITIONAL INPUT FILE.
:CALL(ACVIBF)
Ø5    ØØ2       END OF FILE ON ADDITIONAL INPUT FILE.
:PART(ACVIBF)
:LT(Ø)FG(1,,8Ø)>CHA(4,1ØØ);
:TID=Ø;GTID=4ØØ;
:ACCVIB;
'TID   =      Ø  TYPE:  Ø1  '          'GTID  =     55  TYPE:  Ø1  '
'GTID  =    4ØØ  TYPE:  Ø1  '          'TID   =   6981  TYPE:  Ø1  '
'TID   =    38Ø  TYPE:  Ø1  '          'GTID  =     53  TYPE:  Ø1  '
'GTID  =    38Ø  TYPE:  Ø1  '          'TID   =   7Ø32  TYPE:  Ø1  '
'TID   =    741  TYPE:  Ø1  '          'GTID  =     51  TYPE:  Ø1  '
'GTID  =    361  TYPE:  Ø1  '        :PLAY
'TID   =   1Ø84  TYPE:  Ø1  '
```

By the command MESS(OUTPUT) the program stops, writes out the text OUTPUT, is set in error correction mode and waits for commands.

Example:

```
                EMS1 V1.4
                NAME OF INPUT FILE?
                INTERACTIVE MODE? YES OR NO?
YES
:TOP(FLIMSF);LOOK(99);
GLIMES="IFUND(TID)TID=Ø;FN=1;FR=55;<
FN=FN-1;
IFZER(FN)FN=3<;
MESS(NEXT FR)
IFPOS(FR)
LT(TID)
FG(FN)>GLIS(FR,44Ø,3ØØØ);
TID=TID+1ØØØ;
GLIMES<;"
Ø5    ØØ2      END OF FILE ON ADDITIONAL INPUT FILE.
:IN(99)
```

```
05   002        END OF FILE ON ADDITIONAL INPUT FILE.
:PART(MESDEM)
:LT(0)FG(1>3,,80)>CHA(2,100);
:GLIMES
NEXT FR
!FR=220
!
NEXT FR
!FR=110
!
NEXT FR
!FR=55
!
NEXT FR
!FR=83
!
NEXT FR
!FR=110
!
NEXT FR
!FR=165
!
NEXT FR
!FR=220
!
NEXT FR
!
NEXT FR
!
NEXT FR
!PLAY
!FR=1760
!
NEXT FR
!FR=1320
!
NEXT FR
!FR=880
!
NEXT FR
!FR=660
!
NEXT FR
!
NEXT FR
!FR=440
!
NEXT FR
!
NEXT FR
!FR=0
!
:PLAY
:MIX
:END
```

In non-interactive mode (if you answer NO to the question INTERACTIVE MODE? YES OR NO) MESS is not working and WRITE does not write on the commentary medium, but only on the list-medium (logical unit 5).

```
EMS1 V1.4
NAME OF INPUT FILE?

    INTERACTIVE MODE? YES OR NO?
YES


:PART(ONE)
:MM="MS;MM"
:MS="    ' - - - '
:WRITE(CNT);
:CNT=CNT-1;
:IFZER(CNT)MEX<"
:CNT=2;
:MM;
 'CNT    =      2  TYPE: Ø1 '
 'CNT    =      1  TYPE: Ø1 '
:
:'PAA DETTA SETT KAN ALLTSAA ETT UNDERORDNAT
:MAKRO AVBRYTA ETT OEVERORDNAT'
```

II. LIST and NOLIST

The file, which has been created by SAVE, can naturally contain macro calls. There is a way to get better control of how the macros develop. By the command LIST it is possible to get in the text, not macro calls, but the terms the macro expands into. If the macro calls other macros these terms are given, etc.

In this way it is simpleter to see the consequences of a macro call, which otherwise might be difficult to look through.

The command NOLIST annulls the expansion.

Example:

```
NAME OF INPUT FILE?
INTERACTIVE MODE? YES OR NO?
YES
:PART(BONE)
:BMAK="FG(17,131);"
BS="FG(18,262)"
```

cont.

```
:LIST
:CLEAR
:FG18>CHA(1,100);
:BMAK
:FG(17)>D(50);
:SAVE(CONE)
:MIX;END
OBJECT NO:        1. BLOCK TIME:        0.000   0.059  FREE MIXREC.S: 13820
BLOCKLABEL:   BONE  .GLOBAL TIME:        0.000   0.059  FREE MIXREC.S: 13820
:TOP(CONE);LOOK(9)
'00005' FG18>CHA(1,100);
'00006'     '---MACRO---:  BMAK'
'00007' FG(17,131);
'00008'     '---MACRO---:  BS'
'00009' FG(18,262)
'00010' FG(17)
'00011'>D(50);

05   002       END OF FILE ON ADDITIONAL INPUT FILE.

:EXIT

--------EXIT. END OF THIS RUN.--------

        EMSI VI.4
        NAME OF INPUT FILE?
--- END OF RUN. EXIT TO MONITOR-----
```

12. More about the arithmetic in EMS-3

The arithmetic operations in EMS-3 are:

+	addition
-	subtraction
×	multiplication
/	division
↑	exponentiation

They are always used in such a way that a variable is set equal to

the expression where the operations can be included, operating on numbers or variables. It is consequently not allowed to have operations within terms. The exception is OLD+ and OLD-, as in FG(1,OLD+2ØØ), but OLD is no variable. There can be two operations in the right hand term.

Example:

x=2x3x4 gives x=24.

As the level can be set to 1/4 decibel in the studio, the variables can also be indicated in 1/4s.

Example:

x=81.2Ø235 gives x the rounded value 81.25. The rounding is in this and similar cases always done to the nearest 1/4. Numbers created with the help of a decimal point are called floating numbers as contrary to integers.

y=6Ø.1 gives y=6Ø.Ø

x and y are floating numbers and used in EMS-3 terms in all places indicating level, but not in other places.

When a variable has been given a value it becomes of the same type, integer or floating number, as the first value to the right of the equal sign.

Example:

VAR=1Ø.25+5	VAR becomes a floating number variable
VAR=5+1Ø.25	VAR becomes an integer variable
VAR=SYM+17-NR	VAR becomes of the same type as SYM

Observe that when two integers are divided the rounding is made downwards, so called integer division. The result is an integer.

Example:

x=99999	gives x the highest value that can be given directly
x=9ØØØØ+41071	gives x the highest value that can be represented at all, 131071.
x=32767.75	gives x the highest possible floating number value. This number is also the greatest number that can result from a multiplication.

It is necessary to bear in mind the limitations in precision and range and organize so that they are best taken care of. If you for example want to increase a variable VAR with a 20th of its old value (for instance in a macro) there are the following three bad alternatives:

VAR=VARx21/2Ø	If VAR is greater than 156Ø you will have overflow on multiplication
VAR=VARx1.Ø5	1.05 is rounded to 1.0, VAR is not changed at all.
VAR=VAR/20x21	The precision is lost. The product can only take on values which are multiples of 21.

The best way to do it is:
VAR=VAR/2Ø+VAR.

SUMMARY OF ELEMENTS IN EMS-3

Device terms:

AM(NO,ENTRY,LEVEL)
AMP(NO,LEVEL)
CD(NO,CHANNEL,LEVEL)
CHA(NO,LEVEL)
FF(NO,FILTERCHANNEL,LEVEL)
FG(NO,FREQUENCY,LEVEL,WAVEFORM)
NG(NO,LEVEL,COLOUR1,COLOUR2)
REV(NO,REVTIM,LEVEL)
RM(NO,ENTRY,LEVEL)

Studio points:

FS
FG3,FG6,FG9,FG12,FG15,FG18,FG21,FG24

Commands:

APP
CLEAR, CLEAR(MIX), CLEAR(ALL)
DELETE(SYMBOL)
KEEP, KEEP(MIX), KEEP(ALL)
LIST
MESS(TEXT)

cont.

MIX(TIME)
NOLIST
STDTIM(MS)
WRITE(SYMBOL)

TEMP commands:

CALL(FILENAME)
ERASE(FILENAME)
IN(LINENUMBER)
LOOK(LINENUMBER)
PLAY, PLAY(MIX), PLAY(PARTNAME), PLAY(PART1,PART2)
REPL(FILENAME) {works as ERASE followed by SAVE)
SAVE(FILENAME)
SKIP(LINENUMBER)
TOP(FILENAME)
TRAPP
TRY, TRY(MS), TRY(SEC,MS)

Allow a line of its own to each command.

Local time:

LT(MS), LT(SEC,MS)
Observe that LT(1,5) means the same as LT(1,005, not LT(1,500).

Envelope terms:

ENV(LEVEL,LEVEL,MS,CURVESHAPE,STEP)
Z works as ENV(OLD,0,10,0,10)
D(MS) works as ENV(OLD,OLD,MS)>Z

Glissando terms:

GLIS(FREQUENCY,FREQUENCY,MS,CURVESHAPE,STEP)

Step-terms:

ESTEP(MS)
GSTEP(MS)

EXIT stops the run and starts up EMS-1 for a new program.

>, #, & stand between elements in connection chains, envelope chains, and glissando chains. If the chain continues over several lines it shall be divided so that one of these characters is the last one on the line.

; ought to end each chain.

← can be used in FG- and FF-terms.

'COMMENTARY'

It is adviseable to use commentaries often. It is easy to forget very soon what a program is about.

MACRO NAME="...TEXT..."

Macro definition. It is not allowed to write anything else on the same line after the end-character.

MEX Can be included in the macro text (MACRO-EXIT).

Macros can call each other up to 100 levels. No macro must include CALL.

+, -, *, /, ↑

Arithmetic operations for addition, subtraction, multiplication, division, exponentiation.

Example: C=D*E+F Maximum three integers or variables in the right hand link.

IFPOS(VARIABLE)...<

IFZER(VARIABLE)...<

IFNEG(VARIABLE)...<

IFUND(SYMBOL)...<

IFDEF(SYMBOL)...<

APPENDIX 1. TILLAATNA STUDIOKOPPLINGAR.

FRAAN	TILL
FG(1)	CHA(1)
FG(2)	CHA(2)
FG(3)	CHA(3)
FG3	CHA(4)
	CD(1)
	FG6
FG(4)	CHA(1)
FG(5)	CHA(2)
FG(6)	CHA(3)
FG6	CHA(4)
	CD(1)
	FG9
	RM(1,A)
	RM(1,B)
	RM(2,B)
	REV(1)
	REV(2)
	AM(1,B)
	FF(1)
	FF(2)
FG(7)	CHA(1)
FG(8)	CHA(2)
FG(9)	CHA(3)
FG9	CHA(4)
	CD(1)
	FG12
FG(10)	CHA(1)
FG(11)	CHA(2)
FG(12)	CHA(3)
FG12	CHA(4)
	CD(1)
	FG15
	RM(1,A)
	RM(1,B)
	RM(2,B)
	REV(1)
	REV(2)
	AM(1,B)
	FF(1)
	FF(2)
FG(13)	CHA(1)
FG(14)	CHA(2)
FG(15)	CHA(3)
FG15	CHA(4)
	CD(1)
	FG18

.FJECT

```
FRAAN    TILL

NG       CHA(1)
         CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         RM(2,B)
         REV(1)
         AM(2,B)
         FF(1)
         FF(2)

FF(1)    CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         RM(1,A)
         RM(1,B)
         RM(2,B)
         REV(2)
         AM(1,A)
         AM(2,B)
         FF(2)
         AMP(1)
         AMP(2)

FF(2)    CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         RM(2,B)
         REV(2)
         AM(1,B)
         AM(2,A)
         FF(1)
         AMP(1)
         AMP(2)

REV(1)   CHA(1)
         CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         AM(1,B)
         AM(2,B)
         AMP(1)
         AMP(2)
.EJECT
```

```
FRAAN    TILL

REV(2)   CHA(1)
         CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         AM(1,B)
         AM(2,B)
         AMP(1)
         AMP(2)

RM(1)    CHA(1)
         CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         REV(2)
         AM(1,A)
         AM(2,B)
         FF(1)
         FF(2)
         AMP(1)
         AMP(2)

RM(2)    CHA(1)
         CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         REV(2)
         AM(1,B)
         AM(2,A)
         FF(1)
         FF(2)
         AMP(1)
         AMP(2)

RM(3)    CHA(1)
         CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
         REV(2)
         AM(2,B)
         FF(1)
         FF(2)
         AMP(1)
         AMP(2)

AM(1)    CHA(1)
         CHA(2)
         CHA(3)
         CHA(4)
         CD(1)
.EJECT
```

```
          FRAAN     TILL

          FG(16)    CHA(1)
          FG(17)    CHA(2)
          FG(18)    CHA(3)
          FG18      CHA(4)
                    CD(1)
                    FG21
                    RM(1,B)
                    RM(2,B)
                    REV(1)
                    REV(2)
                    AM(1,B)
                    AM(2,B)
                    FF(1)
                    FF(2)

          FG(19)    CHA(1)
          FG21      CHA(2)
                    CHA(3)
                    CHA(4)
                    CD(1)
                    FG24

          FG(20)    FG21
                    FS

          FG(21)    FG21
                    RM(1,A)

          FG24      CHA(1)
                    CHA(2)
                    CHA(3)
                    CHA(4)
                    CD(1)
                    RM(1,B)
                    RM(2,B)
                    REV(1)
                    REV(2)
                    AM(1,B)
                    AM(2,B)
                    FF(1)
                    FF(2)

          FG(22)    FG24
                    RM(2,A)

          FG(23)    FG24
                    AM(1,A)

          FG(24)    FG24
                    AM(2,A)
.EJECT
```

```
          FRILL     TILL

          AM(2)     CHA(1)
                    CHA(2)
                    CHA(3)
                    CHA(4)
                    CD(1)

          AMP(1)    RM(1,B)
                    RM(2,B)
                    REV(2)
                    AM(1,B)
                    AM(2,B)
                    FF(1)
                    FF(2)

          AMP(2)    RM(1,B)
                    RM(2,B)
                    REV(2)
                    AM(1,B)
                    AM(2,B)
                    FF(1)
                    FF(2)
.EJECT
```

APPENDIX 2.

Samtliga möjliga felutskrifter i EMS-1, EMS-2 och EMS-3.

02	011	ILLEGAL DELIMITER
02	012	ILLEGAL PARAMETER
02	013	ILLEGAL MNEMONIC
02	014	ILLEGAL NUMBER OF PARAMETERS
02	021	ILLEGAL FREQUENCY GENERATOR NUMBER
02	022	ILLEGAL FREQUENCY GENERATOR FREQUENCY
02	023	ILLEGAL FREQUENCY GENERATOR WAVEFORM
02	024	ILLEGAL FREQUENCY GENERATOR INTENSITY
02	032	ILLEGAL FREQUENCY FILTER CHANNEL
02	033	ILLEGAL FREQUENCY FILTER INTENSITY
02	041	ILLEGAL AMPLIFIER NUMBER
02	042	ILLEGAL AMPLIFIER INTENSITY
02	051	ILLEGAL (DIS-) CONNECTION
02	062	ILLEGAL REVERBATION TIME
02	063	ILLEGAL REVERBATION INTENSITY
02	070	TIME MISSING IN D-TERM
02	071	ILLEGAL TIME
02	072	TIME OVERFLOW.
02	073	TEMP DISK OVERFLOW. NOTHING MORE WRITTEN ON THE DISK.
02	081	ILLEGAL NOISE COLOUR
02	082	ILLEGAL NOISE INTENSITY
02	091	ILLEGAL AMPLITUDE MODULATOR NUMBER
02	092	ILLEGAL AMPLITUDE MODULATOR ENTRY
02	093	ILLEGAL AMPLITUDE MODULATOR INTENSITY
02	102	ILLEGAL RING MODULATOR ENTRY
02	103	ILLEGAL RING MODULATOR INTENSITY
02	112	ILLEGAL CHANNEL DISTRIBUTOR INTENSITY
02	121	ILLEGAL DEVICE NUMBER
02	150	ILLEGAL ANALOG TAPE INTENSITY
02	142	ILLEGAL CHANNEL INTENSITY
02	201	ILLEGAL ESTEP VALUE
02	241	ILLEGAL ENV SYNTAX
02	242	ILLEGAL ENV AMPLITUDE
02	245	ENV OR GLISTIME LESS THAN STEP
02	246	ILLEGAL ENV TIME
02	247	ILLEGAL ENV TYPE
02	251	NO DEVICE TO ENVELOPE GIVEN
02	261	ILLEGAL LOCAL TIME VALUE
02	271	ILLEGAL GSTEP VALUE
02	301	NO DEVICE TO GLISSANDO GIVEN
02	302	ILLEGAL GLIS SYNTAX
02	303	ILLEGAL GLIS TIME
02	304	ILLEGAL GLIS TYPE
02	305	ILLEGAL GLIS FREQUENCY
02	142	ILLEGAL CHANNEL INTENSITY
04	001	SYMBOL WITH MORE THAN 6 CHARACTERS.
04	002	INTEGER WITH MORE THAN 5 FIGURES
04	003	TWO DECIMAL POINTS
04	004	A DECIMAL POINT (.) MUST NOT BE USED AS A DELIMITER.
04	010	IN THE EXPRESSION 'A=B' B IS NOT A DEFINED SYMBOL.
04	020	SYMBOL NOT DEFINED.
04	021	SYMBOL DEFINED BUT NOT POSSIBLE IN THIS PART OF THE TEXT.
04	022	DELETION OF MACROS NOT PERMITTED IN MACROS
04	031	TOP NOT EXECUTED IF READ FROM SECONDARY TEXT INPUT
04	032	UNIT 2 NOT FILEORIENTED
04	033	IN,SKIP,DELETE,LOOK NOT EXECUTED IF READ FROM SEC. TEXT INPUT
04	034	FILE NOT PRESENT ON UNIT 2

```
04   035        NO TOP EXECUTED OR SEC. FILE EMPTY
04   036        PARAMETER NOT INTEGER
04   037        ILLEGAL DELIMITER. SHOULD BE ( ; OR CARR. RET.
04   050        LABEL ERROR
04   064        : / ^ OR ) NOT POSSIBLE AS FIRST ELEMENT OF AN EXPR.
04   074        OVERFLOW. ADD OR SUBTRACT
04   075        OVERFLOW. MULTIPLICATION
04   076        TRY TO DIVIDE BY ZERO
04   100        A '=' IS POSSIBLE ONLY AFTER A SYMBOL OR A '=',
04   101        ATTEMPT TO ASSIGN A FIXED SYMBOL
                OR A SYMBOL STARTING WITH A NONALPHABETIC CHAR.
04   102        SYMBOL TABLE FULL
04   104        A ' " '  FOUND IN A NON-MACRO TEXT.
04   103        A MACRO MUST NOT BE CALLED IN ERROR MODE.
04   105        DECIMAL POINT (.) USED ILLEGALLY
04   106        NESTING TOO DEEP. RETURN TO LEVEL 0.
05   001        END OF INPUT FILE WITHOUT AN 'EXIT'
                INPUT FROM UNIT 4.
06   001        SECOND LABEL NOT ON THE MT.
06   002        FIRST LABEL NOT ON THE MT.
09   001        NO EXTENSION TO FILENAME
09   002        INPUT FILE FILEORIENTED. FILENAME ?
09   003        OUTPUT FILE FILEORIENTED. FILENAME ?
09   004        FILE NOT FOUND.
09   005        UNIT 1 NOT FILEORIENTED.
09   006        FILE FMNEMO EMS NOT FOUND
09   007        FILE ERROR EMS NOT FOUND.
09   008        NO FILENAME ON FILEORIENTED DEVICE.
99    99        --------- EXIT. END OF THIS RUN.---------
01   001        NOT A PERMANENT SYMBOL
01   002        NOT A COMMAND
01   003        ILLEGAL PARAMETER IN 'PLAY' TERM
01   004        NOTHING TO PLAY
01   005        TIME PARAMETER MUST BE INTEGER.
01   006        TIME PARAMETER NEGATIVE.
01   007        THE ONLY POSSIBLE PARAMETERS TO 'CLEAR' ARE 'MIX' OR 'ALL'
01   009        THE ONLY POSSIBLE PARAMETERS TO 'KEEP' ARE 'MIX' OR 'ALL'
01   010        KEEP WITHOUT MEANING HERE
01   015        SAVE,ERASE,REPL NOT EXECUTED IF READ FROM SEC. TEXT INPUT
01   016        LEFT PARANTHESIS MISSING AFTER SYMBOL
01   017        , AFTER SYMBOL
01   018        'END' AFTER PREVIOUS 'PART' MISSING
01   019        NO SORTRECORDS ON THE 'MIX' DISK
01   020        PART COMMAND FIRST THING IN A BLOCK!
01   021        'PART' AFTER PREVIOUS 'END' MISSING
01   024        LEFT PARANTHESIS MISSING IN 'STDTIM'-TERM
01   025        ERRONEUS TIMEPARAMETER IN 'STDTIM'-TERM
01   030        FILE ALREADY PRESENT
01   031        FILE NOT PRESENT ON UNIT 6.
01   037        ILLEGAL DELIMITER. SHOULD BE ( ; OR CARR. RET.
01   040        ERROR DETECTED IN "MERGE". NOT ENOUGH SPACE ON OUTPUT UNIT
01   041        DURATION OF MIX =0!
01   042        NO RECORDS ON THE TEMP DISK
01   043        CODEGENERATION ERROR. CODE ON THE MT WILL NOT BE SAVED
                MIX AND TEMP DISKS ARE SAVED.
01   050        LABEL ERROR
01   080        GLOBAL TIME OVERFLOW. GLOBAL TIME SET TO 0.
```

```
05    002        END OF FILE ON ADDITIONAL INPUT FILE.
10    101        FREQUENCY LESS THAN 0. PROGRAM SETS FREQ. TO 0.
10    102        FREQUENCY GREATER THAN 15999. PROGRAM SETS FREQ. TO 0.
10    103        INTENSITY LESS THAN 0. PROGRAM SETS INTENSITY TO 0.
10    104        INTENSITY GREATER THAN 120 DB. PROGRAM SETS INENSITY TO 0.
10    201        ADDRESS ERROR IN SORTRECORD. SORTRECORD IGNORED.
63    001        .OTSER ERROR
15    254        UNCHECKED ERROR IN DECODE PACKAGE.
32    010        BAD ARGUMENT IN IFMT CALL (<0). REPORT TO SYSTEM MANAGER.
98    008        BUFFER OVERFLOW ON .READ
99    002        EMS1 V1.4A  73.02.21

99    004        NAME OF INPUT FILE?
99    005        UNIT 3 NOT FILEORIENTED.
99    010        NOT ENOUGH FREE CORE. MUST BE AT LEAST 2000 (DEC.)
99    006        INTERACTIVE MODE? YES OR NO?
99    007
99    000
64    000         NONE SPECIFIED ERROR.
```

APPENDIX 3.

TABELL OEVER KVARTSTONER

C		CISS		D	
16.35	16.83	17.32	17.83	18.35	18.89
32.70	33.66	34.65	35.66	36.71	37.78
65.41	67.32	69.30	71.33	73.42	75.57
130.81	134.65	138.59	142.65	146.83	151.13
261.63	269.29	277.18	285.30	293.66	302.27
523.25	538.58	554.37	570.61	587.33	604.54
1046.50	1077.17	1108.73	1141.22	1174.66	1209.08
2093.00	2154.33	2217.46	2282.44	2349.32	2418.16
4186.01	4308.67	4434.92	4564.88	4698.64	4836.32
8372.02	8617.34	8869.85	9129.75	9397.27	9672.64

DISS		E		F	
19.45	20.02	20.60	21.21	21.83	22.47
38.89	40.03	41.20	42.41	43.65	44.93
77.78	80.06	82.41	84.82	87.31	89.87
155.56	160.12	164.81	169.64	174.61	179.73
311.13	320.24	329.63	339.29	349.23	359.46
622.25	640.49	659.26	678.57	698.46	718.92
1244.51	1280.97	1318.51	1357.15	1396.91	1437.85
2489.02	2561.95	2637.02	2714.29	2793.83	2875.69
4978.03	5123.90	5274.04	5428.58	5587.65	5751.38
9956.07	10247.80	10548.08	10857.17	11175.31	11502.77

FISS		G		GISS	
23.12	23.80	24.50	25.22	25.96	26.72
46.25	47.60	49.00	50.44	51.91	53.43
92.50	95.21	98.00	100.87	103.83	106.87
185.00	190.42	196.00	201.74	207.65	213.74
369.99	380.84	392.00	403.48	415.30	427.47
739.99	761.67	783.99	806.96	830.61	854.95
1479.98	1523.34	1567.98	1613.93	1661.22	1709.90
2959.96	3046.69	3135.96	3227.85	3322.44	3419.79
5919.91	6093.38	6271.93	6455.71	6644.88	6839.59
11839.82	12186.76	12543.86	12911.42	13289.75	13679.17

A		AISS		H	
27.50	28.31	29.14	29.99	30.87	31.77
55.00	56.61	58.27	59.98	61.74	63.54
110.00	113.22	116.54	119.96	123.47	127.09
220.00	226.45	233.08	239.91	246.94	254.18
440.00	452.89	466.16	479.82	493.88	508.36
880.00	905.79	932.33	959.65	987.77	1016.71
1760.00	1811.57	1864.66	1919.29	1975.53	2033.42
3520.00	3623.14	3729.31	3838.59	3951.07	4066.84
7040.00	7246.29	7458.62	7677.18	7902.13	8133.68
14080.00	14492.58	14917.24	15354.35	15804.27	16267.37

APPENDIX 4.

The relation between modulation in % of the signal entering the B-entry in an amplitude modulator, and the level in dB of the modulated signal on the A-entry.

Modulation %	Level dB
100	100
90	99
80	98
70	97
60	95.5
50	94
40	92
30	89.5
25	88
20	86
15	83
10	80
5	74
1	60

2 times corresponds to 6 dB
5 " " " 14 "
10 " " " 20 "