MSX

YAMAHA

FM Music Macro

YRM-104

OWNER'S MANUAL

NIPPON GAKKI CO., LTD.
PRINTED IN JAPAN

Introduction

Thank you for purchasing the Yamaha FM Music Macro (YRM-104). This unit is an expanded BASIC system allowing the operations of the FM Music Synthesizer to be controlled by BASIC. BASIC programs can be created for FM sound generated performance and special effects as well as for the creation of graphics.

Features

* Up to four voices can be specified for simultaneous playback.

* Playback of eight parts.

* Functions as a rhythm box for the creation of individual rhythm patterns

* Volume and pitch can be changed during performances. This permits the creation of various special sound effects.

* The playback data can be output through MIDI terminals for playing other MIDI compatible instruments such as the Yamaha DX Synthesizers.

Use of this manual

This instruction manual is divided into four sections: basics, applications, reference, and appendices. The sample programs in the manual use MSX BASIC commands. Refer to the MSX BASIC manual for further details on this form of BASIC. Also refer to the instruction manual of the CX5M.

Basics This section describes the operating procedure of the FM Music Macro and precautions. It also deals with the procedure for playback using the FM Music Macro and the basics of voice synthesizing.

Applications

This section describes the way to get maximum enjoyment from your FM Music Macro.

Syntax This section describes the commands used with the FM Music Macro. They are arranged in alphabetic order.

Appendices

The table of commands is arranged for easy reference. In addition to this command table and the memory map, the use of data memory cartridges and special commands are explained here.

MSX is a trademark of Microsoft Corporation

Index

I Basics

I-1 Overview of the FM Music Macro

System Requirements

The following equipment is needed for use of the FM Music Macro:

* CX5M Music Computer (32 K RAM capacity) or MSX personal computer

* Video Monitor or Color TV

* Yamaha FM Sound Synthesizer Unit (SFG-01 built into CX5M, or SFK-01)

Optional equipment:

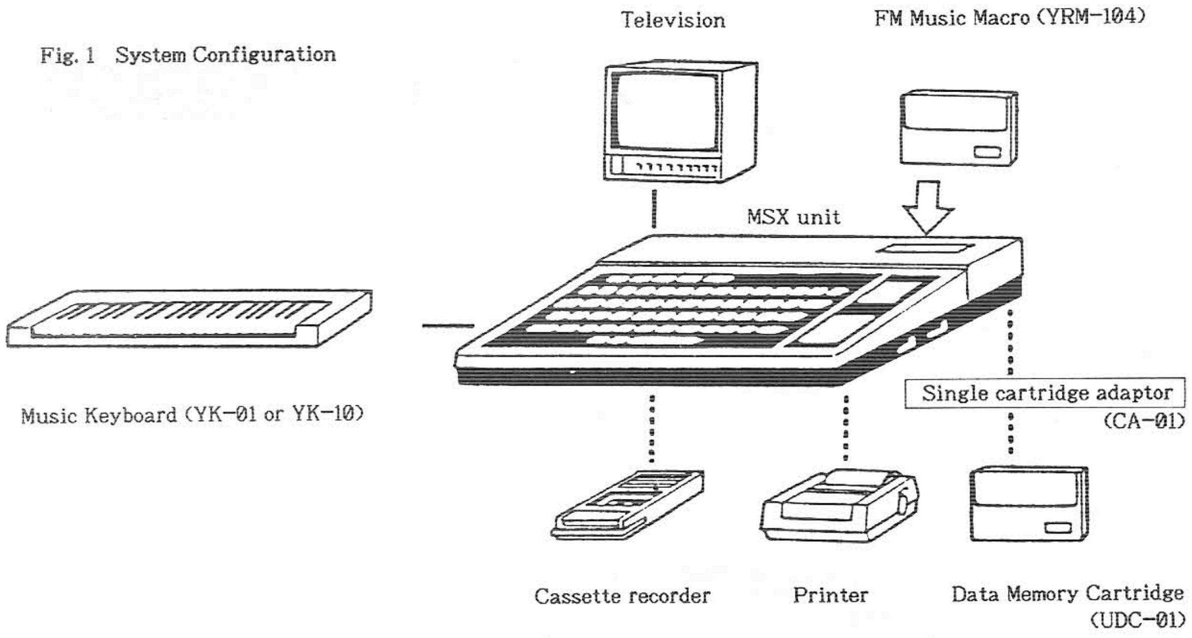
* Yamaha Music Keyboard (YK-01 or YK-10)

* Printer (use a compatible printer)

* Cassette recorder

* Yamaha Data Memory Cartridge (UDC-01)

Fig. 1 System Configuration


Television
FM Music Macro (YRM-104)
MSX unit
Music Keyboard (YK-01 or YK-10)
Single cartridge adaptor (CA-01)
Cassette recorder
Printer
Data Memory Cartridge (UDC-01)

Starting the FM Music Macro

1) Confirm that all of the equipment is correctly connected.

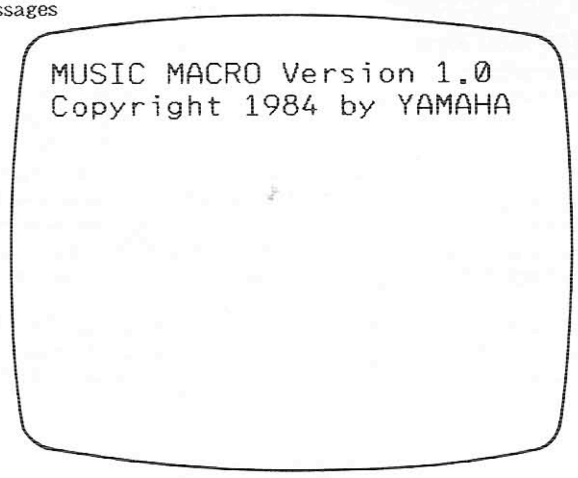
2) Check that the power is off.

3) Insert the ROM cartridge into the computer's cartridge slot.

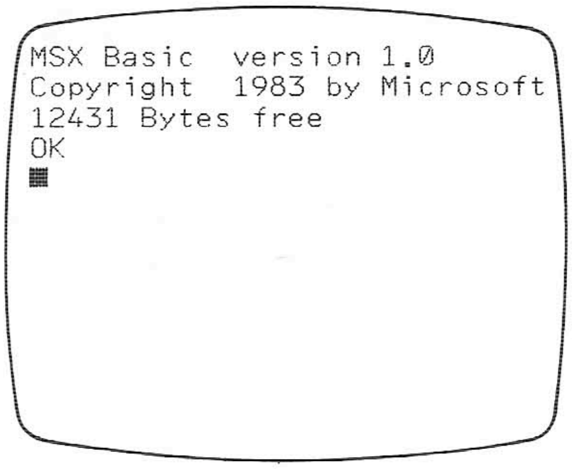
4) Turn the power on and the FM Music Macro will automatically start.

* The "MUSIC MACRO" message will appear as shown in Fig. 2. Then, the BASIC screen is normally displayed with the message "19727 Bytes ". If the program does not start as expected, turn the power of the CX5M unit off and check that the ROM cartridge is inserted correctly.

Fig. 2 Power-on Screen Messages



```
MUSIC MACRO Version 1.0
Copyright 1984 by YAMAHA
```



```
MSX Basic  version 1.0
Copyright  1983 by Microsoft
12431 Bytes free
OK
█
```

Before inputting programs

There is a special control program (MUSIC BIOS) incorporated into the FM Sound Synthesizer. The FM Music Macro uses an expanded form of MSX BASIC which allows easy programming of FM sound generation functions in BASIC. This system operates in the MUSIC BIOS system. The commands for the FM Music Macro are called from MSX BASIC through the use of the CALL command. This means that "CALL" or its abbreviated form "_" (underline) must be attached to the beginning of FM Music Macro commands. The abbreviated form of "_" (underline) is used throughout this manual. The FM Music Macro commands are only effective for the first four characters. Anything after that can be omitted.

(example)

```
CALL USERHYTHM

_USERHYTHM

_USER
```

Precautions when using the FM Music Macro

* When FM sound generation is used, the processing for sound generation control is added to the normal processing of the computer. Consequently, the processing speed for BASIC operations such as keyboard scan are slowed down. Also, the pitch time is set for a VDP interrupt every 1/60 second (1/50 second in a PAL system) and the values are updated at this time. However, during FM sound generation control there may be times when the interrupt cannot be received and thus timing will be slowed down. Certain precautions must be followed when commands for time are used. The FM Music Macro has its own built-in timer which should be used. The PLAY command in MSX Basic uses the interrupts generated by the VDP to calculate the length of sound generation, and care must be taken when using this command.

* Machine language programs should be input below CFFFH as the work area of the FM Music Macro is between D000H and F380H. When the upper limit of the memory is specified by the CLEAR command, the work area of the FM Music Macro will be damaged if an address higher than D000H is specified. Use the area higher than C000H for the stack area.

I-2 Playback procedure

Simple playback

Input the following program.

```
10 _INIT
20 _INST(1)
30 _PHRASE(1,"cdefg")
40 _PLAY(1,1)
```

Enter the RUN command. You should hear the sound of BRASS 1. If you don't, check the connection and the volume of the television (amplifier).

Here is an explanation of the above program:

STEP 1 Line 10 is a command for resetting the FM Music Macro to its initial "power on" state. It is used to prevent new programs from being affected by previously executed programs.

STEP 2 This line is used to specify which of the four FM Music Macro instruments to be played. This is done here by the _INST command. Instrument 1 is selected in this example.

STEP 3 After the above preparations, playback starts. When sounds generated in the CX5M are played, the playback contents are specified by the PLAY command. However, with the FM Music Macro this is divided into two steps: specifying the playback contents and then actual playback. Line 30 specifies the playback contents. This command writes the playback data into the memory for this data. This memory is refered to as a track. Like the tracks of a tape recorder, a total of eight tracks can be used. The number of tracks to be used is usually specified by the _TRACK command but this can be omitted in this program as there is only one track being used. Specification of the playback contents is done by first specifying the track onto which data is to be written, followed by specification of the playback data.

```
            PHRASE(1,"cdefg")
                   ↑    ↑
track specification    playback contents(playback list)
```

The playback contents are specified by a character string (pplayback list). "cdefg" stands for the following scale. Various forms of playback are possible with the use of these symbols.

Symbols	c	d	e	f	g	a	b
Scale	do	re	mi	fa	so	la	ti

STEP 4 Playback can take place after the data is written onto the track. The _PLAY command in line 40 starts playback. This command specifies the instrument and the track containing the data to be played. Both of these are 1 as this program uses only one instrument and one track.

```
_PLAY(1,1)
      ↑ ↑
      | └ track specification
      └── instrument specification
```

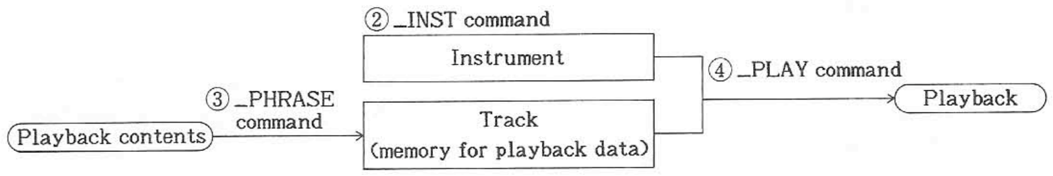
Summary of the playback procedure

① Initialization of system (_INIT command)

② Preparation of instruments (_INST command)

③ Writing of playback contents (_PHRASE command)

④ Beginning of playback (_PLAY command)



Various forms of playback

You should have a good idea by now of the basic playback procedure. Next, change the length of notes and attach sharps (#) and flats (b).

1. Length of notes

Substitute the following for line 30 and run the program again.

```
30 _PHRASE(1,"c1d2e4f8g16")
```

The length of the notes should become progressively shorter. The numbers after the letters c – g change the length of notes. 4 stands for a quarter note and 8 is used to indicate an eighth note. The notes become shorter as the value becomes larger. If continuous notes are going to be the same length, specifying numbers after each one is redundant, so the "L" symbol is used for this. Substitute the following for line 30 and run the program again.

```
30 _PHRASE(1,"l4cdel8fg")
```

The length of the note is determined by the number following the letter which indicates the scale of the note. However, if there is no number, the number following the "L" symbol determines the length. Check this by entering the following for line 30.

```
30 _PHRASE(1,"l4cde2l8fg")
```

Specification of the "L" symbol is effective until another value is specified for L. L is 4 when the power is turned on or when the unit is initialized by the _INIT command. Dotted notes will be explained next. Dotted notes are indicated by a "." (period) after the letter which specifies the scale of the note.

```
_PHRASE(1,"cd.e8.")
```

If there is a number following the letters, "a" – "g", dots must be located after the numbers.

2. #(Sharp) and b(Flat) Notation

The following notations are used to specify sharps and flats:

(sharp): the "#" symbol or the "+" symbol is input immediately after the note.

b (flat): the "−" symbol is input immediately after the note.

For example, to specify a C sharp with a length of 4, either of the following may be input:

```
_PHRASE(1,"c#4")

_PHRASE(1,"l4c#")
```

If the length of the note is specified for the individual note, the "#", "+" or "−" symbol is entered between the note specifier and the note length specifier.

3. Setting octaves

All notes up to now have been confined within a single octave. The "O" symbol is used for the output of higher and lower octaves. Substitute the following for line 30.

```
      _PHRASE(1,"o2cego3cego4cego5c")
```

The number after the "O" symbol sets the octave. This number can be between 0 and 8. o8 can only output a C note.

Specification of the "O" symbol is effective until the symbol is specified again. The value is o3 when the power is turned on or when the unit is initialized by the _INIT command.

The "<" and ">" symbols are used to raise and lower the octave by one. The "<" symbol raises the present octave value by one and ">" lowers it by one.

```
      _PHRASE(1,"o2ceg<ceg<ceg<c")
```

Next, lets use the above commands to create a song.

```
      10 _INIT
      20 _INST(1)
      30 READ A$
      40 IF A$="end" THEN 70
      50 _PHRASE(1,A$)
      60 GOTO 30
      70 _PLAY(1,1)
      80 DATA l8e.g16g4e.d16c4d.e16g.e16d2
      90 DATA l8e.g16g4e.d16c4ded.c16c2
      100 DATA end
```

4. Setting tempo

The tempo of the playback data can be changed by the _TEMPO command. Add the following line to slow down the tempo of the program.

```
25 _TEMPO(60)
```

The tempo should have become slower. The tempo is set by the value contained in parentheses ( ). The number is equivalent to the number of quarter notes per minute. The tempo will slow down as the value becomes larger. Change the tempo value in the above _TEMPO command and listen to the difference in tempo. The initial value set is _TEMPO (120).

The tempo of the playback contents of the character string (playback list) can also be set. This is useful for changing the tempo during playback. Change lines 25 and 90 of the above program to the following.

```
10 _INIT
20 _INST(1)
25 _TEMPO(60)
30 READ A$
40 IF A$="end" THEN 70
50 _PHRASE(1,A$)
60 GOTO 30
70 _PLAY(1,1)
80 DATA l8e.g16g4e.d16c4d.e16g.e16d2
90 DATA t150l8e.g16g4e.d16c4t50ded.c16c2
100 DATA end
```

The tempo set by the _TEMPO command can be changed by the use of the "T" symbol. The value after the "T" symbol specifies the percentage by which the tempo changes. In this example, the tempo will change by 150% (1.5 times) and by 50% (1/2). The tempo specification will be returned to 100% (1 time) of the tempo set by _TEMPO command after the playback of the character string set by the "T" symbol.

5. Changing the voice

The following procedure is used to change the voice. Add the following line to the program.

```
22 _MODI(1,4)
```

Run the program after the line has been input. Up to now, playback has used the BRASS 1 tone. This has now become STRING 1. The _MODINST command is used in this way to change the voice. This command changes not only the voice but also changes data for the instrument. Refer to the syntax section for details.

Line 22 has the values 1 and 4 in parentheses. The first value, "1" is the instrument number which is specified to change the instrument data. The second value, "4" is the voice number. The voice number is set in the range of 1 – 56. 1 – 48 are preset in the FM Sound Synthesizer unit. 49 – 56 are used for tones created using the FM Voicing Program (YRM–102). If voices 49 – 56 are to be used, the voice data must be loaded from the cassette recorder beforehand (refer to pg. 24 and pg. 52).There are a number of other things which can be set for the playback list. Refer to the section on the PHRASE command in the syntax section for further details.

Summary

Length of note	character for scale + number	only effective for that note
	"L" + number	effective until new specification
Setting # and b	"#" : note + "#" or "+"	
	"b" : note + "−"	
Setting octave	"O" + number	
Setting tempo	_TEMPO(<tempo value>): <tempo value> is 0 (stop) − 200 (fastest)	
Setting voice	_MODINST (<instrument number>,<voice number>): <voice number> is 1 − 56	

I-3 Use of rhythm

The FM Music Macro can add a rhythm instrument in addition to the four basic instruments. This allows for the easy addition of rhythm. Input the following program.

```
10 _INIT
20 _USERHYTHM
30 _SELP(1)
40 _RHYTHM(16)
```

Run the program and the playback should be 4 bars of 16 beats. Here is an explanation of the above program.

STEP 1 Line 10 is the _INIT command which resets the FM Music Macro.

STEP 2 Line 20 is the command for specifying the use of the rhythm instrument. This command must be input prior to the use of rhythm related commands. The number of tracks which can be used simultaneously drops from a maximum of eight to six.

STEP 3 The rhythm pattern is selected after the rhythm instrument is set. There are six preset rhythm patterns and two original patterns can also be defined. The following numbers are used to specify the pattern. Line 30 in this example selects a 16 beat rhythm with _SELP(1).

1 – 6 (Preset patterns)

1: 16 beat

2: slow rock

3: waltz

4: rock

5: disco

6: swing

7, 8: definable patterns

The procedure for defining patterns is given on page 38.

STEP 4 The above steps complete the preparations. The _RHYTHM command in line 40 is the rhythm command. This command specifies the number of times that the rhythm is to be repeated. The value specifies the quarter note part. For example, if for one bar the rhythm is to be 4/4, _RHYTHM (4) is specified. If it is 3/4, _RHYTHM (3) is specified. Here _RHYTHM (16) is entered for playback of four bars.

Summary of the rhythm playback procedure

(1) Initialization of the system (_INIT command)

(2) Preparation of the rhythm instrument (_USERHYTHM command)

(3) Select for rhythm patterns (_SELP command)

(4) Beginning of playback (_RHYTHM command)

I-4 Ensemble

A simple expression of the preceeding concepts will allow you to play two or more instruments simultaneously.

```
10 _INIT
20 _TRACK(2)
30 _TEMPO(135)
40 _INST(1)
50 _INST(2)
60 _MODI(1,16)
70 _MODI(2,19)
80 READ A$,B$
90 IF A$="end" THEN 130
100 _PHRASE(1,A$)
110 _PHRASE(2,B$)
120 GOTO 80
130 _STANDBY
140 _PLAY(1,1)
150 _PLAY(2,2)
160 _START
170 DATA c2eg>b.l16<cdc2
180 DATA l8cgegcgegdgfgcgeg
190 DATA a2l4g<c>gl24fgfl16efe2
200 DATA l8cafacgeg>b<gdgcgeg
210 DATA end,""
```

This program uses two tracks, which are designated in line 20. The number of track is specified by the value in parentheses. There are new commands in line 130 and 160, used to synchronize playback of the two instruments. The _STANDBY command temporarily stops playback. This means that playback will not start although there are _PLAY command following this command. The command can be compared to the function of the pause button on a tape recorder. The _START command releases the temporary suspension of playback operations. This allows playback of instrument 1 and instrument 2 to begin simultaneously. The following program includes the rhythm instrument.

```
10 _INIT
20 _TRACK(2)
30 _TEMPO(135)
40 _USERHYTHM
50 _INST(1)
60 _INST(2)
70 _MODI(1,16)
80 _MODI(2,19)
90 READ A$,B$
100 IF A$="end" THEN 140
110 _PHRASE(1,A$)
120 _PHRASE(2,B$)
130 GOTO 90
140 _STANDBY
150 _PLAY(1,1)
160 _PLAY(2,2)
170 _SELP(1)
180 _RHYTHM(16)
190 _START
200 DATA c2eg>b.l16<cdc2
210 DATA l8cgegcgegdgfgcgeg
220 DATA a2l4g<c>gl24fgfl16efe2
230 DATA l8cafacgeg>b<gdgcgeg
240 DATA end,""
```

I-4 Ensemble

A simple expression of the preceeding concepts will allow you to play two or more instruments simultaneously.

```
10 _INIT
20 _TRACK(2)
30 _TEMPO(135)
40 _INST(1)
50 _INST(2)
60 _MODI(1,16)
70 _MODI(2,19)
80 READ A$,B$
90 IF A$="end" THEN 130
100 _PHRASE(1,A$)
110 _PHRASE(2,B$)
120 GOTO 80
130 _STANDBY
140 _PLAY(1,1)
150 _PLAY(2,2)
160 _START
170 DATA c2eg>b.l16<cdc2
180 DATA l8cgegcgegdgfgcgeg
190 DATA a2l4g<c>gl24fgfl16efe2
200 DATA l8cafacgeg>b<gdgcgeg
210 DATA end,""
```

This program uses two tracks, which are designated in line 20. The number of track is specified by the value in parentheses. There are new commands in line 130 and 160, used to synchronize playback of the two instruments. The _STANDBY command temporarily stops playback. This means that playback will not start although there are _PLAY command following this command. The command can be compared to the function of the pause button on a tape recorder. The _START command releases the temporary suspension of playback operations. This allows playback of instrument 1 and instrument 2 to begin simultaneously. The following program includes the rhythm instrument.

```
10 _INIT
20 _TRACK(2)
30 _TEMPO(135)
40 _USERHYTHM
50 _INST(1)
60 _INST(2)
70 _MODI(1,16)
80 _MODI(2,19)
90 READ A$,B$
100 IF A$="end" THEN 140
110 _PHRASE(1,A$)
120 _PHRASE(2,B$)
130 GOTO 90
140 _STANDBY
150 _PLAY(1,1)
160 _PLAY(2,2)
170 _SELP(1)
180 _RHYTHM(16)
190 _START
200 DATA c2eg>b.l16<cdc2
210 DATA l8cgegcgegdgfgcgeg
220 DATA a2l4g<c>gl24fgfl16efe2
230 DATA l8cafacgeg>b<gdgcgeg
240 DATA end,""
```

Summary of ensemble playback

(1) Temporary suspension of playback (_STANDBY command)

(2) Beginning of playback for each instrument (_PLAY and _RHYTHM commands)

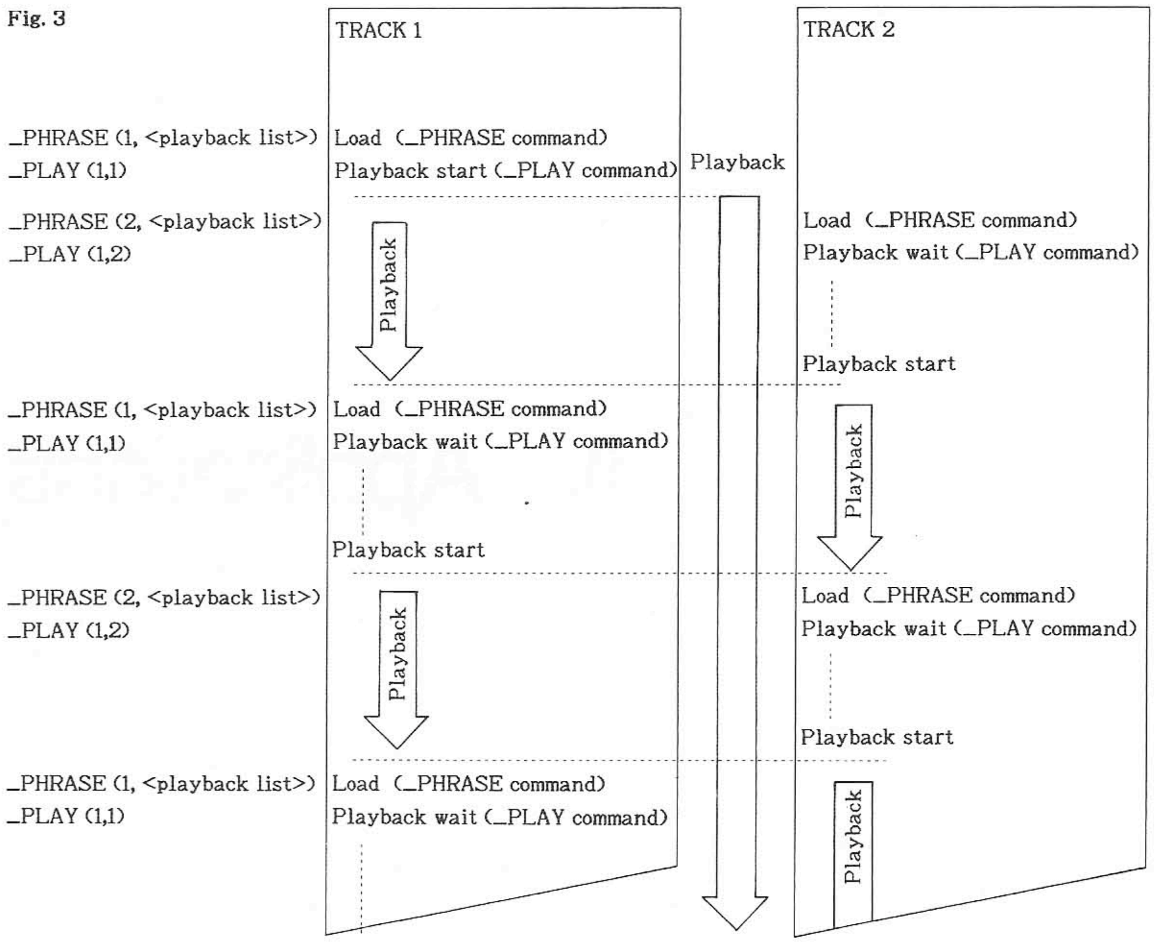
(3) Release of temporary suspension and start of actual playback (_START command)

Ⅱ Applications

II-1 Long Playback

The maximum size for tracks is 4Kb/TRACK. This is equivalent to playback data of approximately 500 notes. Thus with long programs, loading all the playback data into a track, as shown on pg. 9 , will exceed the memory capacity and make playback impossible. In this case, the contents of a track must be rewritten several times. The problem with simply dividing the program is the time lag while loading a new section after playing a previous one. This is prevented by reserving two tracks for one instrument. While one track is playing, playback data is being loaded into the other track.

Fig. 3


TRACK 1
TRACK 2
_PHRASE (1, <playback list>)
_PLAY (1,1)
Load (_PHRASE command)
Playback start (_PLAY command)
Playback
_PHRASE (2, <playback list>)
_PLAY (1,2)
Playback
Load (_PHRASE command)
Playback wait (_PLAY command)
Playback start
_PHRASE (1, <playback list>)
_PLAY (1,1)
Load (_PHRASE command)
Playback wait (_PLAY command)
Playback
Playback start
_PHRASE (2, <playback list>)
_PLAY (1,2)
Playback
Load (_PHRASE command)
Playback wait (_PLAY command)
Playback start
_PHRASE (1, <playback list>)
_PLAY (1,1)
Load (_PHRASE command)
Playback wait (_PLAY command)
Playback

When the _PLAY command is executed, playback will enter the wait mode for the specified instrument if the same instrument is being played. Playback will start at the instant current playback of the instrument is completed. This technique allows the use of the entire BASIC program area and alleviates worries about the size of the tracks.

New data cannot be loaded into a track while that track is being played. Also, the _PLAY command cannot be used to start playback of instruments already in the playback wait mode. This means that the loading of playback data can only begin after the previous playback operations are completed. There are two ways of confirming the completion of playback of one section. One way is with the use of the _WAIT command, and the other is with the _ON EVENT ( ) GOSUB command. The following explains the use of both procedures.

Use of the _WAIT command

The _WAIT command is a command which suspends execution of a program until completion of playback of a specified instrument or rhythm. This command allows for easy confirmation of playback completion. In Figure 3, the _WAIT command can be inserted before the _PHRASE command after the third repetition. The following is an example of an actual program.

```
10 _INIT
20 _TRACK(2)
30 _INST(1)
40 _TEMPO(140)
50 _MODI(1,16)
60 T=N AND 1:T=T+1
70 N=N+1
80 READ A$
90 IF A$="end" THEN END
100 _ERASE(T)
110 PRINT"TRACK";T;"TO WRITE IN"
120 _PHRASE(T,A$)
130 _PLAY(1,T)
140 IF N>1 THEN PRINT"WAITING":_WAIT(1)
150 GOTO 60
160 DATA K2l8br8<er16d16cgf>b<edc>b<d4cr8
170 DATA k2l8br8<er16d16cgfbgfedcd>b<l16dd
180 DATA k2l16o4cddd>b<dgbbe!f#aadagfcfee>b<ed>b<d>a<d>agdd
190 DATA k2l16cdddcdfggde!ggcgfe!>b<e!dd>a<dccd>b<cr16o4fff
200 DATA k2l16o4ed-c>ba<ff<fed-c>bafffed-c>ba<ff<ffed-c>bafff
210 DATA k2o4l16cfffd-g-g-g-dgggea-a-a-e!afbf#b!g<c>g#<c#>a<d>b<ecf
230 DATA K2l8br8<er16d16cgf>b<edc>b<d4cr8
240 DATA k2l8br8<er16d16cgfbgfedcd>b<r8
250 DATA end
```

The command in line 100 clears the track. The variable T is the track number. The value of T alternates between 1 and 2.

Use of the _ON EVENT (n) GOSUB command

The FM Music Macro is capable of generating interrupts upon the completion of playback etc. The _ON EVENT ( ) GOSUB command allows the execution of a subroutine when the program is interrupted. A program for rewriting playback data can be contained in the subroutine. With the _WAIT command, other processing can take place while the execution of the program is being interrupted. This can be, for example, the drawing of pictures on the screen during playback.

Sample Program

```
10 _INIT
20 _ON EVENT(1) GOSUB 180
30 _EVENT(1) ON
40 _TRACK(2)
50 _INST(1)
60 _TEMPO(140)
70 _MODI(1,16)
80 FOR I=1 TO 2
90 READ A$
100 _PHRASE(I,A$)
110 _PLAY(1,I)
120 NEXT I
130 PRINT"--FM MUSIC MACRO-- "
135 IF X=1 THEN END
140 GOTO 130
150 '
160 '-- trap routine ---
170 '
180 T=N AND 1:T=T+1:N=N+1
190 READ A$
200 IF A$="end" THEN X=1:RETURN
210 _ERASE(T)
220 _PHRASE(T,A$)
230 _PLAY(1,T)
240 RETURN
250 '
260 '
270 DATA t127K2l8br8<er16d16cgf>b<edc>b<d4cr8
280 DATA k2l8br8<er16d16cgfbgfedcd>b<l16dd
290 DATA k2l16o4cddd>b<dgbbe!f#aadagfcfee>b<ed>b<d>a<d>agdd
300 DATA k2l16cdddcdfggde!ggcgfe!>b<e!dd>a<dccd>b<cr16o4fff
310 DATA k2l16o4ed-c>ba<ff<fed-c>bafffed-c>ba<ff<ffed-c>bafff
320 DATA k2o4l16cfffd-g-g-g-dgggea-a-a-e!afbf#b!g<c>g#<c#>a<d>b<ecf
330 DATA K2l8br8<er16d16cgf>b<edc>b<d4cr8
340 DATA k2l8br8<er16d16cgfbgfedcd>b<r8
350 DATA end
360 '
```

This program divides the piece into four sections. There is no need for playback completion to be checked up to the second phrase (refer to Fig. 3). The _PHRASE and _PLAY commands are repeated twice in a manner similar to the _WAIT command (lines 80 - 120). The next line begins an infinite loop which prevents further progression. This does not rewrite playback data but looks instead at lines 20 and 30. These two commands cause the program to jump to the subroutine in lines 180-240 when playback is completed. This subroutine rewrites playback data.

The RETURN command at the end of the subroutine returns the program to the original routine which is to repeat the display. This program continues playback until there is no more playback data. Then, the program is ended by lines 135-200.

The parentheses in the _ON EVENT ( ) GOSUB in line 20 contain 1. This value determines the instrument whose playback completion generates an interrupt. In this program there is only one instrument and therefore 1 is the value in the command. The _EVENT ON command in line 30 permits the generation of an interrupt. The execution of this command generates interrupts at the completion of playback and for the program to go to the subroutine specified in the _ON EVENT (n) GOSUB command. The value in parenthesis, in this case 1, is the same instrument number that was specified in the _ON EVENT (n) GOSUB.

Events

The _WAIT command and _ON EVENT (n) GOSUB commands allow you to advance the program to the next line or interrupt the program when certain special conditions occur. These certain special conditions are referred to in this manual as "events". The following events are specified by their respective numbers.

1. Playback completion of instrument 1

2. Playback completion of instrument 2

3. Playback completion of instrument 3

4. Playback completion of instrument 4

5. Completion of rhythm playback

6. The time set in the internal timer of the FM Music Macro

II-2 Marks

Normally, playback of the playback data loaded into tracks is started from the beginning of the track. However, the track can be divided into sections and these sections can be played in any order desired. This is done by attaching numbers to the data (called marks) when it is loaded into the track by the _PHRASE command. Input the following program. When the program is run, the INPUT command will ask for a number. If 1 is input a CEG chord will be output; if 2 the chord will be CFA; and 3 will cause the output of a BDG chord.

```
10 _INIT
20 _INST(1,3)
30 READ M,A$
40 IF M=0 THEN 70
50 _PHRASE(1,A$,M)
60 GOTO 30
70 X$=INKEY$:IF X$="" THEN 70
80 X=VAL(X$)
90 IF X>3 OR X<1 THEN 70
100 _PLAY(1,1,X)
110 _WAIT(1)
120 GOTO 70
130 DATA 1,L8[ceg],2,L8[cfa],3,L8[o2b<dg],0,""
```

This program loads the three chords (CEG, CFA, and BDG) into one track. Playback is by the specification of one of these. The letter (variable M) after the playback list in the _PHRASE command of line 50 is the mark. In this example the numbers are as follows: 1 is (ceg), 2 is (cfa), and 3 is (bdg). These numbers need not be consecutive order. They can be chosen from between 1 and 254.

The way the tracks are divided by the marks is shown in the following diagram.

Mark 1	Mark 2	Mark 3
〔ceg〕	〔cfa〕	〔o2b< dg〕

In line 100, the _PLAY command specifies which part is to be played by writing a mark number (variable X) after the instrument number and the track number.

If the mark number is omitted, a mark number equal in value to the track number will be automatically read.

Ⅲ Syntax

In this section, all the commands for the FM Music Macro program are presented in alphabetical order in the following format:

Function a brief description of the function of the command

Format the format of the command. The following rules apply when commands are input.

* Only the first four characters of the commands are interpreted.

* CALL or its abbreviation "_" (underline) is attached to the beginning of all commands.

(example)

```
CALL INIT
_INIT
```

* Items expressed in upper case letters are input as such. They can be input as lower or upper case letters.

* Items enclosed in < > are specified by the user.

* Items enclosed in ( ) can be omitted. If they are omitted, the initial value or the value set before that is used.

* The following format is used for omitting parameters when the command has multiple parameters delimited by commas (",").

```
_INST(<instrument number>(,<voiced number>)(,<MIDI>)(,<MIDI channel>))
```

The parameters enclosed in ( ) can be omitted. If the rest of the parameters following the omitted parameter are also omitted, the commas need not be input. However, if subsequent parameters are specified, they must be preceeded by all of the commas. For example, if only <voiced number> is omitted from the above command, the new format will be:

```
_INST(1,,2,3)
```

A _INIT command or similar command having no specified items must be followed by a colon (:) when used directly before the "ELSE" of a MSX BASIC IF – THEN – ELSE command.

For example: `100 IF X=1 THEN _INIT : ELSE X=1`

Example an actual example of the use of the command

Explanation use of the command, detailed functions, and precautions

Sample Program a short program using the command

CANCEL

Function	cancels an instrument
Format	_CANCel ((<instrument number>))
Example	_CANC(1)

Explanation This command releases the sound generator from the instrument allocated to it by the _INST command. There can be a maximum number of eight voices for the instruments of the FM Music Macro. Instrument 2 cannot be defined if eight voices have already been defined for Instrument 1. In this case, the _CANCEL command is used to release the sound generator from instrument 1, permitting the voices to be re-allocated.

The <instrument number> is between 1 and 4. If it is 0 or omitted, the command applies to all instruments. The instruments will then be initialized to the following:

Tone:	BRASS 1
Transpose:	none
Volume	100 (maximum)
Portamento:	finger portamento
Portamento speed:	portamento off
Sustain:	off
Trigger mode:	multi

Refer to INST

CLDVOICE

Function loads voice data into the memory

Format _CLDVoice(([<option>][,<switch>]))

Example _CLDV(1)

Explanation This command loads voice data created with the FM Voicing Program, from cassette tape or from a data memory cartridge.

The <option> is used for specifying whether or not the file name or voice name will be displayed when the voice data is loaded. The value will be 0 if omitted.

0: file name or voice name is not displayed

1: file name or voice name is displayed

This command will load the first voice data it finds when it is executed. If loaded data is to be used, the _SELVOICE command must be used to regitster the voice.

The <switch> is to select whether data is going to be loaded from a cassette recorder or data memory cartridge. The cassette recorder is selected if this value is omitted.

0: cassette recorder

1: data memory cartridge

The _MCKS command is used to save data onto data memory cartridges (refer to page 72).

Refer to SELVOICE

ERASE

Function clears the contents of the specified track

Format _ERASe(<track number>)

Example _ERAS(2)

Explanation This command clears the contents of the track specified by the <track number>. The _ERASE command is used to erase the old contents of a track when that track is to be rewritten.

If the specified track is being played, the _ERASE command will not be executed and a "Device I/O error" will occur. Execute this command only when the specified track is not being played.

The <track number> can be set from 1 to the number specified by the _TRACK command.

Sample program

```
10 ' _ERASE sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 PRINT"WRITE IN cdefg and PLAY":PRINT
70 _PHRASE(1,"cdefg")
80 _PLAY(1,1)
90 _WAIT(1)
100 PRINT"WRITE IN gfedc and PLAY":PRINT
110 _PHRASE(1,"gfedc")
120 _PLAY(1,1)
130 _WAIT(1)
140 PRINT"AFTER ERASE"
150 PRINT"WRITE IN gfedc and PLAY"
160 _ERASE(1)
170 _PHRASE(1,"gfedc")
180 _PLAY(1,1)
```

EVENT (n) ON/OFF/STOP

Function permits, prohibits, and holds the generation of interrupts in the program

Format _EVENt([<event number>]) ON

_EVENt([<event number>]) OFF

_EVENt([<event number>]) STOP

Example _EVEN(1) ON

Explanation This command permits, prohibits, or holds the interrupt specified in the _ON EVENT(n) GOSUB.

The following are the types of <event numbers>.

1-4: interrupt upon the completion of playback started by the _PLAY command. 1 – 4 are the values corresponding to the instrument numbers.

5: interrupt upon completion of rhythm playback

6: interrupt based upon the timer of the FM Sound Synthesizer unit

If the <event number> is omitted, the command applies to all events.

(example) _EVENT ( ) ON

_EVEN (n) ON permits the generation of interrupts. The input of this command permits the generation of an interrupt upon the execution of the specified event. The program will be interrupted during execution and will go to the subroutine specified by the _ON EVENT (n) GOSUB command.

_EVENT (n) OFF prohibits the generation of interrupts. The input of the command prevents interrupt upon the execution of the specified event.

_EVENT (n) STOP holds interrupt operations. The execution of this command holds the generation of the interrupt until the execution of the _EVENT (n) ON command. The execution of the _EVENT (n) ON command causes the program to jump immediately to the subroutine specified by the _ON EVENT (n) GOSUB command. The _EVENT (n) STOP command is only executed during the _EVENT (n) ON command mode.

Refer to ON EVENT (n) GOSUB

INIT

Function initializes the FM Music Macro

Format _INIT

Example _INIT

Explanation This is the condition when the power of the FM Music Macro is turned on. The following settings are the initial values:

Track number:	1 (contents cleared)
Tempo:	120
Registered tone data:	cleared
Transpose:	none
Tune:	0
Tone:	100
Registered rhythm pattern:	cleared
Rhythm pattern:	1

(Contents of the instruments)

Tone:	BRASS 1
Transpose:	none
Volume:	100 (maximum)
Portamento:	fingered portamento
Portamento speed:	portamento off
Sustain:	off
Trigger mode:	multi

INMKEY

Function checks if the keys on the Music Keyboard are being pressed

Format _INMKey (<variable>)

Example _INMK(A)

Explanation The value of the specified variable is 0 when the keys of the Music Keyboard are not being pressed. The value of the variable corresponds to the key codes when the keys are pressed.
The program does not stop at places where there is a _INMKEY command. The following example is used to allow key input.

(example)
100 _INMK (A): IF A=0 THEN 100

The codes corresponding to the keys are the following.

```
YK-01 lowest note   F…41
                    |
      highest note  C…84
                    |
YK-10 lowest note   C…36
                    |
      highest note  C…84
```

Sample program

```
10 '_INMKEY sample
20 '
30 DIM N$(11)
40 _INIT
50 CLS
60 FOR I=0 TO 11
70 READ N$(I)
80 NEXT I
90 _INST(1)
100 _PLAY(1,9)
110 _INMK(A)
120 Q$=INKEY$:IF Q$<>"" THEN 160
130 IF A=0 THEN 110
140 PRINT N$(A MOD 12);
150 GOTO 110
160 _STOP(1):END
170 DATA DOH, DOH#, RE, RE#, MI, FAH, FAH#, SOH, SOH#, LAH, LAH#, TI
```

INST

Function defines the instruments to be used by the FM Music Macro

Format _INST(<instrument number>(,<number of voices>)(,<MIDI>)(,<MIDI channel>))

Example _INST (1,4)

Explanation Up to four instruments can be freely used by the FM Music Macro. The _INST statement specifies the output and number of voices of each instrument. The instruments must be defined to be used.

<instrument number> Input a number between 1 and 4.

<number of voices> This specifies the maximum number of voices between 1 and 8 which can be played back simultaneously. An error will occur if the total number of voices exceeds 8 when more than one instrument is used. The value is 1 if specifier is omitted.

<MIDI> This sets whether or not data will be output to the MIDI channel.

1 MIDI OFF (when omitted)

2 MIDI ON

<MIDI channel> This sets the MIDI transmission channel for playback data. The setting range is 1 – 16. The value is 1 if specification is omitted.

LENGTH

Function checks the length of the playback data in the track

Format _LENGth([<variable 1>][,<variable 2>][,<variable 3>]...[,<variable 8>])

Example _LENG(T1, T2, T3, T4)

Explanation The _PHRASE command checks the length of the playback data loaded into the track. <variable 1> – <variable 8> correspond to tracks 1 through 8. The variables return the total length of the phrases loaded into the tracks as numeric values. The units of the numeric values are equivalent to 1/192 of a full note. For example, if the numeric value is 48, the length of the phrase will be 1/4 (48 X (1/192)) or the length of a quarter note.

This _LENGTH command is used for such applications as checking that the length of the various phrases which are being played simultaneously are the same during extended playback which changes between tracks.

The variable determined in the _LENGTH command is reset by the execution of the _INIT and _ERASE commands and by changing the mark numbers in a track.

Refer to PHRASE

Sample program

```
10 '_LENGTH sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 _PHRASE(1,"cdefgab<c")
70 _LENGTH(A)
80 PRINT(A*1/192);"BAR"
90 _PLAY(1,1)
```

LFO

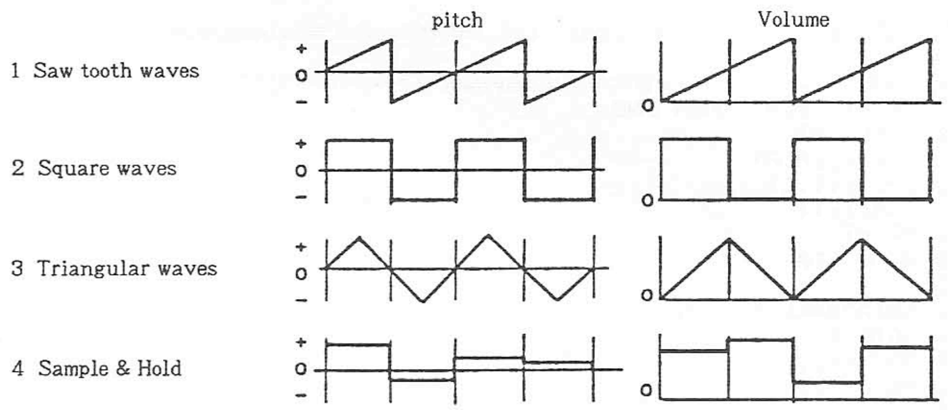
Function changes the LFO data

Format _LFO(<waveform number>(,<speed>)(,<tremolo>) (,<vibrato>))

Example _LFO (3, 50, 60, 20)

Explanation The LFO (Low Frequency Oscillator) is an extremely low frequency oscillator used for the production of tremolo and vibrato effects.

There are a number of parameters which are set for each tone. These are changed by the _LFO command.



Sample & Hold outputs random values.

The <waveform number> specifies the waveform output by the LFO (by altering pitch and volume). The range of this parameter is 1 – 4. The corresponding waveforms are shown in the above diagrams.

The <speed> specifies the speed (frequency) of the LFO in relation to the volume. The setting range is 0 – 100. The larger the number specified, the higher the frequency and thus, the faster the speed.

The <tremolo> is the setting of the modulation in relation to volume. The setting range is 0 – 100. The larger the number specified, the more the volume will change.

The <vibrato> is the setting of the modulation in relation to pitch. The setting range is 0 – 100. The larger the number specified, the more the pitch changes.

In addition to the <tremolo> and <vibrato>, the sensitivity of the LFO can also be set for each tone. There are some tones which are not affected by the LFO. The _MODI command can be used to set the sensitivity.

* If an instrument is selected after the LFO is set, the LFO data of the newly specified tone will apply. There will be no change with the SFG-01 or SFK-01 is voices having the following preset voice numbers are selected.

Voice numbers 31, 37 – 40, 42 – 46, 80

Refer to MODINST

Sample program

```
10 '_LFO sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 _PHRASE(1,"l2cdefgab<c")
70 _PLAY(1,1)
80 _WAIT(1)
90 _MODI(1,,,,,,,,,100,100)
100 '
110 READ A$,M,P
120 IF A$="end" THEN END
130 PRINT A$
140 _LFO(,,M,P)
150 _PLAY(1,1)
160 _WAIT(1)
170 PRINT
180 GOTO 110
190 '
200 DATA"MAKE VIBRATO LOUD"
210 DATA 0,100
220 DATA"ELIMINATE VIBRATO"
230 DATA 0,0
240 DATA"MAKE TREMOLO LOUD"
250 DATA 100,0
260 DATA"ELIMINATE TREMOLO"
270 DATA 0,0
280 DATA "end",0,0
```

LOOK

Function checks the playback state of the instruments

Format _LOOK([<variable 1>][,<variable 2>][,<variable 3>][,<variable 4>])

Example _LOOK(A, B, C, D)

Explanation This command checks the playback status of the various instruments and assigns these to the variable.

The final played mark number is assigned upon completion of playback. 1000 marks are added to the mark number being played.

0 is assigned to instruments not defined by the _INST command.

<variable 1> – <variable 4> correspond to instruments 1 through 4.

Sample program

```
10 '_LOOK sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 _PHRASE(1,"cdefgab<c",2)
70 _PLAY(1,1,2)
80 PRINT"INSTRUMENT 1";TAB(20);"INSTRUMENT 2"
90 GOSUB 180
100 PRINT "PLAYING └ MARK 2";TAB(20);"└ UNDEFINED"
110 _WAIT(1)
120 PRINT
130 GOSUB 180
140 PRINT"END └ MARK 2";TAB(20);"└ UNDEFINED"
150 END
160 '
170 '
180 _LOOK(A,B):PRINT A:TAB(20) B
190 RETURN
```

MODINST

Function changes the contents of the instruments

Format _MODInst (<instrument number>(,<voice number>)(,<transposing>)(,<volume>)(,<portamento>)
(,<portamento speed>) (,<sustain>)(,<trigger mode>)(,<LFO sync>) (,<tremolo sensitivity>)
(,<vibrato>))

Example _MODI(1, 16,, 80)

Explanation The instruments are set by the _INST and _CANCEL statements to the following values. The _MODINST command is used to change these values. It has no effect on MIDI.

Tone:	BRASS 1
Transposing:	none
Volume:	100 (maximum)
Portamento:	fingered portamento
Portamento speed:	portamento off
Sustain:	off
Trigger mode:	multi
LFO sync:	off
Tremolo sensitivity:	set value of BRASS 1
Vibrato sensitivity:	set value of BRASS 1

The <instrument number> specifies the tone of the instrument defined by the _INST command.

The <voice number> specifies the tone of the instrument. The setting range is 1 – 56.

1 – 48: The tone is contained in the FM Sound Synthesizer unit (47 and 48 are reserved for future applications. There will be no sound output if they are selected).

49 – 56: Tones registered by the _SEL command.

<tranposing> allows the various instruments to be transposed separately, unlike the _TRANS command. The setting range is −12 to + 12 in half steps.

The <volume> is not affected by the V symbols of the playback list (refer to the _PHRASE command). This sets the volume balance for each instrument. The setting range is 0 – 100.
100 specifies the maximum volume.

<portamento> selects the portamento mode.

0: finger portamento (portamento only during legato playback)

1: full portamento (portamento at all times)

The <portamento speed> sets the rate at which the portamento changes. The setting range is 0 – 100. 100 is the slowest setting. 0 has the same effect as when the portamento is off.

<sustain> sets the length of sustain after the key is released.

0: the length determined for each tone

1: twice the length determined for each tone.

<trigger mode> determines whether or not there will be attack when the keys of the keyboard are depressed.

0: multi trigger (attack when the keys are depressed)

1: single trigger (no attack during legato playback)

<LFO sync> specifies whether or not there will be synchronization of the LFO at the instant when the keys are depressed.

0: off (no synchronization)

1: on (LFO starts from the beginning of the waveform every time a key is depressed)

<tremolo> specifies the amount of tremolo for each instrument. The setting range is 0 to 100. The larger the value is, the greater the degree of sensitivity although there are actually four stages of alteration.

<vibrato> specifies the amount of vibrato for each instrument. The setting range is 0 to 100. The larger the value is, the greater the degree of sensitivity although there are actually eight stages of alteration.

* If the tone of the instrument is changed after the <LFO sync>, <tremolo sensitivity>, or <vibrato sensitivity> is specified, the values will be determined according to the newly set tone.

* <portamento>, <portamento speed>, and <trigger mode> are only effective for single notes.

Sample program 1

```
10 '_MODI sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 _PHRASE(1,"cdefgab<c")
70 _PLAY(1,1)
80 _WAIT(1)
90 PRINT "CHANGE VOICE":PRINT
100 _MODI(1,4)
110 _PLAY(1,1)
120 _WAIT(1)
130 PRINT "CHANGE VOLUME":PRINT
140 _MODI(1,,,80)
150 _PLAY(1,1)
```

Sample program 2

```
10 '_MODI sample2
20 '
30 _INIT
40 _INST(1)
50 _MODI(1,,,,1,40)
60 _PHRASE(1,"c<c<c<c")
70 _PLAY(1,1)
```

ON EVENT (n) GOSUB

Function defines the subroutine that the program goes to upon the event

Format _ON EVENt((<event number>)) GOSUB <line number>

Example _ON EVEN (6) GOSUB 1000

Explanation This command defines the first line of the subroutine that the program jumps to when the an interrupt is generated upon the execution of an event. An interrupt is generated upon the execution of the specified event and the program executes the subroutine when the program is in the _EVENT (n) ON mode.

The interrupt subroutines can be defined according to each event. The <event number> specifies the type of event.

1-4: interrupt upon the completion of playback started by the _PLAY command. 1 - 4 are the values corresponding to the instrument numbers.

5: interrupt upon completion of rhythm playback

6: interrupt based upon the timer of the FM Sound Synthesizer unit

If the <event number> is omitted, the command applies to all events.

(example) _EVENT ( ) GOSUB

The <line number> is the beginning line of the subroutine that the program jumps to upon generation of the interrupt. Return from the interrupt subroutine in the same way as subroutines are called by the GOSUB command; the RETURN command is used. If a line number is not specified by the RETURN command, the program suspended by the interrupt will be returned to. Specification of a line number will cause execution to begin from that line.

The program will go to the EVENT STOP mode while the interrupt subroutine is being executed and return to the EVENT ON mode with the execution of the RETURN command.

Note: This command only defines interrupts. Execution of this command does not cause an interrupt to be generated.

The priority of interrupts is as follows:

1. BASIC interrupt

2. Instrument 1

3. Instrument 2

4. Instrument 3

5. Instrument 4

6. Rhythm

7. Timer

Processing begins from again from high priority interrupts if there are interrupts on hold when the processing of one interrupt is completed.

Sample program

```
10 '_ON EVENT(n) GOSUB sample
20 '
30 CLS
40 _INIT
50 _TRACK(2)
60 _ON EVENT(1) GOSUB 190
70 _ON EVENT(2) GOSUB 230
80 _EVENT() ON
90 _INST(1):_INST(2)
100 _MODI(2,4)
110 _PHRASE(1,"cdefgab<c")
120 _PHRASE(2,"efg")
130 _STANDBY
140 _PLAY(1,1)
150 _PLAY(2,2)
160 _START
170 IF X=2 THEN END
180 GOTO 170
190 '
200 PRINT "END INSTRUMENT 1"
210 X=X+1
220 RETURN
230 '
240 PRINT "END INSTRUMENT 2"
250 X=X+1
260 RETURN
```

PATTERN

Function defines rhythm patterns

Format _PATTern (<pattern number>,<variable(0)>)

Example _PATT(7, A$(0))

Explanation This command defines the various rhythm patterns. It allows the creation of rhythm patterns other than the patterns which are contained in the unit.

The procedure for the creation of rhythm patterns is as follows:

(1) The rhythm pattern is given as array variables.

The name of the variable is PT$ in this case.

PT$ (0) gives the length of the pattern.

"3": quarter notes X 3

"4": quarter notes X 4

"8": quarter notes X 8

PT$(1) – PT$(5) gives a character string of ones and zeros for the rhythm pattern of the various percussion instruments.

PT$ (1): high–hat close

PT$ (2): high–hat open

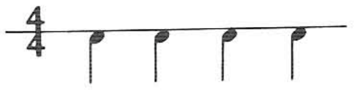
PT$ (3): bass drum

PT$ (4): high tomtom

PT$ (5): low tomtom

High–hat open and high–hat close cannot be sounded simultaneously. Only high–hat open will be sounded if both are designated to play simultaneously.

The specification of the pattern is in the units of 1/12 of a quarter note. One is written in the on position (when sound is output). The sound will continue as long as the ones continue. Thus, if "11" is written there will be only one sound not two.



The above rhythm pattern is set as follows:

PT$ (3)="100000000000100000000000100000000000100000000000"

The length of the character string must correspond to the length of the pattern.

PT$(0) = "3" 36 characters

PT$(0) = "4" 48 characters

PT$(0) = "8" 96 characters

(2) Register the rhythm pattern using the _PATTERN statement.

The pattern is registered by the _PATTERN statement after PT$(0) – PT$ (5) have been specified by the above procedure.

The <pattern number> is 7 or 8. Two patterns can be set.

<variable(0)> specifies the character variable into which the pattern data is written. The subscript is (0) in this example. If the subscript is considered to be (m), the pattern length, PT$ (m) can be PT$ (m+1) – PT$ (m+5) for the writing of the pattern data.

Sample program

```
10 '_PATTERN sample
20 '
30 _INIT
40 _USERHYTHM
50 '
60 A$(0)="4"
70 FOR I=1 TO 5
80 READ A$(I)
90 NEXT I
100 _PATT(7,A$(0))
110 '
120 _SELP(7)
130 _RHYTHM(16)
140 DATA 100100100100100100100100100100100100100100101010
150 DATA 000000000000000000000000000000000000000000000000
160 DATA 100000000000000000000100100000000100000000000100
170 DATA 000000000000100000000100000100000100100000000000
180 DATA 000000000000100000000100000100000100100000000000
```

PHRASE

Function writes the playback data onto the specified track

Format _PHRAse(<track number>,<playback list>(,<mark number>))

Example _PHRA (1, "CDEFGAB04C")

Explanation This command writes the playback data expressed in the character string onto the designated track and prepares it for playback.

The <track number> specifies the track onto which playback data is written. This value can be selected from track 1 up to the number of tracks specified by the _PLAY command. Writing onto tracks is not possible while they are being played. This will cause the generation of a Device I/O error.

The <playback list> is a character string which expresses the playback contents. It consists of one or more symbols. The playback list can be composed of character constants, character variables or a combination of the two. Letters used in the playback list can be either upper or lower case.

(Example)

```
_PHRASE(1,"cde")
_PHRASE(1,A$)
_PHRASE(1,"o4"+A$)
```

The <playback list> can be written as <character expression 1 >,<character expression 2>, ... < character expression n >. This will result in simultaneous playback of the character expressions. This procedure allows the playback of multiple parts for the same instrument. The maximum number of parts must be specified in the _INST command prior to the use of this procedure.

(Example)

```
_PHRASE(1,"cdefg","efgab") or _PHRASE(1,"cdefg,efgab")
```

* The pitch of notes

A-G[#/+/−/!]

The pitch of notes is expressed by their scale (Do, Re, Mi, Fa, So. La, Ti). C − G correspond to Do − Ti of C major while A and B correspond to La and Ti. Raising by a semitone can be expressed by attaching (#) or the plus (+) sign after A − G. Attaching a minus (−) sign after the note will lower it by a semitone. Notes which have had their key changed by the S or K command can be returned to their original value by attaching an exclamation mark (!) to them.

These temporary designations only apply to the scale of the note directly preceeding them.

(Example)

```
_PHRASE(1,"def#")
```

O <numeric value>

The range of the scale explained above is 0 − 8. The initial value is o3 which is a scale beginning with C and continuing to include one octave. o4 is one octave above o3, and o2 is one octave below o3.

(Example)

```
_PHRASE(1,"o2cego3cego4cego5c")
```

The octave can also be raised and lowered through the use of the "<" and ">" marks. The "<" mark raises the octave by one and the ">" mark lowers it by one.

(Example)

```
_PHRASE(1,"n60n62n64")
```

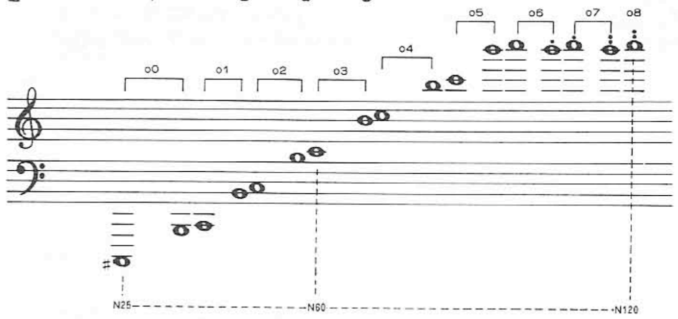
* The octave range played is dependent upon the selected tone. Automatic adjustment up or down in one octave units occurs if the range is exceeded during playback.

N <numeric value>

This expresses the pitch of the note in the range of 25 to 120. N25 is equivalent to C# in the o0 range. The note will raise in half-note step , such as N26, N27, etc. N120 is equivalent to C in the o8 range. N0 has the same function as R (rest).

(Example)

```
_PHRASE(1,"o2ceg<ceg<ceg<c")
```



* Some types of voice data designate the absolute pitch of notes, and raise or lower the pitch by one or two octaves relative to other tones. The latter kind of tones have a direct effect upon the the designated pitch of the notes in the playback list. The actual pitch of the notes as perceived by the ear is the sum of the designated pitch of the notes, transposing and the absolute pitch of the tone data.

* Length of note

L <numeric value> **The length of the notes is note/<numeric value>.**

L1	Full note	L8	Eighth note
L2	Half note	¦	
L3	Third note	¦	
L4	Quarter note	¦	
¦		L64	Sixtyfourth note

The initial value is L4 (quarter note). Specifying the note length will cause all the following notes on the same track to be the same length. This is effective until another value is specified for the note length. The length of only one note can be changed by attaching the <numeric value> expressing the note length after the scale name. For example, L16C and C16 are the same.

(Example)

```
_PHRASE(1,"L8ceL4gaL8ge")
```

. <period>

This is attached after scale symbol, in the same way as dotted notes, to multiply the present length of the note by 1.5 times. The multiplication factor will be 9/4 if two dots are attached and 27/8 if three dots are attached.

(Example)

```
_PHRASE(1,"cde.l8c")
```

R <numeric value>

This expresses the rests. The specification syntax for the <numeric value> is the same as the command. The length specified by the L or W command applies when the <numeric value> is omitted.

W <numeric value>

This expresses the length of the note in a numeric value between 1 and 96. W1 is equivalent to 1/96 of a full note (1/24 of a quarter note). W2 is 2/96 of a full note and W3 is 3/96 of a quarter note. The length is determined the portion of a full note equal to 1/96 multiplied by the <numeric value>.

(Example)

```
_PHRASE(1,"w12cew24gaw12ge")
```

* Tempo

"T <numeric value>"

This specifies the tempo. The <numeric value> sets the percentage of the tempo specified by the _TEMPO command. The setting range is 1 – 200. T50, for example, sets the tempo at 50% of the tempo specified by the _TEMPO command. The initial value is T100.

(Example)

```
_PHRASE(1,"cdet200cde")
```

* Dynamics

"V <numeric value>"

This sets the dynamics of the note.

The range of the <numeric value> is 1 – 100.

The greater the value, the stronger the note.

The setting is effective until the next setting of dynamics. The initial value is V50.

(Example)

```
_PHRASE(1,"v30cv50dv70ev90f")
```

* There are some tones such as PORGAN 1 and 2, etc. which are not affected by the V command. The V command alters the velocity of the notes. The degree to which the velocity will be altered is determined by the Velocity Sensitivity setting for tones are created by the FM Voicing Program.

* Tie

"&"

Placing the command between two notes having the same pitch will extend the length of the notes to the equivalent of two notes. When the notes on both sides of the & command have a different pitch, only the note in front will be played for 100% of its value. The "&" symbol is only valid for characters in one character string.

(Example) `_PHRASE(1,"cd&de&f")`

* Time

M <numeric value>

Sets the time at a value equivalent to <numeric value>/4. The numeric value may be between 3 and 8. (4 quarter time).

/character string/

The character string enclosed in the slash marks is considered to be one bar. The length of one bar is specified by the M command. If the total length of the notes of the character string is less than the length of one bar, the remaining portion is considered to be rests. An error will occur if the total length of the notes exceeds the length of one bar.

(Example) `_PHRASE(1,"m4/cde/cfa")`

* Key

S <numeric value>

This specifies the key. The <numeric value> is equivalent to the number of sharps (#). For example, if S2 is specified the key will be D major (B minor) and there will be no need to attach sharps or plus signs to any C or F that follow.

(Example) `_PHRASE(1,"s2defgab<cd")`

K <numeric value>

This specifies the key. The <numeric value> is equivalent to the number of flats (b). For example, if K1 is specified, the key will be F major (D minor) and there will be no need to attach flats or minus signs to notes which follow.

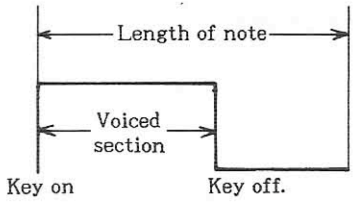
(Example) `_PHRASE(1,"k3cdefgab<c")`

* Execution

% <numeric value>

This is used to express staccato and tenuto. The <numeric value> expresses the proportion of the voiced section (from key on to key off) to the total length of the note. The setting range is 1 to 100. When the value is 100, the voiced section will be the entire length of the note.
The beginning of the _PHRASE command sets the value at 90%, making the "%" symbol valid only for notes in the same character string.

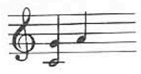
(Example) `_PHRASE(1,"cde%20cde")`



*Chords

[<character string>]

The procedure for the output of a chord for one instrument is to either divide the playback list with commas (",") or to use square brackets ([]). The notes enclosed in brackets will be output simultaneously. When the length of notes differ, rests are written after the shorter length of notes. This will result in the next note being played after the short rest has been played. Use the _INST command to specify the maximum number of voices when chords are to be used.



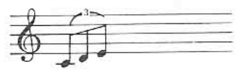
The above score is entered as follows: _PHRASE (1, "[c2L4g]a"), or the following:

_PHRASE(1,"L4ga","L2c") or _PHRASE(1,"L4ga,L2c")

* Notes surabondantes

(<character string>) <numeric values>

The L command can be used to play to divide a note of a specific length into X number of notes, for example, triplets, quintuplets and other notes surabondantes. However, they can be input much more easily by the use of { } brackets. For example, the input of "{CDE}" will be the same as L12CDE. This is expressed on the following score.



In other words, the note divided by <numeric value> are divided into equal parts according to the number of the symbols in the { } brackets. If the <numeric value> is omitted, the length set by the L or W command will apply.

(Example) `_PHRASE(1,"cde{gag}4")`

* The setting of the "0", "<", ">", "L", "W", "V", "S", "K" and "M" symbols will apply to all the notes of the same track until they are reset.

* All of the set values will be initialized upon execution of the _INST or _ERASE commands.

* All of the <numeric values> described above can be either character constants or numeric variables. In this case, "=<name of variable>" is substituted for <numeric value>.

```
(Example) 10 _INIT                    40 _PHRASE(1,"n=i;")
          20 _INST(1,,2)              50 NEXT I
          30 FOR I=48 TO 96           60 _PLAY(1,1)
```

* ( ) and { } can not **be mixed as** ({...}) or {(...)}.

<mark numbers>

The <mark numbers> can be freely used within the range of 1 to 254. If the <mark number> is omitted, the <mark number> will be considered to be equal to the <track number>.

Refer to PLAY, TRACK

PLAY

Function plays music

Format _PLAY(<instrument number> <track number> [,<mark number>])

Example _PLAY(1, 1, 1)

Explanation This command plays the designated marked section of the designated track. Playback data must be written into the track beforehand by the _PHRASE command.

The <instrument number> selects the instrument from the instruments defined by the _INST command.

The <track number> specifies the track to be played back. Playback using the Music Keyboard is possible by specifying the the <track number> as 9.

If the <track number> is 9, scanning of the Music Keyboard will not be automatically suspended when the program is suspended. The _STOP command is used to suspend the Music Keyboard.

The number of tracks must be set by the _TRACK command when tracks 2 – 8 are used.

The <mark number> sets the mark number for playback. If the <mark number> is omitted, the <mark number> is considered to be the same as the <track number>. Specifying the <mark number> as 255 will result in playback of the section after the last played section. When playback is suspended by the _STOP command, the beginning of the track is played back.

When the designated instrument is playing, new playback of the same instrument is performed upon playback completion (playback wait).

This playback wait mode uses only one _PLAY command. Setting of the playback wait mode by the _PLAY command will result in a Device I/O error when another _PLAY command is executed.

Playback of two or more instruments on the same track or Music Keyboard cannot be played by the _PLAY command.

Refer to STOP

RCANCEL

Function cancels the rhythm instrument

Format _RCANcel

Example _RCAN

Explanation This command release the sound generation allocated to the rhythm instrument. Using this, the total number of instrument voices will increase from 6 to 8.

Refer to USERHYTHM

REPORT

Function returns the system variables

Format _REPOrt ((<variable 1>) (,<variable 2>)(,<variable 3>)

Example _REPO (A,B)

Explanation This command returns the updated value of the mark number, number of rhythm repeats, etc. which were last played.

<variable 1> gives the value of the error flag. Errors are normally displayed as "Syntax error" or a similar message. However, there are cases when playback does not start from the playback wait mode by the _PLAY command and the error is not indicated. Here, the point where the error occured is not known and thus a BASIC Syntax error is not generated. However, the FM Music Macro sets an error flag. There are 5 varieties of errors which register in bits 0 − 4.

```
○○○ * * * * *
    │ │ │ │ └── The buffer of the Music Keyboard has overflowed.
    │ │ │ └──── A MIDI I/O error
    │ │ └────── The track/music keyboard is being played.
    │ └──────── The contents of the track are incorrect.
    └────────── An interrupt request is generated by a certain event while processing of
                an interrupt which was generated by the same event is in progress.
                Occurs when pitch processing time is longer than the interrupt pitch.
```

The error flags are reset by the _INIT command when _PLAY is executed.

<variable 2> gives the mark number of the last played section.

<variable 3> gives the remaining number of times the rhythm is to be repeated.

Sample program 1

```
10 '_REPORT sample 1
20 '
30 CLS
40 _INIT
50 _TEMPO(60)
60 _USER
70 _SELP(1)
80 _RHYT(16)
90 _REPORT(,,D)
100 IF D=D2 THEN 90
110 D2=D
120 PRINT"AFTER";D;"VALUE"
130 IF D=0 THEN END
140 GOTO 90
```

Sample program 2

```
10 '_REPORT sample 2
20 '
30 CLS
40 _INIT
50 _ON EVENT(1) GOSUB 260
60 _ON EVENT(2) GOSUB 300
70 _EVENT() ON
80 _TRACK(3)
90 _INST(1)
100 _INST(2)
110 _MODI(2,10)
120 _PHRASE(1,"l2cdefgab<c")
130 _PHRASE(2,"l2efg")
140 _PLAY(1,1):_PLAY(2,2)
150 _PLAY(1,3):_PLAY(2,1)
160 GOSUB 220
170 FOR I=1 TO 3000:NEXT I
180 PRINT"_INITIALIZE"
190 _INIT
200 GOSUB 220
210 END
220 '
230 _REPORT(A)
240 PRINT"-REPORT- ";BIN$(A):PRINT
250 RETURN
260 '
270 PRINT"PLAY inst1-track3"
280 GOSUB 220
290 RETURN
300 '
310 PRINT"PLAY inst2-track1"
320 GOSUB 220
330 RETURN
```

RHYTHM

Function used for rhythm playback

Format _RHYThm (<number of repetitions>, (<mark number>))

Example _RHYT(40,2)

Explanation This command plays back the rhythm patterns selected by the _SELP command.

The <number of repetitions> specifies the number of repetitions for the rhythm. It is set in number of the quarter note. If 120 is specified, 30 bars of 4/4 time are repeated, or 40 bars of 3/4 time are repeated.

The <mark number> can be set to any value between 1 and 254. 10 is specified if the <mark number> is omitted.

Sample program

```
10 '_RHYTHM sample
20 '
30 CLS
40 _INIT
50 _USERHYTHM
60 _SELP(1)
70 INPUT"NUMBER OF REPETITIONS";A
80 _RHYTHM(A)
```

RHYTHM

Function used for rhythm playback

Format _RHYThm (<number of repetitions>, (<mark number>))

Example _RHYT(40,2)

Explanation This command plays back the rhythm patterns selected by the _SELP command.

The <number of repetitions> specifies the number of repetitions for the rhythm. It is set in number of the quarter note. If 120 is specified, 30 bars of 4/4 time are repeated, or 40 bars of 3/4 time are repeated.

The <mark number> can be set to any value between 1 and 254. 10 is specified if the <mark number> is omitted.

Sample program

```
10 '_RHYTHM sample
20 '
30 CLS
40 _INIT
50 _USERHYTHM
60 _SELP(1)
70 INPUT"NUMBER OF REPETITIONS";A
80 _RHYTHM(A)
```

RSTOP

Function stops rhythm playback

Format _RSTOp

Example _RSTO

Explanation Rhythm playback is stopped when the _RSTOP command is executed.

Sample program

```
10 '_RSTOP sample
20 '
30 CLS
40 _INIT
50 _USERHYTHM
60 _SELP(1)
70 _RHYTHM(16)
80 FOR I=1 TO 700:NEXT I
90 PRINT"STOP RHYTHM"
100 _RSTOP
```

SELPATTERN

Function selects the rhythm pattern.

Format _SELPattern (<pattern number>)

Example _SELP (3)

Explanation This command selects two rhythm patterns for playback data. These rhythm patterns can be either the six pre-recorded patterns or user defined patterns.

The <pattern number> is from 1 to 8. 1 to 6 are the patterns contained in the unit, and 7 and 8 are patterns defined by the _PATTERN command. The patterns contained in the unit are as follows:

1: 16 beats

2: slow rock

3: waltz

4: rock

5: disco

6: swing

Refer to PATTERN

Sample program

```
10 '_SELPATTERN sample
20 '
30 _INIT
40 CLS
50 _USERHYTHM
60 INPUT"RHYTHM PATTERN NUMBER(1-6)";A
70 IF A<0 OR A>6 THEN 30
80 IF A=0 THEN END
90 _SELP(A)
100 _RHYTHM(8)
110 _WAIT(5)
120 GOTO 60
```

SELVOICE

Function registers the loaded voice data to be used.

Format _SELVoice ([<voice number 1>][,<voice number 2>]...[,<voice number 8>])

Example _SELV (6, 8, 25, 33)

Explanation Selects 8 voices from the voice data loaded by the _CLDVOICE command and renders it usable.

The <voice number> is the voice number when the voice was created using the FM Voicing Program.

When the voice of the instruments is specified by the _MODINST command, the numbers of the voices registered by the _SELVOICE command, are 49 to 56. If the voice number is omitted, the previously set number applies.

Refer to CLDVOICE

SOUND

Function directly controls instruments.

Format _SOUNd (<instrument number>,<control mode>

[,<key number>][,<fine>][,<velocity>][,<level>])

Example _SOUN(1, 1, 60,, 70)

Explanation This command allows the sound output of instruments to be directly controlled without using tracks or the Music Keyboard. The use of this command is a little more difficult than the use of the _PLAY command, but it allows the pitch and volume of the sound output to be changed during playback, and thus, is useful for the production of special effects sound. It cannot be used with MIDI.
The <instrument number> selects the instrument from those defined by the _INST command.
The <control mode> specifies the mode of operation.

0: no key on/off (used for changing the pitch and volume while the note is being sounded)

1: key on (note is sounded)

2: key off (note is not sounded)

The <key number> specifies the pitch. The procedure is the same as the N command in the playback list. (Refer to the _PHRASE command.)

When the value is 25, the note is a C# in octave 0.

The note is raised a half step for each increment, and thus 120 is equivalent to a C in octave 8.

The <fine> changes the pitch to less than a half step. The setting range is 0 to 100. The pitch will be approximately a half step higher than the pitch set by the <key number>.

The <level> specifies the volume level. The setting range is 0 to 100. The volume will increase as the value increases. If the value is omitted, the volume will be equivalent to 100.

Sample program

```
10 '_SOUND sample
20 '
30 _INIT
40 _INST(1)
50 _MODI(1,12)
60 _SOUND(1,1,40)
70 FOR J=.4 TO 4 STEP .2
80 _SOUND(1,1,40)
90 FOR I=30 TO 50 STEP J
100 _SOUND(1,0,I)
110 NEXT I
120 NEXT J
130 FOR J=1 TO 30
140 FOR I=30 TO 50 STEP 4
150 _SOUND(1,0,I)
160 NEXT I
170 NEXT J
180 _SOUND(1,2)
```

STANDBY

Function temporarily suspends playback.

Format _STANDby

Example _STAN

Explanation Execution of this command temporarily suspends playback. This means that playback will not start even if a _PLAY command is executed. When more than one instrument (including rhythm instrument) are started at the same time, the _STANDBY command is used to temporarily suspend playback. The _PLAY command is then executed. Simultaneous playback of all instruments will begin upon execution of the _START command.

Refer to START

Sample program

```
10 '_STANDBY sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 _PHRA(1,"cdefgab<c")
70 _STANDBY
80 PRINT"_PLAY"
90 _PLAY(1,1)
100 PRINT"_DO NOT START PERFORMANCE FROM STANDBY"
110 FOR I=1 TO 500:NEXT I
120 PRINT"_START"
130 _START
```

START

Function releases the temporary playback suspension mode.

Format _STARt [(<option>)]

Example _STAR

Explanation This command releases the temporary suspension mode engaged by the _STANDBY command and starts playback.

When the <option> is 1, the temporary suspension mode is released when the synchronization signal (written by the _SYNCOUT command) is read from the cassette recorder. In this case, the motor of the cassette recorder is automatically turned on by the execution of this command.

When the <option> is 0 or when omitted, the temporary suspension mode is unconditionally released.

Refer to STANDBY

STOP

Function suspends playback

Format _STOP (<instrument number>)

Example _STOP (2)

Explanation This command suspends playback of the designated instrument. Scanning of the Music Keyboard is also suspended if it is assigned instead of a track.

When the Music Keyboard is assigned to an instrument by the _PLAY command, scanning of the Music Keyboard will not be suspended when the program is suspended. This can result in slower BASIC processing. Scanning can be suspended by either the _STOP command or the _INIT command

Refer to PLAY

SYNCOUT

Function outputs a playback synchronization signal to a cassete recorder.

Format _SYNCout

Example _SYNC

Explanation The FM Music Macro and the cassette recorder must be synchronized to start at the beginning of the desired selection when making stereo recording with a multi-track tape recorder. The _SYNCOUT command outputs a synchronization signal to the tape recorder, which finds the beginning of a certain selection.

Refer to START

TEMPO

Function specifies the playback tempo

Format _TEMPo (<tempo value>)

Example -TEMP

Explanation This command sets the playback tempo. The setting range is 0 - 200. The value is the number of quarter notes which are sounded in one minute. For example, if the value is 60, the tempo will be equivalent to a quarter note every second. Playback will be suspended if the tempo value is 0.

The tempo set in the playback list by the T command will be reset to 100% on execution of the _TEMPO command.

Refer to PHRASE

Sample program

```
10 '_TEMPO sample
20 '
30 _INIT
40 T=120
50 _ON EVENT() GOSUB 200
60 _EVENT() ON
70 _INST(1)
80 _PHRA(1,"L8cgegcgegdgfgcgegcafacgeg>b<gdgcgeg")
90 PRINT"_TEMPO( 120 )
100 _PLAY(1,1)
110 A$=INKEY$:IF A$="" THEN 110
120 IF A$=CHR$(30) AND T<200 THEN T=T+5:GOTO 160
130 IF A$=CHR$(31) AND T>0 THEN T=T-5:GOTO 160
140 IF A$=CHR$(27) THEN _INIT:END
150 GOTO 110
160 PRINT"_TEMPO(";T;")";
170 IF T=0 THEN PRINT"-PAUSE" ELSE PRINT
180 _TEMPO(T)
190 GOTO 110
200 _PLAY(1,1)
210 RETURN
```

Pressing the (↑) key while this program is running will increase the tempo while pressing the (↓) key will reduce the tempo. The program can be suspended by depressing the (ESC) key.

TIMER

Function   starts and sets the pitch (time) of the timer

Format   _TIME (<pitch>[,<mark number>])

Example   _TIME (100, 1)

Explanation Although there is a timer in MSX BASIC, the separate timing feature of the FM Macro may be used when a timer is needed.

The _TIMER command sets the pitch between the generation of interrupts, and simultaneously starts the timer. If this is set by the EVENT ON command, an interrupt is generated after the elapse of the <pitch>, and the program will execute the subroutine specified in the _ON EVENT (n) GOSUB command.

The <pitch> is set in 1/100th sec units. The setting range is 1 to 24,000.

The <mark number> can be set anywhere between 1 and 254. 11 will be selected if the <mark number> is omitted.

Refer to   TSTOP

Sample program

```
10 '_TIME sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 N=10
70 _TIME(100)
80 PRINTUSING"##";N
90 N=N-1
100 IF N=-1 THEN 130
110 _WAIT(6)
120 GOTO 80
130 _TSTOP
140 PRINT"END"
```

TRACK

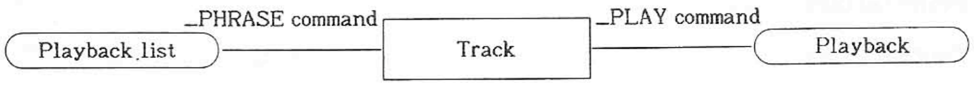
Function defines the number of tracks to be used.

Format _TRACk (<number of tracks>)

Example _TRAC (3)

Explanation This command defines the number of tracks to be used with the _PHRASE or _PLAY command. This allows simultaneous playback of the number of tracks defined.

The setting range of the <number of tracks> is 1 to 8. If the number of tracks is omitted, 1 will be selected. Playback data is written on each track by the _PHRASE command, and playback is started by the _PLAY command.



The size of each track will decrease as the number of tracks increases.

TRANSPOSE

Function transposes all instruments.

Format _TRANSpose (<numeric value>)

Example _TRAN (4)

Explanation This command simultaneously transposes all instruments. The <numeric value> can be set in half tone increments, and the instruments can be transposed up or down a maximum of one octave. The setting range is −12 to 12.

Sample program

```
10 '_TRANSPOSE sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 _PHRASE(1,"cegec")
70 _PLAY(1,1)
80 _WAIT(1)
90 INPUT"TRANSPOSE TO WHICH?(-12~12)";K
100 IF K=99 THEN END
110 IF K<-12 OR K>12 THEN 90
120 _TRANS(K)
130 _PLAY(1,1)
140 GOTO 80
```

Run the program and listen to the range of transposing by the input of a numeric value between −12 and 12. The program can be suspended of the input of 99.

TSTOP

Function stops the timer.

Format _TSTOp

Example _TSTO

Explanation This command stops the timer of the FM Music Macro. If the program used with the FM Music Macro is suspended, the timer does not automatically stop. The timer can be stopped with the _TSTOP or _INIT command.

Refer to TIMER

TUNE

Function tunes the FM Tone Generation system

Format _TUNE (<numeric value>)

Example _TUNE (50)

Explanation This command is used to tune the FM Music Macro to other instruments. The setting range is −100 to 100. The value can be set to a maximum of 1/2 tone (one semitone) up or down.

Sample program

```
10 '_TUNE sample
20 '
30 CLS
40 _INIT
50 _INST(1)
60 _MODI(1,12)
70 PRINT"_TUNE( 0 )"
80 _SOUND(1,1,60)
90 A$=INKEY$:IF A$=CHR$(27) THEN 160
100 IF A$=CHR$(30) AND K<100 THEN K=K+5:GOTO 130
110 IF A$=CHR$(31) AND K>-100 THEN K=K-5:GOTO 130
120 GOTO 90
130 PRINT"_TUNE(";K;")"
140 _TUNE(K)
150 GOTO 90
160 _SOUND(1,2)
```

Pressing the (↑) key while this program is running will increase the tempo while pressing the (↓) key will reduce the tempo. The program can be suspended by depressing the (ESC) key.

USERHYTHM

Function defines the rhythm instrument.

Format _USERhythm

Example _USER

Explanation This command is used to define the rhythm instrument. If use of the rhythm instrument is attempted when it has not been defined by the _USERHYTHM command, an error will be generated.

The rhythm instrument use two voices. Thus, when the _USERHYTHM command is used, the total number of instrument voices is decreased from 8 to 6.

Refer to PATTERN, RHYTHM, SELPATTERN, RSTOP and RCANCEL

VLIST

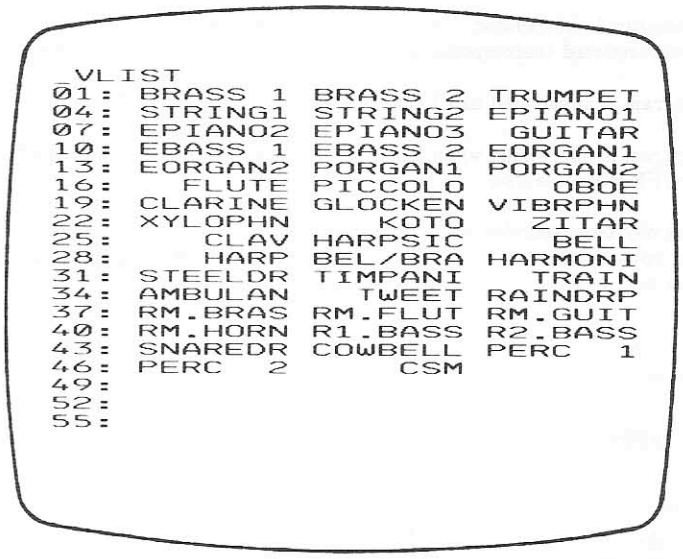
Function displays the tone table on the screen.

Format _VLISt

Example _VLIS

Explanation This command displays a table of all tones, including the tones contained in the FM synthesizer unit and the tones registered by the _SELVOICE command.

Display of the tone names by _VLIST.



```
_VLIST
01:  BRASS  1 BRASS  2 TRUMPET
04:  STRING1  STRING2  EPIANO1
07:  EPIANO2  EPIANO3   GUITAR
10:  EBASS  1 EBASS  2 EORGAN1
13:  EORGAN2  PORGAN1  PORGAN2
16:    FLUTE  PICCOLO     OBOE
19:  CLARINE  GLOCKEN  VIBRPHN
22:  XYLOPHN     KOTO    ZITAR
25:     CLAV  HARPSIC     BELL
28:     HARP  BEL/BRA  HARMONI
31:  STEELDR  TIMPANI    TRAIN
34:  AMBULAN    TWEET  RAINDRP
37:  RM.BRAS  RM.FLUT  RM.GUIT
40:  RM.HORN  R1.BASS  R2.BASS
43:  SNAREDR  COWBELL  PERC   1
46:  PERC   2      CSM
49:
52:
55:
```

WAIT

Function suspends program interruption during playback.

Format _WAIT (<event number>)

Example _WAIT (2)

Explanation Execution of the _WAIT command will suspend program execution until playback of the designated instrument or rhythm has been completed. Designation is by the <event number>.

The <event number> is the same as the _ON EVENT (n) GOSUB command and is a numeric value between 1 and 6.

1 -- 4: The program is suspended until the playback which was started by the _PLAY command has been completed (corresponds to the numbers of the various instruments).

5: The program is suspended until completion of rhythm playback.

6: The program is suspended when the time set with the _TIMER command has elapsed (timer of FM Music Macro).

* The program may be suspended for a long period of time if the event set with the _WAIT command has not been executed or has already been completed. Pressing the CTRL key and the STOP key will release the _WAIT command.

Sample program

```
10 '_WAIT sample
20 '
30 _INIT
40 _TRACK(2)
50 _TEMPO(80)
60 CLS:COLOR 10,1
70 N=1
80 _INST(1)
90 READ A$:_PHRASE(1,A$):_PLAY(1,1)
100 READ A$
110 T=N AND 1:T=T+1:N=N+1
120 IF A$="end" THEN END
130 _ERASE(T)
140 _PHRASE(T,A$)
150 _PLAY(1,T)
160 X=INT(RND(1)*31):Y=INT(RND(1)*22)
170 LOCATE X,Y
180 PRINT"*"
190 _WAIT(1)
200 GOTO 100
210 DATA g,f,b-,o4c8,o4f8,o4d,o4c8,o4f8,o4d.b-,o4c,g,f
220 DATA g,f,b-,o4c8,o4f8,o4d,b-,o4e-.o4c,b-
230 DATA end
```

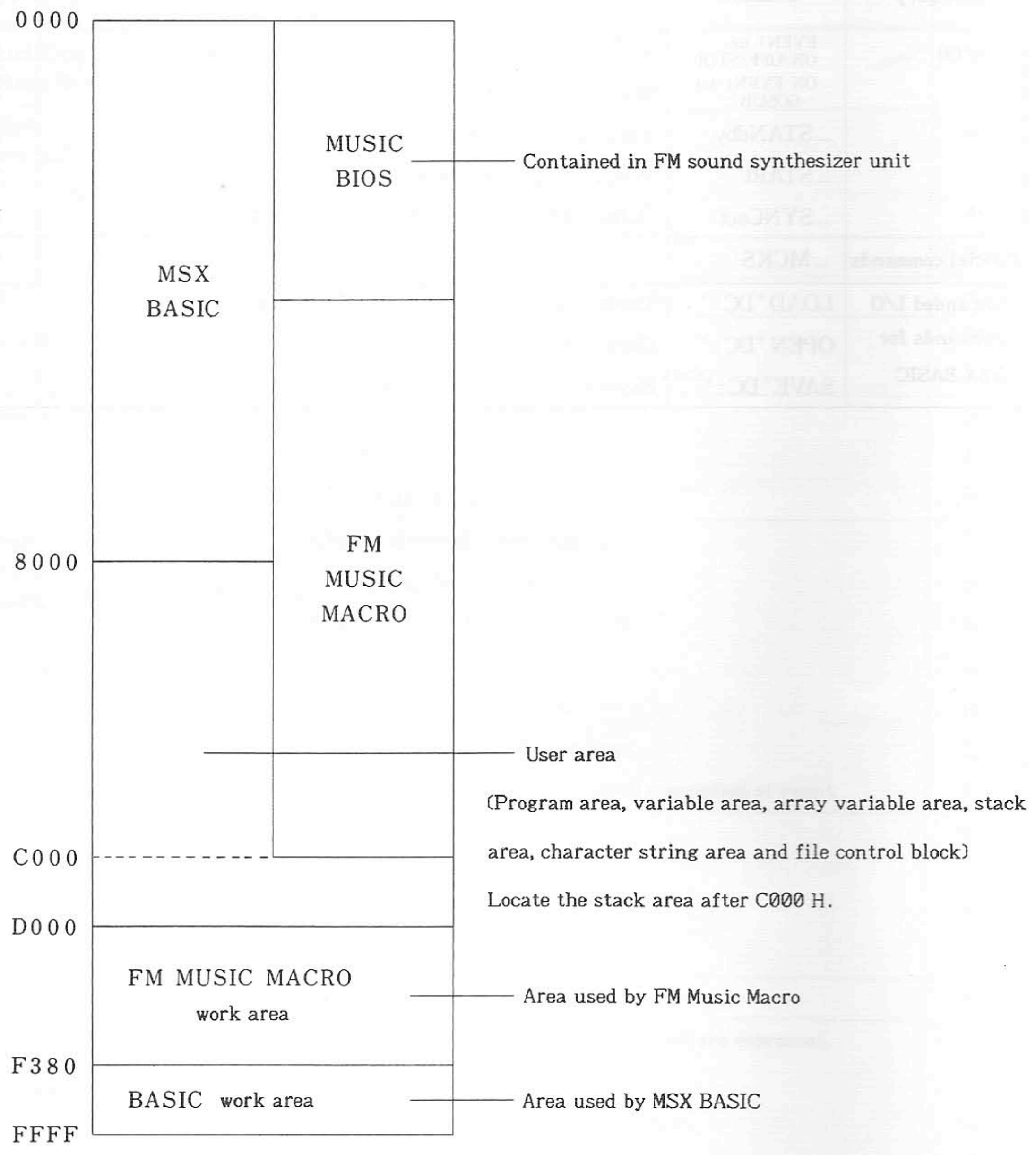
Ⅳ Appendix

IV-1 Table of FM Music macro Commands

Category	Command	Function	Page
Initializes FM Music Macro	_INIT	Initializes FM Music Macro.	27
Defines instruments	_CANCel	Releases instrument	23
	_INST	Defines instrument	29
	_RCANcel	Releases rhythm instrument	46
	_USERhythan	Defines rhythm instrument	64
Defines tracks	_TRACK	Defines number of tracks to be used.	60
Playback data	_PHRAse	Writes playback data on track.	40
	_LENGth	Checks length of playback data on track.	30
Playback	_ERASe	Clears contents of track.	25
	_PLAY	Starts playback.	45
	_STOP	Stops playback.	56
Changing and checking parameters	_LOOK	Checks playback conditions.	33
	_MODInst	Changes instrument content.	34
	_REPOrt	Gives values of system variables.	47
	_TEMPo	Sets playback tempo.	58
	_TRANspose	Transposes all instruments.	61
	_TUNE	Tunes FM tone generation system.	63
	_WAIT	Suspends program execution until completion of event.	66
Rhythm	_PATTern	Defines rhythm patterns.	38
	_RHYThm	Starts rhythm playback.	49
	_RSTOp	Suspends rhythm playback.	50
	_SELPattern	Selects rhythm pattern.	51
LFO	_LFO	Changes LFO data.	31
Direct accessing of SFG-01 or SFK-01	_INMKey	Checks whether keys of Music Keyboard are depressed.	28
	_SOUNd	Directly controls instrument.	53
	_TIMEr	Starts and sets interval (time) of timer.	59
	_TSTOp	Stops timer.	62

Category	Command	Function	Page
Control	_EVENt (n) ON/OFF/STOP	Permits/prohibits/holds generation of interrupt.	26
	_ON EVENt (n) ~GOSUB	Defines interrupt sub-routine.	36
	_STANdby	Temporarily suspends playback.	54
	_STARt	Releases temporary suspension of playback.	55
	_SYNCout	Outputs playback synchronization signal.	57
Special commands	_MCKS		72
Expanded I/O commands for MSX BASIC	LOAD "DC:"	Loads program from data memory cartridge.	71
	OPEN "DC:"	Opens file on data memory cartridge.	71
	SAVE "DC:"	Saves program onto data memory cartridge.	71

IV-2 Memory map


Hexadecimal addresses
0000
8000
C000
D000
F380
FFFF
MSX BASIC
MUSIC BIOS
FM MUSIC MACRO
FM MUSIC MACRO work area
BASIC work area
Contained in FM sound synthesizer unit
User area
(Program area, variable area, array variable area, stack area, character string area and file control block)
Locate the stack area after C000 H.
Area used by FM Music Macro
Area used by MSX BASIC

IV-3 Use of data memory cartridges

Yamaha data memory cartridges (UDC-01) can be substituted for a cassette recorder. This allows rapid and simple READ/WRITE operation.

SAVE/LOAD of programs

Programs are saved/loaded by the SAVE/LOAD commands in BASIC.

The device name is specified by "DC:".

Saving programs

```
SAVE "DC:"
```

Loading programs

```
LOAD "DC:"
```

If LOAD "DC:", R is entered, the program will be executed immediately after being loaded.

* The maximum size of programs that can be saved/loaded is 4Kb.

Files

Files can be created on the data memory cartridge by specifying the device name by OPEN "DC:" in BASIC. This allows the use of file related commands in BASIC. For the use of these commands, refer to the MSX BASIC reference manual.

Refer to the following BASIC commands:

OPEN

CLOSE

INPUT#

LINEINPUT#

PRINT#

EOR

Ⅳ-4 Special commands

The following commands allow the FM Music Macro to perform the following functions not supported by MSX BASIC.

```
_MCKS (7, 1, 16)
```

This command writes voice data onto a data memory cartridge. **It must be executed immediately** after voice data is read from cassette tape by the _CLDV command.

```
_MCKS (47, 3, 18, 0, <level>)
```

This command controls the volume of the rhythm instrument. The <level> is a numeric value between 0 and 255. When it is 255, the volume is at its maximum level. The initial value is 255.

```
_MCKS (11, 0)
```

This command suspends sound output of all instruments. It outputs a MIDI all-note off signal. However, there are certain instances where the synthesizer ignores the all-note off signal.

With the above commands, which directly call a kernel of the FM Music Macro, if values other than those specified above are input, the unit will function improperly.

IV-5 Sample programs

Rhythm box

Rhythm box

This is a simple rhythm box using marks. After inputting the RUN command, the number of rhythm repetitions (number of bars) is entered. Next, the code is entered for one bar. The code is selected from c, d, f, g or a, and is entered as a lower case character. These codes correspond to the following chords.

c: ceg	f: cfa
d: cfa	g: eag
e: egb	a: ace

```
10 '**********************
20 '*                    *
30 '*     SAMPLE         *
40 '*                    *
50 '*  BACKING MUSIC BOX  *
60 '*                    *
70 '**********************
80 '
90 '
100 CLS
110 _INIT
120 _TRACK(2)
130 INPUT "NUMBER OF REPETITIONS";X
140 IF X<=0 THEN 100
150 DIM A$(X)
160 _INST(1):_INST(2,3)
170 _MODI(1,11):_MODI(2,9)
180 _USERHYTHM
190 _SELP(4)
200 _TEMPO(150)
210 '
220 '
230 READ M,A$,B$
240 IF M=0 THEN 290
250 _PHRASE(1,A$,M)
260 _PHRASE(2,B$,M)
270 GOTO 230
280 '
290 PRINT "INPUT CODE"
300 PRINT"(c.d.e.f.g.a)"
310 FOR I=1 TO X
320 PRINTUSING"###";I;:INPUT A$(I)
330 IF A$(I)="b" OR A$(I)>"h" OR A$(I)<"a" THEN PRINT CHR$(7):GOTO 320
340 NEXT I
350 _STANDBY
360 _RHYTHM(X*4)
370 FOR I=1 TO X
380 M=ASC(A$(I))-96
390 _PLAY(1,1,M):_PLAY(2,2,M)
400 IF I=1 THEN _START:ELSE _WAIT(1)
410 NEXT I
420 '
430 '
440 '
450 DATA 1,l8o2aa<a>ar8a<a>a,%20r4l8[o2a<ce]r4[>a<ce]r4
460 DATA 3,l8o3cc<c>cr8c<c>c,%20r4l8[ceg]r4[ceg]r4
470 DATA 4,l8o3dd<d>dr8d<d>d,%20r4l8[dfa]r4[dfa]r4
480 DATA 5,l8o3ee<e>er8e<e>e,%20r4l8[egb]r4[egb]r4
490 DATA 6,l8o3ff<f>fr8f<f>f,%20r4l8[cfa]r4[cfa]r4
500 DATA 7,l8o2gg<g>gr8g<g>g,%20r4l8[o2b<dg]r4[>b<dg]r4
510 DATA 0,"",""
```

Special effects

The following is an example of special effects of two instruments using the _SOUND command. The cursor will appear in the middle of the screen when the program is run. Moving the cursor with the cursor keys will cause the sound to change.

Cursor keys	Change	Instrument 1	Instrument 2
↑	Pitch	HIGH	LOW
↓		LOW	HIGH
→	Volume	Increase	Decrease
←		Decrease	Increase

```
10 '**********************
20 '*                    *
30 '*   SAMPLE _SOUND    *
40 '*                    *
50 '**********************
70 '
80 _INIT
90 _INST(1)
100 _INST(2)
110 _MODI(1,2,,,,,,,,0,100)
120 _MODI(2,5,,,,,,,,0,100)
130 _LFO(1,80,,50)
140 '
150 FOR I=1 TO 8
160 READ S$
170 A$=A$+CHR$(VAL("&B"+S$))
180 NEXT I
190 SPRITE$(0)=A$
200 '
210 KC=20:FR=0:VE=28:MD=1
220 SCREEN 2
230 LINE(11,12)-(243,180),,B
240 LINE(100,96)-(154,96),3
250 LINE(127,74)-(127,118),3
260 '
270 PUTSPRITE0,(VE*4+12,172-KC*4),8
280 _SOUND(1,MD,KC+40,,100,VE+44)
290 _SOUND(2,MD,100-KC,,100,100-VE)
300 MD=0
310 '
330 A$=INKEY$:IF A$="" THEN 330
340 IF A$<CHR$(27) OR A$>CHR$(31) THEN 330
350 ON ASC(A$)-26 GOTO 390,440,490,540,600
360 '
390 _INIT
400 END
410 '
440 VE=VE+1:IF VE>56 THEN VE=56
450 GOTO 260
460 '
490 VE=VE-1:IF VE<0 THEN VE=0
500 GOTO 260
510 '
540 KC=KC+1
550 IF KC>40 THEN KC=40
560 GOTO 260
570 '
600 KC=KC-1
610 IF KC<0 THEN KC=0
620 GOTO 260
630 '
660 DATA 00010000,00010000,00111000,11111110,00111000,00010000,00010000,00000000
```

Music & graphics

This final program displays graphics while the music is playing. The tune is Mussorgsky's Pictures at an Exhibition. This program is a little longer than the others, because of the complicated nature of the screen display.

The program uses a machine language routine for changing the background color.

```
10 '*********************
20 '*                   *
30 '* MUSIC PLAY SAMPLE  *
40 '*                   *
50 '*       PROMENADE    *
60 '*                   *
70 '*********************
80 '
90 '---machine routine set------
100 '
110 CLEAR 200,&HCEFF
120 AD=52992!
130 READ A$:A=VAL("&h"+A$)
140 IF A$="" THEN 180
150 POKE AD,A
160 AD=AD+1
170 GOTO 130
180 DEF USR 1=&HCF00
190 '
200 '
210 DATA 23,23,56,21,00,28,01,00,10,f3,7d,d3,99,7c,e6,3f,f6,40,d3,99,7a,d3,98,fb
,23,0b,79,b0,20,eb,c9
220 DATA ""
230 '
240 '---sprite initial-------------
250 '
260 COLOR,1
270 SCREEN 2,2
280 FOR J=0 TO 2
290 X$=""
300 FORI=1TO32
310 READ S$
320 X$=X$+CHR$(VAL("&h"+S$))
330 NEXT I
340 SPRITE$(J)=X$
350 NEXT J
360 '
370 '---music initial-------------
380 '
390 _INIT
400 _TRACK(8):_TEMPO(95)
410 _INST(1):_INST(2,2)
420 _INST(3,3):_INST(4,2)
430 _MODI(1,1):_MODI(2,1,12)
440 _MODI(3,1):_MODI(4,4,12)
450 '
460 _ON EVENT(4) GOSUB 880
470 _EVENT(4) ON
490 '
500 '---music start-----------
510 '
520 FOR T=1 TO 2:T2=T+2:T3=T+4:T4=T+6
530 READ A$,B$,C$
540 _PHRASE(T,A$):_PHRASE(T2,B$,">"+B$)
550 _PHRASE(T3,C$):_PHRASE(T4,A$)
560 IF T=1 THEN _STANDBY
570 _PLAY(1,T):_PLAY(2,T2)
580 _PLAY(3,T3):_PLAY(4,T4)
590 IF T=1 THEN _START
600 NEXT T
```

```
610 '
620 '---main routine --------
630 '
640 COLOR ,1
650 FOR I=27TO 230 STEP 2
660 X=I*20-2432
670 IF X<0 THEN Y=191-127*(-1*X)/(I-X):LINE(I,64)-(0,Y),3:GOTO700
680 IF X>255 THEN Y=191-127*(X-255)/(X-I):LINE(I,64)-(255,Y),3:GOTO700
690 LINE(I,64)-(X,191),3
700 NEXT I
710 FOR I=2 TO 5 STEP .2
720 Y=EXP(I)+57
730 LINE(0,Y)-(255,Y),3
740 NEXT I
750 X=INT(RND(1)*256)
760 Y=INT(RND(1)*63)
770 PSET (X,Y),10
780 IF N<6 THEN 750
790 FORI=0 TO 8
800 IF F=1 THEN 1010
810 X=INT(RND(1)*246+5)
820 Y=INT(RND(1)*181+5)
830 C=INT(RND(1)*15+1)
840 PUT SPRITEI,(X,Y),C,INT(I/3)
850 NEXT I
860 GOTO 790
870 '
880 '---trap routine-----
890 '
900 T=N AND 1:T=T+1:N=N+1:T2=T+2:T3=T+4:T4=T+6
910 IF 3<N AND N<7 THEN CL%=3*16+(N+1):Z=USR1(CL%)
920 READ A$,B$,C$
930 IF A$="end" THEN CL%=53:Z=USR1(CL%):F=1:RETURN
940 _ERASE(T):_ERASE(T2)
950 _ERASE(T3):_ERASE(T4)
960 _PHRASE(T,A$):_PHRASE(T2,B$,">"+B$)
970 _PHRASE(T3,C$):_PHRASE(T4,A$)
980 _PLAY(1,T):_PLAY(2,T2)
990 _PLAY(3,T3):_PLAY(4,T4)
1000 RETURN
1010 '
1020 '---end--------------
1030 '
1040 _EVENT() OFF
1050 Q$=INKEY$:IF Q$="" THEN 1050
1060 COLOR 15,4,7
1070 END
1080 '
1090 '---sprite data------------
1100 '
1110 DATA 0,0,0,0,0,0,0,0
1120 DATA 0,0,0,0,3,7,f,e
1130 DATA 80,c0,e0,b0,90,90,90,90
1140 DATA a0,80,80,80,80,80,0,0
1150 DATA 0,0,7,7,4,4,7,7
1160 DATA 4,4,4,4,4,1c,3c,c8
1170 DATA 7,7f,f9,81,7,7f,f9,81
1180 DATA 1,1,1,7,f,e,0,0
1190 DATA 7,7,4,4,7,7,4,4
1200 DATA 4,1c,3c,38,0,0,0,0
1210 DATA 0,e0,fc,1f,3,e1,fd,1f
1220 DATA 1,1,1,1,1,7,f,e
```

```
1230 '
1240 ' ---music data ------------
1250 '
1260 DATA k2gfb<l8v60cv40fv50d4v60l8cv40fl4v50d>bv58<c>v50gv45f
1270 DATA r1r4r1r2
1280 DATA k2gfb<l8v60cv40fv50d4v60l8cv40fl4v50d>bv58<c>v50gv45f
1290 '
1300 '
1310 DATA k2gfb<l8v60cv40fv50d4v60l8cv40fl4v50d>bv58<c>v50gv45f
1320 DATA k2>gfgfdfbgce!f
1330 DATA k2[o2b<d][c>a][b<d][ca][af][ca][bf][gd][e!g][c>g][a<c]
1340 '
1350 '
1360 DATA k2fgdl8fgc4gal4f<fdl8c>bf4
1370 DATA k2r1r4r4r4>fbgf
1380 DATA k2fgdl8fgc4gal4fffff
1390 '
1400 '
1410 DATA k2fgdl8fge4b<c>l4a-<a-fl8ed->a-4
1420 DATA k2r1r4r2>a-<d->ba-
1430 DATA k2fgdl8fge4b<c>l4a-a-a-a-a-
1440 '
1450 '
1460 DATA k2v30a-ba-l8v40b<cv50e>ba-4v60l8<d-ev70fa-g-feg-fd-e4
1470 DATA k2v30l2>>g-l4fv40g-v50g-<g-v60l8fev70l4d-l8efl4a-ba-
1480 DATA k2v30l2[e.>l4a-]ba-v40l2[<e.l8>b]<ce>bl4a-v60l8[<d-fa-][ea-<c]v70l4[>f
a-<d-]l8[e>bg-][fa-<d-]l4[c>ea-][fb<d-][>ea-<c]
1490 '
1500 '
1510 DATA k2v30a-ba-l8b<ce>bv70<l4cdcl8v74dfv77gdv80c4
1520 DATA k2v30l2>>g-l4fg-l8<g-r8v70l2>bl4av74bv77bv80<b
1530 DATA k2v30l2[e.l4>a-]ba-<[el8>b]<ce>bv70l2[<g.l4c]dcv77l2[g.l8d]fgdl4c
1540 '
1550 '
1560 DATA k2v75o4l8fgv80a<cv75>bagbafg4v50l8ae!l4fadv40ad
1570 DATA k2v75l8>agv80l4fv75l8gal4<cdcv50>abfgv40fg
1580 DATA k2v75l8[fa<c][e!c>g]v80l4[a<cf]v75l8[gd>b][a<cf][l4e!c>g][a<df][e!c>g]
v50[a<c][d>f][a<c][>be!]v40[a<c][>be!]
1590 '
1600 '
1610 DATA k2l8o4fcl4dfdl8fcd4l4c>ab<c>al8b<d
1620 DATA k2>dgdgdgcfgcfg
1630 DATA k2[af][bf][af][fb][af][bf][gl8e]e![l4cf][df][gl8e]e!l4[cf][df]
1640 '
1650 '
1660 DATA k2o4c>a<cfl8edc>bl4<cdfl8gbl4fg
1670 DATA k2>cfc>fgl8abl4abagfg
1680 DATA k2[ge!][fdc][ge!]f[egb]l8[cfa][df]l4[cf][df][fa][gb]fg
1690 '
1700 '
1710 DATA k2o4fl8edc>b<l4cdfl8gbl4fgf>gf
1720 DATA k2>>fgl8abl4abagfgf<gf
1730 DATA k2f[egb]l8[cf][df]l4[cf][df][fa<c][>gb<e]>fgfgf
1740 '
1750 '
1760 DATA k2l8o4gbl4fgf>gfb<l8cfd4cfl4d>b
1770 DATA k2>c>fgf<efdc>bag<g
1780 DATA k2[gb<e!]>fgf[e>b][a<c][>b<f][cfa][dfb][cfa][dfb][fd>b]
1790 '
1800 '
1810 DATA k2o4c>gfv75gfbl8<t95ct90ft85l4dt80>bt75<et70ct35
1820 DATA k2>ce!fv70>bagfb<gcf
1830 DATA k2[ceg][e!c>g][a<c]v70[d>g][a<c][d>b][<cfa][dfb][dg>b][<eg<c][>cfa]
1840 '
1850 '
1860 DATA k2t35v75b,k2v75>b,k2v75[df]
1870 DATA end,"",""
```

SINCE 1887 YAMAHA

NIPPON GAKKI CO., LTD. HAMAMATSU, JAPAN

OMD-102 84102 Printed in Japan