December 26, 1984

SFG01 Music BIOS

Reference Manual

v1.0

December, 26, 1984

Nippon Gakki Co.,
Software Development

December 26, 1984

INTRODUCTION

This reference manual is intended to explain the utilization of the "SFG-01" Control Program version 1.0" which resides within the internal 16Kbyte ROM of the "SFG-01".

December 26, 1984

TABLE OF CONTENTS

December 26, 1984

CHAPTER I Outline

1-1 Program Configuration

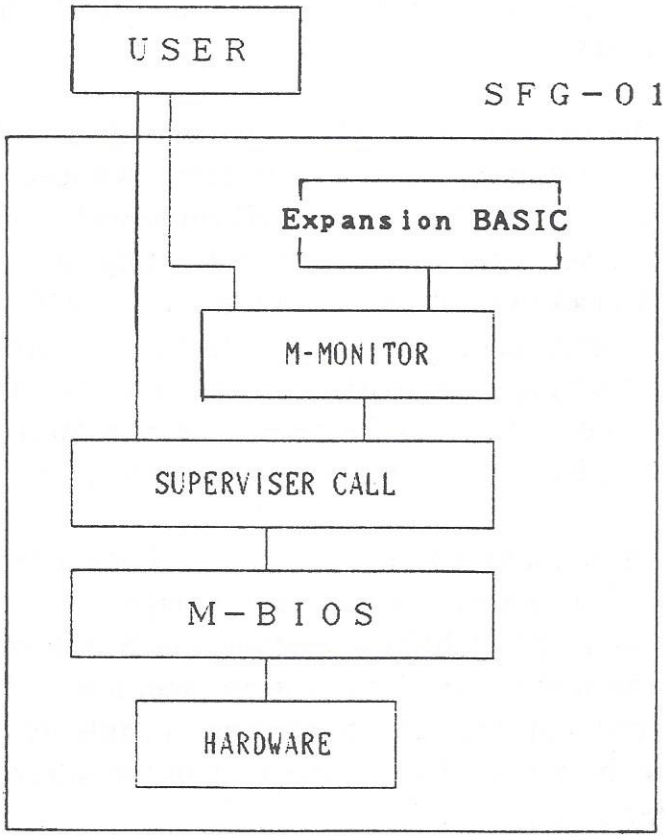
The basic configuration of this program (SFG-01 Control Program) follows that depicted in Fig. 1.1.

The M-BIOS (Music Bios) controls the hardware of the SFG-01.
As a tool box, it provides the user various basic I/O modules (MBIOS) and utilities required for instrumental sound sysnthesis and music processing.
With the use of MBIOS modules, parameter handling for sound synthesis and computer music performance can be carried out without any necessity for the user to directly access the hardware of the SFG-01.
Additionally, MBIOS includes several built-in utilities. If used, these will provide convenient supplementary services to the user that would otherwise have to be programmed by the user.
These utilities are:

1) Reocording of performance and playback
2) Automatic rhythm generation
3) Loading/saving of voice/automatic performance data onto CMT(Cassette Magnetic Tape recorder)

MBIOS functions can be invoked by issuing SV-Call (Supervisor Call) to the system.

The M-Monitor (Music Monitor) is a demonstration program that converts the MSX computer into a synthesizer, and will operate on MBIOS. M-Monitor can be invoked from BASIC, by issuing "CALL MUSIC".



Program Configuration (Fig. 1.1)

1-2 Design Concept

The MBIOS was designed with the following main concepts in mind.

First, MBIOS users should not have to worry directly about the hardware. The controls of the built-in FM sound chip, and that of midi interface, are all carried out by the MBIOS, freeing the user from this task.

Second, the concept of a virtual instrument should be defined in the user's program space. By instrument, we mean the processing system for real-time key-on/key-off requests (events).
MBIOS defines the instrument by using an IDB(Instrument Definition Block) that is linked to the built-in FM sound generator IC or MIDI interface as the actual instrumental outputs.
Thus (considering event data being input to the instrument), once the instrument is defined, the user can control the FM sound generator IC by manipulating only the event data.

Third, the slot management of the MSX system should be left up to user. This enables a multiple number of slots to be used together with the M-BIOS. In other words, the M-BIOS does not address the slots by itself (with some exceptions). This enables the user to call out the IRQ processing module of the MBIOS by switching the slot to SFG-01 , even when the interrupt is received at the user's slot.

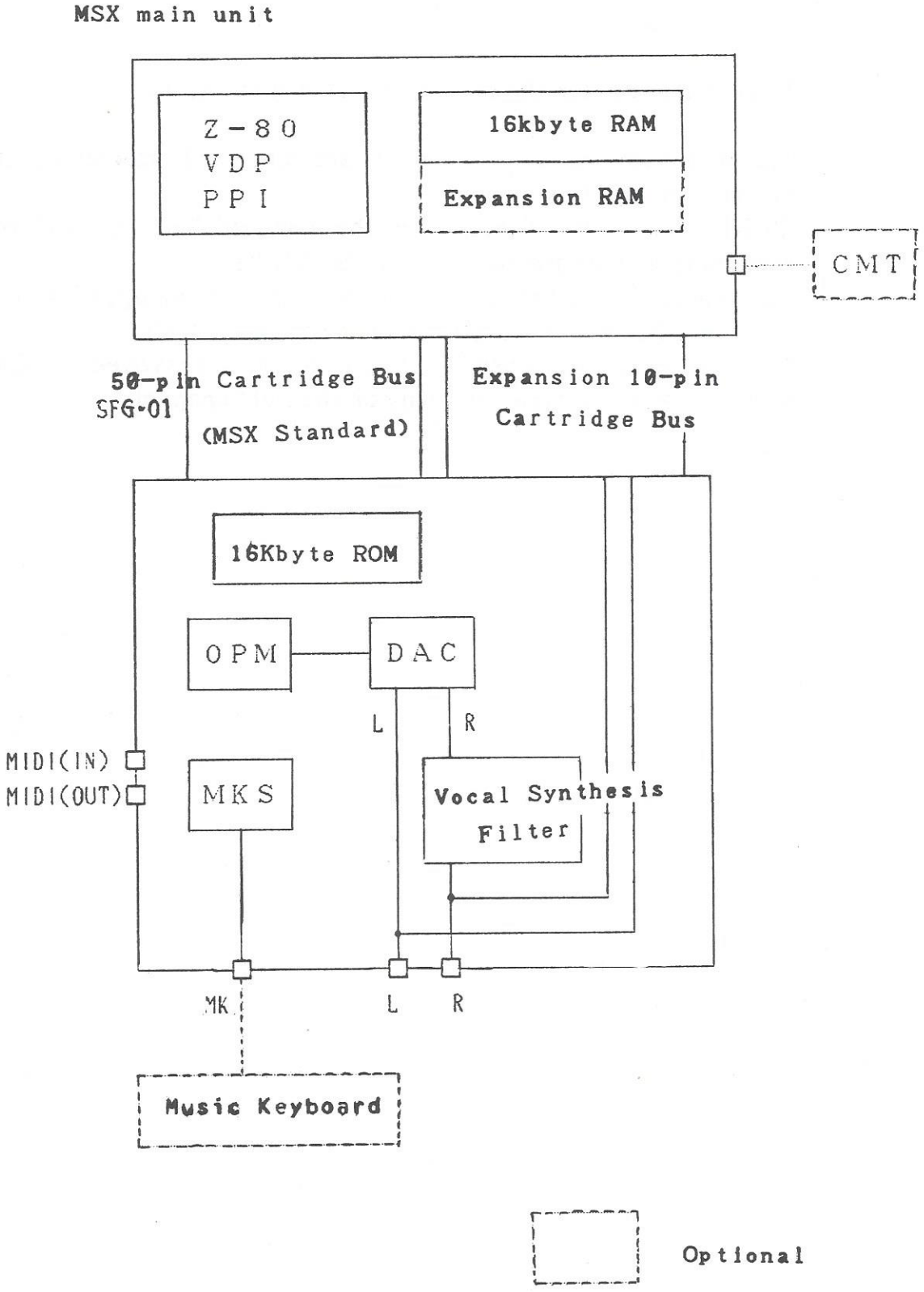
Fourth, there may be parallel prcessing by service calls.
That is, the processes of Keyboard Scanning and Instrument Performance (PLAY), as well as the real-time processes such as event buffer handling for automatic reformance, can run simultaneously by appropriately issuing MBIOS calls in the user's main program or in his interrupt routines.

1-3 Hardware Configuration

The configuration of the hardware is shown in Fig. 1.2.

The MSX main unit and the SFG-01 are connected together by the 60-pin cartridge bus.
50 pins out of the 60-pins form the standard MSX bus, and the remaining 10 pins are not in use in the SFG-01.
Additionally, the right channel of the SFG-01 is equipped with a low pass filter with a cut-off frequency around 3.5 KHz.
The filter is enabled when CSM vocal synthesis is invoked, for CSM vocal synthesis is carried out only on the right channel.

Hardware Configuration (Fig.-1.2)



1-4 Interface With MSX-BASIC

This program uses the MSX-BASIC module, HOOK, and MSX main unit hardware as shown below.

1) During M-BIOS CMT access, two HOOK's are used.
   H. KEYI (FD9Ah)
   H. ERRO (FFB1h)

   Also one BIOS call is used.
   J. STMO (00F3h)

Note that, once a hook is used, MBIOS destroys the previous contents of hook entries, and will not restore them, even after completion.

2) The first 16 bytes starting from Address 0000h of MBIOS is a header to enable the MSX-BASIC to jump to the Music-Monitor of MBIOS. (The same contents will also appear from Address 4000h and up due to the image caused by address decoding scheme of the SFG-01).

0000h (ID)	"AB"
0002h (INIT)	Address of RETURN instruction
0004h (STATEMENT)	Address of "CALL MUSIC"
0006h (DEVICE)	0000h
0008h	0000h
000Ah and after	0000h

Interfaces that fall outside the MSX interfacing standards are as follows:

3) During CSM vocal synthesis, IRQ from VDP is reset directly.

4) During CMT access, the following BASIC CMT access modules are referenced:
BLOAD (6FD7h), SRCCAS (70B8h), CSRDON (72E9h),
BPRMIC(700Bh) and CBLODL(702Fh)

5) During CMT access, the slots are switched by directly accessing the BASIC slot registers inside PPI.

6) VDP and PPI are directly accessed in M-Monitor.

7) In M-Monitor, the character font table at 1BBFh of the BASIC ROM is directly referenced.

1-5 Versions

This program contains a 9 byte version code from address 0080h.

0080h:	"MCHFM0"	Program ID code
0086h:	03h	ROM serial #
0087h:	00h	FM sound chip type
0088h:	03h	software version #

For the identification of this program from the application program, it is only necessary to look for the first 6 byte code "MCHFM0" from address 0080h.

Chapter II Basic Functions

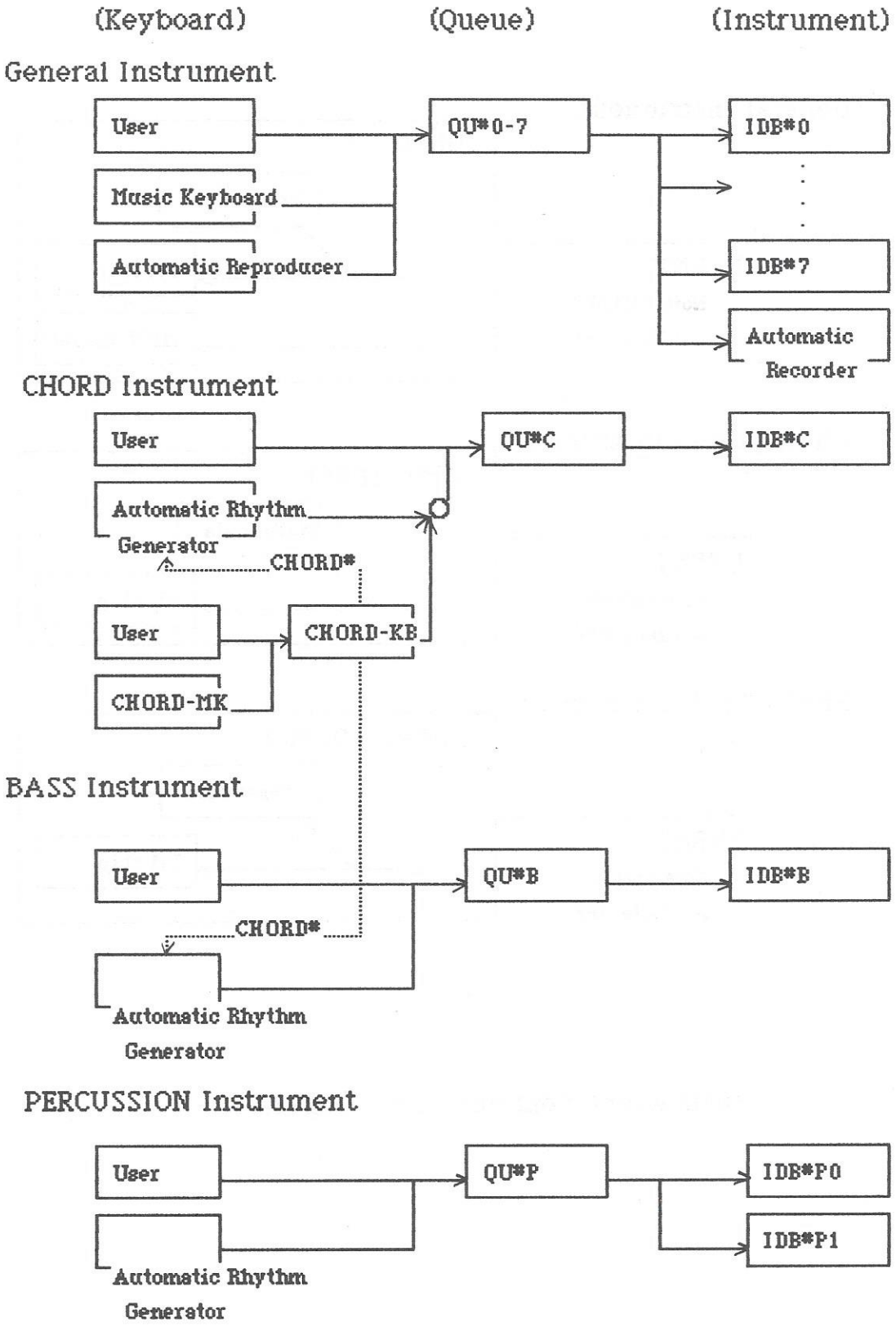
2-1 Keyboard, Queue, Musical Instrument, and Event

The Keyboard, Queue, and Instrument form the fundamental structure of M-BIOS.
Fig. 2.1 depicts the relationship of these three main functional units.

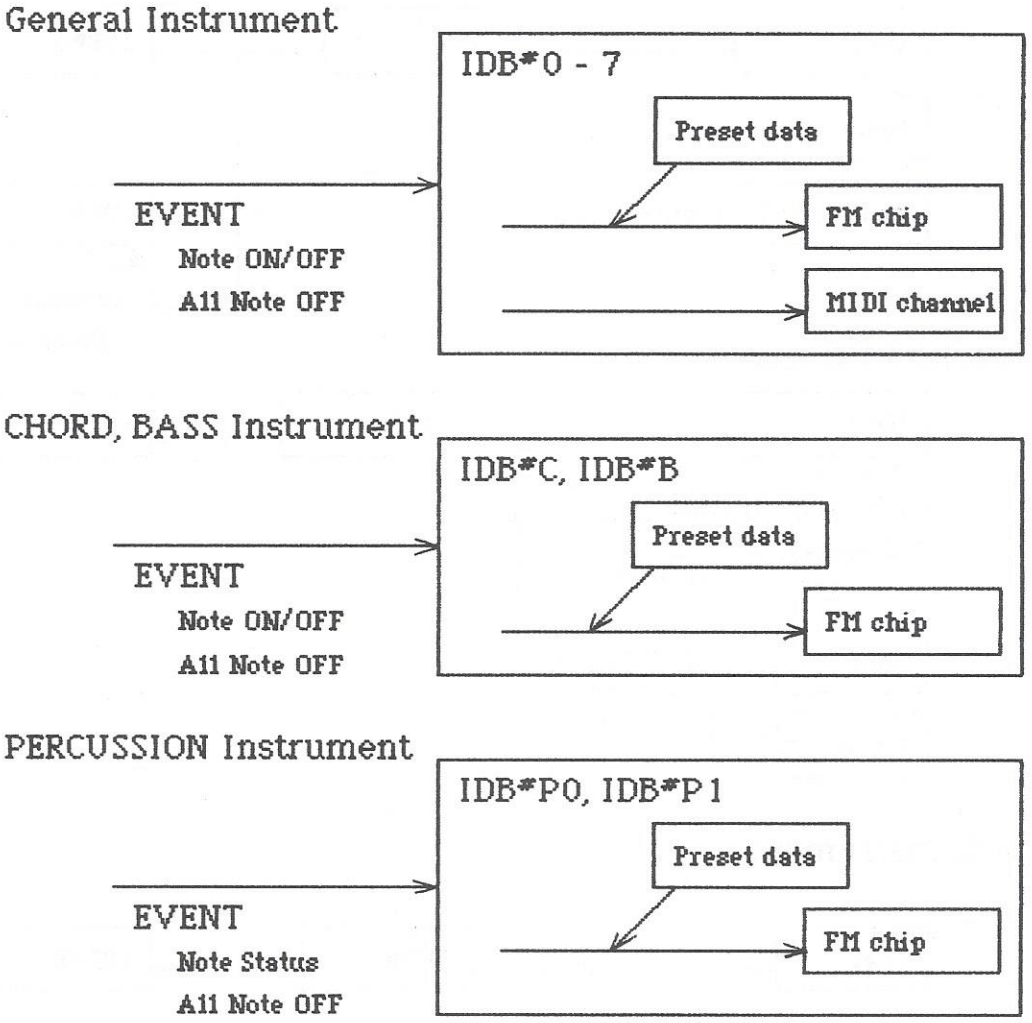
The modules on the left side of Fig. 2.1 all deals with keyboard related functions. That is, keyboards are input to the music processing system. They issue key-on/off requests and associated velocity inputs. Since the requests are time dependent, they are called events.
MK, a music keyboard attached to SFG-01 unit, is one of the keyboards.
Any automatic performance process provided by the user could also be categorized as a keyboard since it issues events to the system as well.
MBIOS not only supports the "melody" instruments, called general instruments, but also supports such "rhythm" instruments as Chord, Bass, and Percussion instruments.
Any functional modules acting as inputs to these instruments are categorized as keyboards, too.

In the middle of Fig. 2.1, Queue's are depicted.
These functional modules act as processing buffers between input (events) and the next functional modules, called Instruments.

Instruments are depicted at the rightmost portion of figure 21.
These are the function units to play the events coming from the Queue, using Instruments that are defined as IDB's.



Relationship between Keyboard,QUEUE & Instrument (Fig-2.1)



Instrument Configuration (Fig-2.2)

2-1-1 Musical Instrument and Event

In this manual, "instrument" is meant to be an event processing system.
There are 8 "general instruments" and 4 "special instruments".
The general instrument is meant to be a normal instrument that plays most of the music, accepting keyboard performance events (such as melody).
The special instrument is specifically designed for rhythmic performance. It is possible for the MBIOS user to program these rhythmic instruments by using general instruments and appropriately designing the performance buffer.
However, MBIOS already provides some useful rhythmic instruments for the user who does not want to specifically program the rhythm instruments.
These are: Chord, Bass, Percussion #0, and Percussion#1 instruments.

The instruments are defined by the control block, called IDB (Instrument Definition Block).
8 general IDB's are referenced as IDB#0, IDB#1,IDB#2,... , IDB#7.
Chord, Bass, Percussion are referenced as IDB#C, IDB#B, IDB#P0, and IDB#P1 respectively.
Hereafter, depending on the situation, the instrument may be refered to as IDB.

The CSM vocal synthesizer, to be discussed later, is also defined by IDB, and is referenced as IDB#CSM.

IDB#	=0 thru 7	IDB#0 to IDB#7
	8	IDB#C
	9	IDB#B
	10	IDB#P0
	11	IDB#P1

Fig. 2.3 IDB numbers

2-1-1-1 General Instruments

The general instruments (IDB#0 - IDB#7) are the most commonly used instruments other than Chord, Bass and Percussion instruments.
They are defined by the allocation of the channel usage of FM sound generator IC and MIDI channel, as well as by the preset data for the FM sound chip.
Input to IDB is event data. It is retrieved and processed by the P-Call from the single linked Queue to the IDB.

There are three events: Note-on, Note-off, and All-Note- off.
Note-on and Note-off are each comprised of 2 bytes of data.

	bit15 8 7 0		
Note-on	1	KC#	VELOCITY
Note-off	0	KC#	0

All-note-off (no data)

KC#(Keycode number) indicates an 8-octave range, with the note name being represented by the lower 4 bits, and the octave by the upper 3 bits. This KC# format is internal to SFG-01 , and different from MIDI keycode representation.

	bit7 3 0		
KC#	0	Octave	Notename

<0h> C#	<8h> G
<1h> D	<9h> G#
<2h> D#	<Ah> A
<3h> -	<Bh> -
<4h> E	<Ch> A#
<5h> F	<Dh> B

<6h> F#	<Eh> C
<7h> -	<Fh> -

While the SFG-01 internal KC# looks as shown above, the MIDI KC# is a linearly arranged number, with 00h being the lowest note and 7fh the highest note.
Some examples between internal format and that of MIDI are depicted here:

Note	Internal KC#	MIDI format KC#
C#-1	00h	0Dh
A4	4Ah	45h
C3	7Eh	6Ch

The SFG-01 internal velocity code covers an 8-bit range between 00h (Minimum) and FFh (Maximum).
Observe that this again is different from MIDI velocity format. MIDI velocity can be derived from internal velocity format by dividing internal code by 2.

When the events are transmitted from MIDI interface, they are attached with additional information to indicate MIDI destination channel number, KC# converted to MIDI format, and velocity that is converted to MIDI format as well.
They are as following:

Note-on

1001	MIDI#
0	MKC#
0	MVelocity

Note-off

1000	MIDI#
0	MKC#
0	1000000

All-note-off

1 0 1 1	MIDI#
0 1 1 1 1 1 1 0	
0 0 0 0 0 0 0 0	
1 0 1 1	MIDI#
0 1 1 1 1 1 1 1	
0 0 0 0 0 0 0 0	

Where MIDI# is a destination MIDI channel.
MIDI#=0 - 15 for midi channel # 1 to 16 ,respectively.
MKC# and MVelocity stand for KC# and Velocity in MIDI format, respectively.

In event processing of IDB's, users are advised to note the following consideration. MBIOS performes special processing when channels alloted to the IDB are all in use and still there is a request to the channel of that IDB.

That is, the FM sound generator IC has 8 channels (8 notes), out of which as many as desired up to the maximum of 8 channels can be allocated to IDB. However, when the request for note-on events exceed the available channels declared in IDB, the processing will differ depending on whether only one channel is being used, or if two or more channels are being used.
When only one channel is being used, and if the second note-on request is issued while the first note is still on, then the second note-on is granted, stopping the first note and pushing it into the stack.
Then, the note-off of the second key will enable the first key to be popped back and keyed on again(two level stacking system).

When two or more channels are being used, and if the last note-on request is made while all 8 channels are used up, the first note will simply be keyed off, and the last one will be granted.
Stacking will not occur (last note priority system).

2-1-1-2 Chord Instrument

The Chord Instrument (IDB#C) is a pre-constructed instrument by the fixed allocation of the FM sound generator IC channel (fixed to channel 3 and 4) and by the corresponding preset data.
Events are retrieved fromQU#C and processed by P-CALL.
Based on the Chord# given by the corresponding event, 3 notes (4 notes for a 7th) will be generated simultaneously for the Chord instrument.
There are 3 events (Note-on, NOte-off, and All-note-off) supplied with CHORD#.

	bit15 8	7 0
NOTE ON	1 0 Chord#	0 0 0 0 0 0 0 0
NOTE OFF	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
ALL NOTE OFF	(no data)	

The Chord# is comprised of 4 bits for the name of the root note, and 2 more bits to indicate whether the cord is major/minor and the existence of the 7th note.
Actual pitch of the root note is set between KC#=35h(octave 3,F) and KC#=44h(octave 4,E). This is equivalent to the range between 35h and 40h in MIDI key code.

	bit5 4	0
Chord#	x x	note name

<XX>
<00>Major
<01>Minor
<10>Major 7th
<11>Minor 7th

Once the root name is specified, associated development for a major chord will be, +386, +700, +1016 cents, and +300, +702, +1002 for a minor chord.

Additionally, Chord instruments always handle events with last-event priority, and single trigger processing.

2-1-1-3 Bass Instrument

The Bass Instrument (IDB#B) is made up by the allocation of the FM sound generator IC channel that is fixed to channel 5, and by the corresponding set-up data.
The event will be retrieved from QU#B and processed by the P-CALL.
There are 3 events: Note-on, Note-off and All-note-off.
These are supplied with a Chord# and a Pointer# for the bass-line pitch offset table.

	bit15 8	7 0
Note-on	1 0 Chord#	0 0 0 0 0 Pointer#
Note-off	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
All-note-off	(no data)	

The Pointer# is a 3-bit pointer (displacement) to the entry of the bass-line pitch offset tables where relative pitch offset to the root note is held.
There are two such tables, one for a major and the other for a minor chord.
When the Pointer# is specified, the contents of the table addressed by the Pointer#(pitch offset) are added to the root note indicated by the Chord#. Therefore, by issuing Chord#/Pointer# information sucsessively with appropriate intervals, it is possible to produce the bass notes within the chord frame described in the bass-line pitc offset table.
For a bass instrument, the pitch range of the root note of the Chord#
is set between KC#=25h(octave 2, F) and KC#=34h(octave3,E).
This is equivalent to the range between 29h and 34h in MIDI key code.

Additionally, this instrument handles events with last-event priority, and single trriger mode.

2-1-14 Percussion Instrument

The Percussion Instruments (IDB#P0, IDB#P1) are handled together in pairs, and are made up by the allocation of the FM sound generator IC, whose channels are fixed to channel 6 for IDB#P0, and fixed to channel 7 for IDB#P1, and by the corresponding preset data. Events are retrieved from QU#P and processed by P-CALL.

IDB#P0 provides the functions of hi-hat open (HHO), hi-hat close (HHC) and hi-tom(HT).
IDB#P1 provides those of the bass drum (BD), and the low-tom (LT).
The corresponding events are Note-status and All-note-off.

	bit15															0
Note-status	x	x	x	x	0	x	0	0	0	0	0	0	0	0	0	0

bit 15	HHC (trigger)
bit 14	LT (low tom on)
bit 13	BD (bass drum on)
bit 12	HT(high tom on)
bit 10	HHO (hi-hat open on)

All-note-off no data

When Note-status data is issued, it is compared with the previous Note- status.
For the bits that have changed from 0 to 1, the corresponding instruments are triggered.
For the bits that have changed from 1 to 0, the corresponding instruments are shut off.

Also HHC and HHO use the same IDB. The availability of each instrument is mutually exclusive, with HHC receiving priority if both are requested simultaneously.

2-1-2 Queue

This section is illustrated in the middle portion of Fig.2.1.

Queue is a process to handle the queue buffer that is capable of accepting up to a maximum of 16 events.
There are queue's for 8 general events (QU#0, ... , QU#7) and 3 specialized events (QU#C,QU#B, QU#P, for Chord, Bass, and Percussion respectively).
Queue's are named as QU#x, and refered as.

QU#x	
0 thru 7	QU#0 thru QU#7
8	QU#C
9	QU#B
10	QU#P

Queue's primary function is to buffer the event flows between the keyboard and the process of playing IDB.
It ensures the asynchronicity of two independent processes (keyboard handling and playing IDB's) running simultaneously in the program space. Working as a FIFO, this effectively increases the data processing rate without having to miss the note or seriously delay the performance timing.

The Queue also merges the event data from a multiple number of keyboards, and distributes them to a multiple number of IDB's that are linked to a queue.

The All-note-off event erases presently queued events from the queue , and sets the All-note-off flag in the queue.

2-1-3 Keyboard

This section is illustrated in the leftmost portion of Fig. 2.1.

The keyboard is, in effect , the event generator.
The external keyboard attached to the SFG-01 (MK) is a music keyboard.
An automatic performance buffer is treated as a music keyboard, too.
For "keyboards", like MK or automatic peformance buffer, events are then input to QU#0-QU#7.
The automatic rhythm generator and the user provided rhythm buffers are considered as special keyboards, whose events are routed to QU#C, QU#B, and QU#P.
The attached keyboard (MK) has a Chord-sensitive range, from which CHORD-MK is defined.

CHORD-KB is a special place to hold a once-depressed key code within CHORD-MK (or in user-simulated chord-mk). The purpose of holding such a key code is to provide the chord# to the rhythm generator (chord and bass) , and it holds the chord# even after the key-off code is generated.
Note that certain data is routed to the auto-rhythm or the bass line generator.
There is an SV-call available to change the content of the code name held in CHORD-KB.
It is used to change the chord# in CHORD-KB (providing it to rhythm generators such as chord and bass). CHORD-KB is linked to QU#C and maintains the key-off state.

Output of CHORD-MK is ,as explained in section 2-1-1-2, threefold: Note-on, Note-off, and All-note-off ,which all affect QU#C.

2-2 Major parameters by which the FM sound generator IC creates sounds

The simulation of instruments by the internal FM sound generator IC can be set up completely in accordance with the user's requests. This section covers topics relating to the setup parameters for voicing. To drive the sound generator IC, first of all it is assumed that the readers of this manual are already familiar with the hardware specification of FM sound generator IC (OPM).
That is, there are 8 channels available in the FM sound generator IC, and only one hardware LFO is provided.
For details concerning the tone data, refer to the "YRM-102 FM Voicing Program Reference Manual".
Also, the effects and principles of the major voicing/performance parameters are covered in this section.
In order to access the desired parameters, however, it is necessary to modify some of the variables in the IDB and MIDB, or to issue appropriate S-calls to MBIOS. For details , refer to Chapter IV, MBIOS syntax.

2-2-1 Portamento, Trigger, and Sustain Mode

The performance mode is further divided into three main modes, Portamento mode, Trigger mode, and Sustain mode.
The first two modes are effective only for general instruments with a single tone generator. Sustain mode can is available for all general instruments.

Poratamento is determined by the Speed and the Mode. The Portamento Speed determines the rate of pitch shift during the portamento, with the Speed=0 being equivalent to no portamento effect at all.
There are two Portamento Modes, Full Portramento and Fingered Portamento.
In Full Portamento, the Portamento will take effect for any Note-on, and in Fingered Portamento, it will take effect only while key stacking is taking place. The stacking will take place when only one channel is assigned to the instrument, and when the first key still being pressed is pushed onto the stack by the second key-on, or when the first key is popped out of the stack by the release

of the second key.
Though the portamento is normally modified by IDB variables, since MBIOS executes the portamento by using Clock-A of the sound generator IC, changing the CLOCK-A interval value in MIDB from the initial setting (8000h) will also cause the portamento speed to change.

The trigger type can be determined by using the Trigger Mode.
By trigger, we mean that the envelope is generated from the very beginning at the key-on request.
A trigger will be generated for every Note-on event during the MULTI Trigger mode. However, in the case of a single channeled instrument, it is possible not to generate a trigger during key stacking, thus continuing the envelope when the second key is on , and only changing the pitch accordingly. This is called Single Trigger mode.

Sustain controls the release rate (RR) of the envelope after Note -off (mostly to lengthen the release time).
When Sustain mode is off, the release rate for each operator will be computed according to RR of the voice as normal. However, if Sustain mode is on, MBIOS will output the Sustain rate (RR) contained in IDB to all the operators whose channels are involved in Note-off events.
Thus , it is possible for the user to write his own Sustain handling routine under the Sustain mode by simply modifying the contents of Sustain rate of theIDB.

2-2-2 KC Range

It is possible to set the KC sensitivity range of the IDB, where only KC#'s within that range are only accepted.
Any deviation from this range in the Note-on event will not be accepted. However, the Note-off events are not limited in this manner. All-note-off events will be accepted and appropriate Note-off processing will be carried out.
This function is useful to implement a register sensitive keyboard, such as split keyboard.

2-2-3 Transposition

There are 3 ways to realize transposition; the transposition of the entire system, the transposition of an individual IDB, and the transposition by means of altering voice data.
Transposition of the entire system is obtained by modifying the Master Transpose variable in MIDB.
Individual instruments can be transposed via the Instrument Transpose variable in the IDB.
Transposition by voice data is a pitch-shift that has been preprogrammed in the voicing data by use of the YRM102 voicing program. The primary purpose of this feature is to include the "pipe" length of the instrument (8',16',1 3/5',etc.) in the voicing parameters.
When transposition via IDB is attempted for the IDB that has been enabled by pitchbend, the transposition will not be accessible by the user.
This is because the pitch bend function uses the Instrumant Transpose function to accomplish pitch bend.
Also Pitch Bend Depth by IDB is a single parameter used for the entire system so that it affects all the instruments with the same amount of pitch bend.
The Pitch Bend Depth can be programmed from +/- 1 half tone to +/- 1 octave with half-tone resolution.

2-2-4 Volume, Brilliance

Volume and Brilliance are both OL (Output Level) offsets, with Volume being the offset for the carrier, and Brilliance that for the modulator.
Volume is a parameter that exists for every instrument, and can be adjusted for every carrier of the instrument.
However, there is only a single Brilliance parameter within the entire system, and this is applied to only the Brilliance-enabled modulators.
The range of both Volume and Brilliance offsets against OL's is between 0 dB (MAX) and -48 dB (MIN).

2-2-5 LFO

The operation of the LFO is determined by the Speed, Waveform, AMD and PMD commands.
Speed sets the frequency of the LFO.
AMD sets the output level of the LFO for amplitude modulation, and PMD sets the output level of the LFO for pitch modulation.
As depicted in the routing of the LFO, in Fig. 2. 4, LFO outputs adjusted by AMD and PMD are common to all the instruments.
AMS and PMS set the sensitivity to LFO modulation for each individual instrument.
The Triggered Sync function initializes the LFO phase to 0(zero) in synchronization with Note-on events.

2-2-6 Noise

The OPM has a noise generator which is started when operator 3 of channel 7 receives a key-on command.
When a Noise Enable is used for the IDB that uses only channel #7, noise will be generated at the given Noise Frequency.
Noise Frequency controls the period that quasi-random number series repeats by itself. It should be set appropriately so as to obtain sufficiently long series of random number generation.

2-2-7 Velocity

Velocity refers to the touch intensity at Note-on time (Initial Touch), and affects an offset level for the OL of each operator.
The central value (80h) of the Velocity is used as the normal setting.
When the Velocity increases, the volume gets louder if the carrier is sensitive to the velocity, and the sound gets brighter if the modulator is sensitive to the velocity.
The Velocity Depth covers the effective range of the effect caused by Velocity. Depth is adjustable for the carriers over a range of +-12 dB, and +-6 dB for the modulators.

2-2-8 Envelope

The Envelope can be set independently for each operator.
As shown in Fig. 2. 5, the shape of the Envelope is determined by AR (Attack Rate), D1R (1st Decay Rate), D2R (2nd Decay Rate), RR (Release Rate), and by SL (Sustain Level) between the 1st Decay and 2nd Decay.
OL (Output Level) offsets the standard level of envelope.
The actual OL commanded in the FM sound generator IC is a sum total of such offsets as OL (original offset), OL Adj (adjusted offset for algorithm difference), Keyboard Scaling (Keyboard scaled offset), Velocity(Velocity offset) and Volume/Brilliance(offset due to volume or brilliance control).
Note that all the offset amounts are in terms of attenuation from a 0 dB standard.
The offset range varies depending upon the purpose of usage. For example, OL (original offset) can be set over a range of 0 dB to -96dB, while OL Adj can be set over a range of 0 dB to -12 dB.

2-2-9 Keyboard Scaling

There are two types of keyboard scaling.
One is for the rate(AR,D1R,D2R,RR) of the envelope. The keyboard rate scaling is carried out by hardware internal to the FM sound generator IC.
The higher the KC#, the faster the rates. Depth determines the amount of keyboard rate scaling.
The other type of keyboard scaling is that for level. Unlike keyboard rate scaling, this is accomplished by MBIOS.
KS selects two types of keyboard scaling curves. (See Fig. 2.6).
When a scaling value corresponding to KC# is taken out of the curve, it is multiplied by the Depth to yield the key-scale dependent offset (adjusted by Depth in effect). Depth serves as a sensitivity adjustment for level scaling. (Up to -24 dB).

2-2-10 Other Functions

Multiple (Harmonic number, indicated as an F in the FM voicing program) creates integer multiples (1/2,1,2,3,...,15) of the keyboard

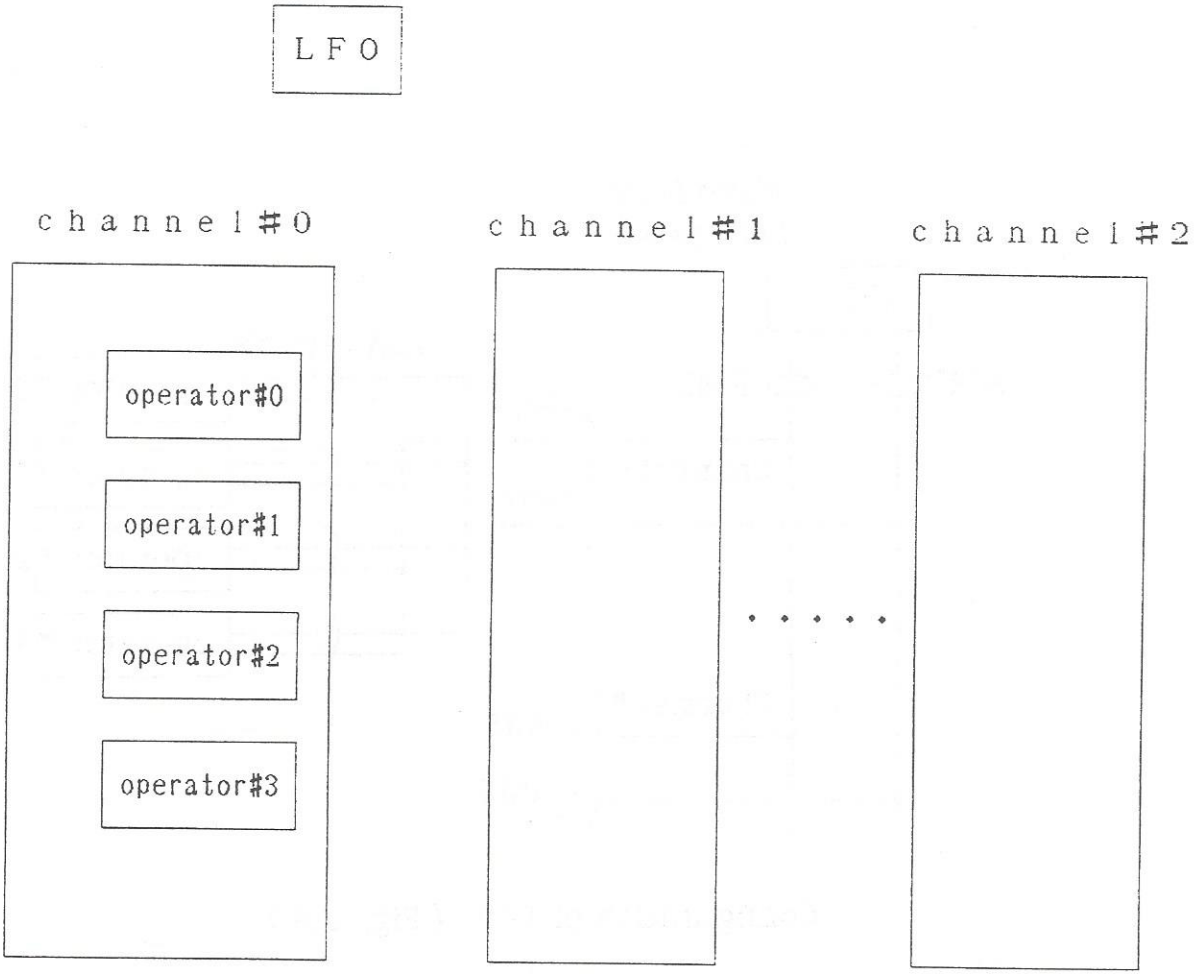
pitch for each operator.
The ratio between the Multiple of the carrier frequency and the Multiple of the modulator frequency plays an important role in determining harmonic structure of the sound.
DT2 (Detune#2/Inharmonic) is used to create inharmonic multiples of the keyboard pitch.
It is useful to create an inharmonic pitch for sound such as a gong, bell, etc.

DT1 (Detune#1/Fine) is used to shift the pitch of the operator slightly out of tune. Detuning is useful to obtain a chorus effect, or richer sounds.

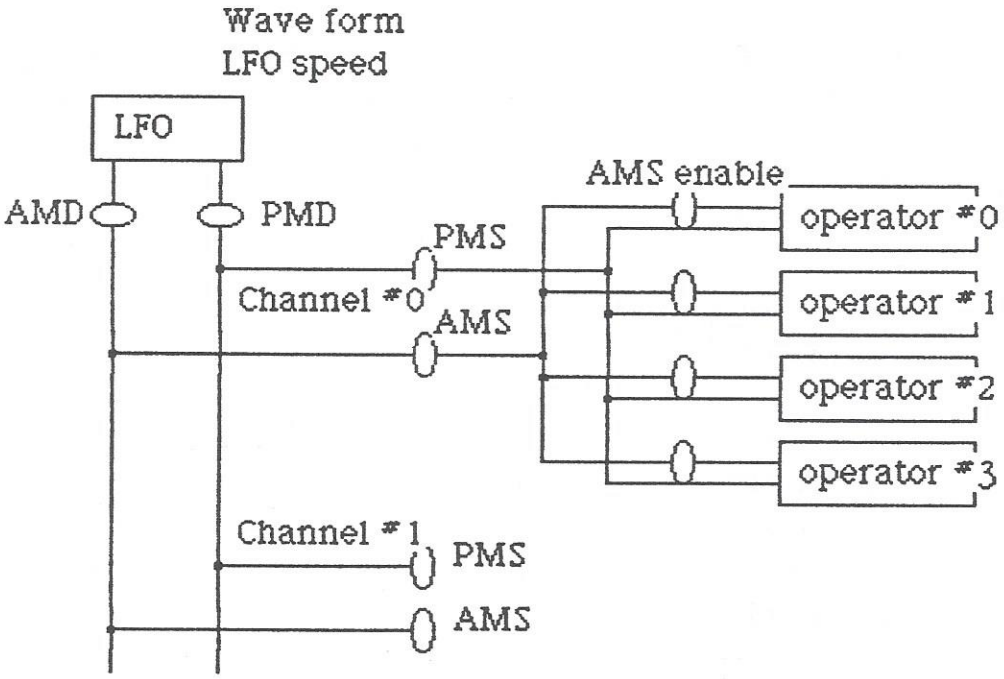
The Feedback Level adjusts the amount of feedback to the first operator (from itself) of each channel over a maximum range of up to 4 *PI radians.
This is useful to enrich the upper harmonic structure of the sound caused by the 1st operator.

Algorithm determines how the operators are connected together.
There are 8 ways to connect the 4 operators in each channel.

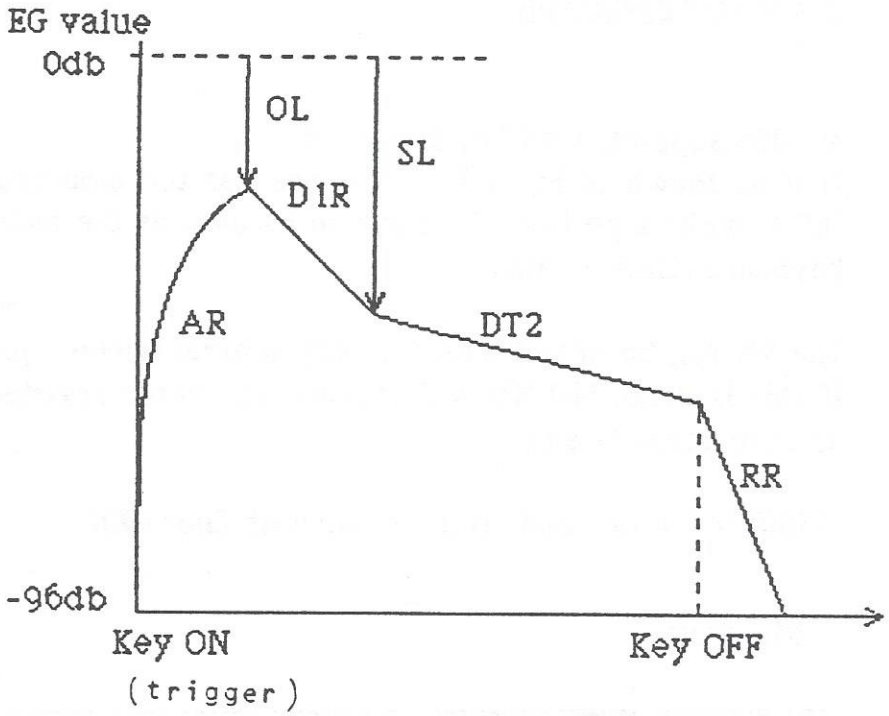
Stereo L/R is an output-enable function that allows the output to be routed to either the left, right, or both channels, as desired.



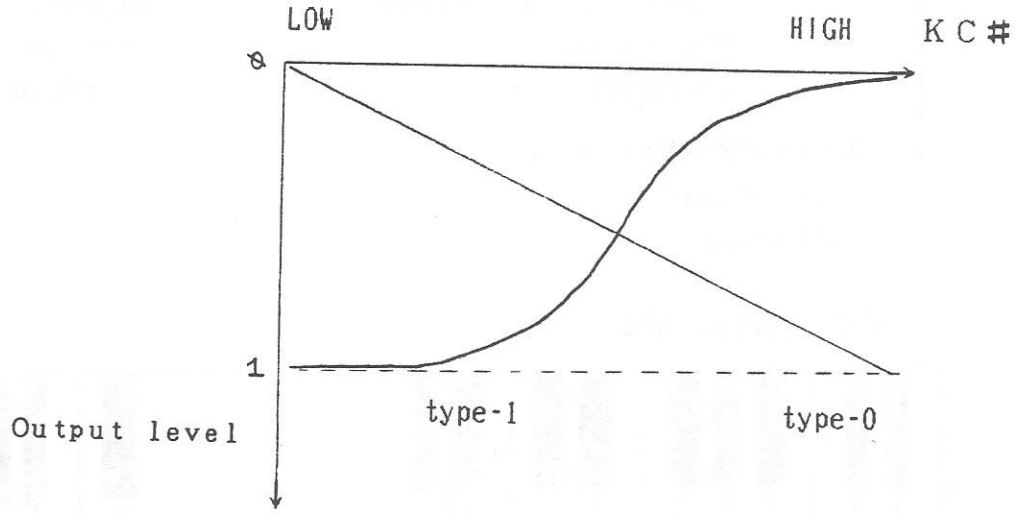
Configuration of OPM (Fig. 2.3)



Configuration of LFO (Fig. 2.4)



Envelope (Fig. 2.5)



Keyboard Scaling (Fig 2.6)

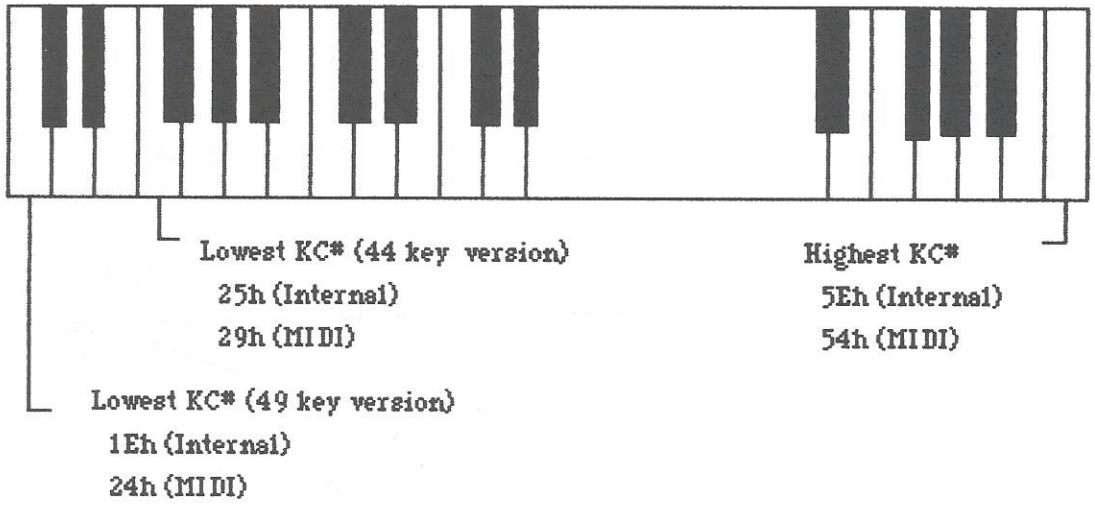
2-3 MUSIC KEYBOARD

M-BIOS supports a 49 key keyboard.
It is as shown in Fig. 2.7. It is used as the mounted keyboard (MK), while a portion of it can also be used as the mounted chord keyboard (CHORD - MK).

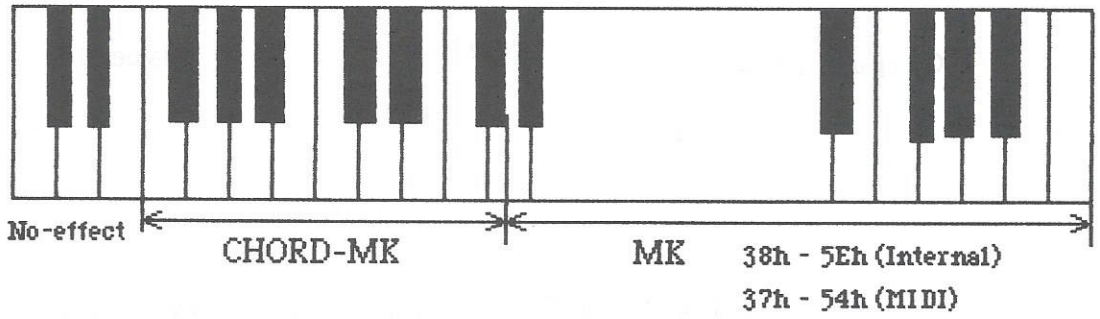
The MK can be linked with the only general Queue, QU#0 - QU#7. If this is done, M-BIOS will register all events registered by MK into the linked Queue.

CHORD-MK, if invoked, is linked up with Chord-KB.

MK only



MK + CHORD-MK



(Fig-2.7)

2-4 Auto Rhythm Generator

MBIOS provides a ready-made automatic rhythm generator.
The processing is carried out via two types of rhythm buffers, called RHB (RHythm Buffer).
One is for auto chord and bass-line-performance.
Another is for percussion instruments.
MBIOS ca define up to 16 sets of RHB's (2 RHB's per set, one chord/bass RHB and one percussion RHB).
Of the 16, 6 sets are already used by MBIOS as preset auto rhythm patterns.
The patterns are:

0	16 beat	(8/4)
1	slow rock	(4/4)
2	waltz	(3/4)
3	jazz	(8/4)
4	disco	(4/4)
5	swing	(8/4)

When a user-programmed RHB is to be used, he has to set the pointer to his RHB in the MIDB prior to its usage.
There are pointer tables for RHB's in the MIDB. MBIOS preset patterns are pointed to in the first 6 entries of the tables.

RHB is a 96 byte buffer in which 96 events can be programmed. Each event , or byte, is supposed to last for 1/48 note duration. Thus the entire RHB will have the maximum length of 96 times 1/48 note duration: that is, 8 quarter notes long.
In the MIDB, the time signature of the rhythm performance should be specified.
Three kinds of time signatures are available.

TIME SIGNATURE:

designation	time
0	4/4
1	8/4
2	3/4

In the following, the RHB patterns represented in terms of quarter notes are depicted for each time signature.

4/4	1	2	3	4^	1	2	3	4
8/4	1	2	3	4	5	6	7	8^
3/4	1	2	3^	3	1	2	3	3

When the RHB is played, performance will be repeated from the top of the buffer again when the beat marked by ^ (hatted beat) is completed.
Although the pattern remaining in the RHB after the hatted (^) beat is not played, it is recommended to fill the RHB with the above pattern entirely. This makes possible the switching of time signature in the middle of an auto rhythm generation.

The event format to be filled in the RHB is as following:

For chord and bass:

bit 7 0

*	*	*	*	0	*	*	*

bit0-2	walking bass table pointer disp.		
bit4-5	BASS	‹00›	No operation
		‹10›	Note-off
		‹11›	Note-on
bit6-7	CHORD	‹00›	No operation
		‹10›	Note-off
		‹11›	Note-on

For percussion:

bit7 0

*	*	*	*	0	*	0	0

bit2	HHC trigger
bit4	Low tom
bit5	Bass drum
bit6	Hi tom
bit7	HHO

When starting the auto rhythm, two options are available.
The "immediate start" will start the auto rhythm immediately upon issuing of the start auto rhythm command.
When "Sync-start" is specified, MBIOS awaits an event occuring on either CHORD-MK or MK, and then the auto rhythm starts.

To keep auto rhythm running, it is also required to provide a clock to MBIOS, via a clock command.

2-5 CSM Vocal Synthesis

The vocal synthesis supported by MBIOS is based upon the technique called CSM (Composite Sinusoidal Method).
It simulates the spectrum characteristics of the human voice by the generation of a few sine waves of different frequencies.
In the case of MBIOS, 4 operators (4 sine waves) approximate the spectrum envelope of the voice.

$$y(t) = \sum_{i}^{3} Ai * E(t) * \sin(w i * t + \varphi_i)$$

Operator #1 of channels#0 through 3 are used to implement the above sine waves.
CSM vocal synthesis data is divided into overall data and frame dependent data. "Window" is a time frame (approximately 20 ms) used to analyze CSM parameters , and it will be used as an interval to reconstruct the voice.
The overall data includes the envelope, E(t).
The frame dependent data includes the frequency, wi, amplitude, Ai, and pitch information of the vocal sound. Pitch can be obtained by the interval of resetting the sine wave generation, causing the pitch-dependent harmonic components to spread around the formant frequency wi.

When the CSM driver is active, due to heavy IRQ traffic, MBIOS suspends all other processes and concentrates on only CSM synthesis.
The CSM driver also requires preset data to be loaded into the FM sound generator IC. Hence IDB#CSM is used. The format of IDB#CSM is identical to that of any other IDB. However, except for voice # 46 in the voice library (and hence can not be modified) there is no user processable data in the IDB#CSM.

2-6 Voice Library

Voice parameters used to simulate an instrumental sound are handled together and packed into 64 bytes of data (48 bytes of data in the case of the system preset library).
Its map is shown in section 4-10.

The voice library holds these voice parameter sets.
It has a capacity of 48 voices in MBIOS (called SVL, System Voice Library), and can be expanded to another 48 voice area in user RAM called as UVL (User Voice Library).
Voices are refered to by number; 0 - 47 for SVL voices, and 64 to 111 for UVL voices. Voice numbers 48 - 63 are reserved.
Although it is possible to address all the voices, MBIOS assumes the following voices of SVL are special voices dedicated to special functions.
That is; voices 36 - 39 for IDB#C, voices 40 - 41 for IDB#B, voice 44 for IDB#P0, voice 45 for IDB#P1, and voice 46 for IDB#CSM.

MBIOS is equipped with a CMT (cassette tape) utility to save UVL and load it back to UVL. In this case , the file name to transfer is permanently fixed to "VOICE".

The following is a directory of SVL.

Content of SVL(System Voice Library)

0* BRASS 1	16* PICCOLO	32* TRAIN
1* BRASS 2	17* OBOE	33* AMBULAN
2* TRUMPET	18* CLARINE	34* TWEET
3* STRING1	19* GLOCKEN	35* RAINDRP
4* STRING2	20* VIBRPHN	36 RM.BRAS
5* EPIANO1	21* XYLOPHN	37 RM.FLUT
6* EPIANO2	22* KOTO	38 RM.GUIT
7* EPIANO3	23* ZITAR	39 RM.HORN
8* GUITAR	24* CLAV	40* R1.BASS
9* EBASS 1	25* HARPSIC	41 R2.BASS
10* EBASS 2	26* BELL	42 SNAREDR
11* EORGAN1	27* HARP	43 COWBELL
12* EORGAN2	28* BEL/BRA	44 PERC 1
13* PORGAN1	29* HARMONI	45 PERC 2
14* PORGAN2	30 STEELDR	46* CSM
15* FLUTE	31* TIMPANI	47 (Undefined)

* Enables the loading of LFO parameters when used.

2-7 Recording and Playback

MBIOS supports the recording of events retrieved from a queue buffer, or playback of the recorded data.
However, since there is only a single buffer available, it is not possible to do both functions at the same time.

Prior to calling recording or playback function,, the user must provide a buffer where bulk of event data is stored or retrieved. The name of this buffer is EVB (Event Buffer). There is a service call available to tell MBIOS where the EVB is going to be.

Both recording and playback will be automatically finished when the end of the EVB is reached. Recording will also be terminated when an ALL-note-off from a corresponding Queue is processed.

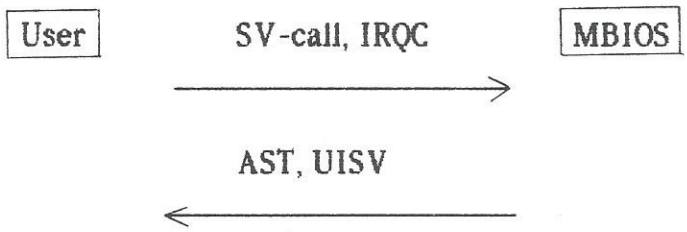
MBIOS also provides CMT transfer of the EVB to save or load with a cassette recorder.

Chapter III MBIOS Interface

3-1 User Interface

MBIOS control is handled via the SV-call (supervisor call) and IRQC (IRQ-call).
On the other hand, MBIOS can call the user via AST (Asynchronous System Trap) and UISV (User Interrupt Service Vector).

The genral format to transfer data between the MBIOS and the user program is by means of registers and tables (or buffers).
The latter include the MIDB (Master Instrument Definition Block), IDB (Instrument Definition Block), EVB (Event buffer), RHB (Rhythm Buffer) and UVL (User Voice Library).
These are the buffers that are accessible in the program by both MBIOS, and the user program.
There also exist some temporary buffers used in SV-call processing only during the specific SV-call routine.



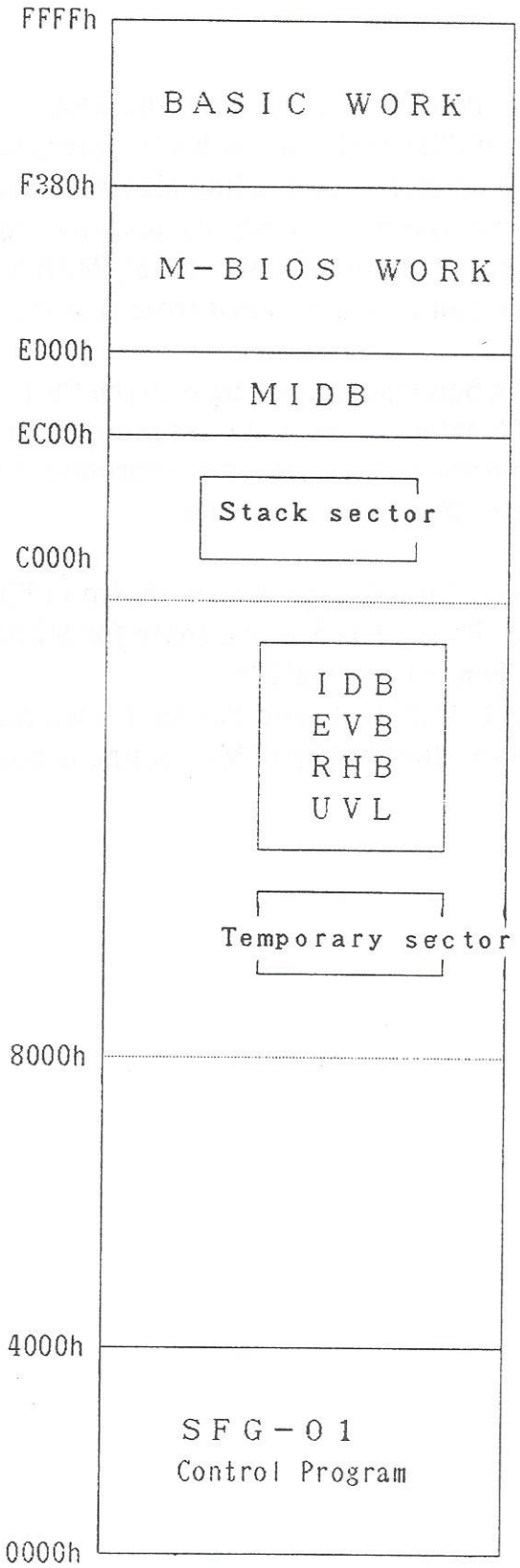
3-2 Memory management

The management of the cartridge slot is left up to the user.
For example, assume that 0000h-3FFFh of the BASIC interpreter is mapped in front, and when an interrupt needing MBIOS service just arrives to the system. It is then the user's responsibility to switch the slot so as to map the MBIOS slot in front, then get IRQC service by MBIOS, and finally switch back to the original slot in order to exit from that interrupt.

However, as an exception, when loading/saving into the CMT, upon the CMT service call request, MBIOS switches the slot (assuming the primary slot#0 for BASIC only) by itself appropriately to appropriate the CMT service provided in the BIOS of BASIC interpreter.

The memory allocation under MBIOS control is as shown in FIG. 3.1.
The area from ED00h to F37Fh is a fixed work space for MBIOS.
The area from EC00h to ECFFh is for the MIDB.
Other areas such as IDB, EVB, RHB, UVL, and the stack area can be allocated anywhere between 8000h (or 4000h when CMT routine is not used) and EC00h.

Memory arrangement (as seen M-BIOS) (FIg. 3.1)



3-3 Supervisor Call

There are 6 different SV-call's available, as follows:

I-call	(Initialize)
R-call	(real time)
K-call	(Music keyboard)
P-call	(Play)
S-call	(Set up)
M-call	(Receive MIDI)

Of the above, R, K and S require arguments to be transferred.
Once issued, an SV-call will not return to its call-source until its processing has been completed.
However, it does not mean that SV-call's have to be issued one after the other , waiting for the previous one to finish.
The above 6 call's can be issued simultaneously under certain conditions.
This feature enables the parallel processing of music events.

That is, while P and K calls are being processed, R and M calls also can be issued.
To do this, the system was designed that P-calls and K-calls will function in either interrupt-enabled or disabled conditions.
The other SV-calls will run properly, only if interrupts are disabled.

I-call (Initialize)

---

calling sequence:

```
DI
IM1
CALL 0090h
```

register conditions:

	in	out
[A]	-	*
<C>	-	*
[BC]	-	*
[DE]	-	*
[HL]	-	*
[IX/IY]	-	*
[alternate R.]	-	o

where

-	contents do not matter
arg	arguments associated with function code
*	contents will be destroyed
o	contents will be maintained

I-call is an initialization requirement for MBIOS.
It is required to set the interrupt mode to mode-1 prior to making the I-call.
Calling address is at 0090h.
Also interupts should be disabled before I-call.

R-call (Real time)

Calling sequence:

```
CALL  0093h
```

Registers:

	in	out
[A]	func#	status
‹C›	-	error
[BC]	arg	*
[DE]	arg	*
[HL]	arg	*
(IX/IY]	-	o
[alternate R.]	-	o

R-call is a real time processing call.
To call it, load the A-register with the desired function code.
Calling entry address is 0093h.
The functions of R-calls involve generation of events and clocks. Due to the real time nature of the processes, they are done very quickly under the IRQ-disabled conditions.
R-call can be issued during K and P calls.
Run time error, if detected, will be indicated by the ‹C› flag.

K-Call (Music keyboard)

Calling sequence:

```
DI (or EI)
CALL        0096h
```

Registers:

	in	out
[A]	func#	0
‹C›	-	error
[BC]	arg	*
[DE]	arg	*
[HL]	arg	*
[IX/IY]	-	o
[alternate R.]	-	o

K-call is used for the initialization of MK and CHORD-MK, and for scanning of MK and CHORD-MK.
The entry address is 0096h.
For function calls with 01h(scan MK) and 02h(report MK), the interupt can be either enabled or disabled. However for the function call 00h(init MK), disable the interrupt before K-call.
‹C› indicates error when set upon completion of the call.
The busy condition occurs when a K-call is issued before the previous K-call has been completed. The second K-call is ignored.
This will be indicated by ‹C›.

P-Call (Play)

Calling sequence:

```
DI/EI
CALL        0099h
```

Registers:

	in	out
[A]	-	0
‹C›	-	error
[BC]	-	*
[DE]	queue map	*
[HL]	-	*
[IX/IY]	-	o
[alternate R.]	-	o

P-call retrieves events from the queue and plays them using the corresponding IDB. Calling address is at 0099h.
It can be issued whether the interrupts are enabled or disabled.
‹C› indicates an error when set upon completion of the call.
The busy condition occurs when P-call is issued before the previous P-call has been completed. The second P-call is ignored.
This will be indicated by ‹C›.

S-Call (Set up)

Calling sequence:

Call 009Ch

Registers:

	in	out
[A]	func#	error#
‹C›	-	error
[BC]	arg	*
[DE]	arg	*
[HL]	arg	*
[IX/IY]	-	o
[alternate R.]	-	o

S-call is a request that does not require real time processing.
Calling address is 009Ch.
The interrupt should be disabled prior to issuing an S-call.
S-call will be ignored if it is issued when K-call or P-call is busy. If that happens, it will be indicated by ‹C›.
‹C› indicates an error when set upon completion of the call.
The busy condition occurs when S-call is issued before the previous S-call has been completed. The second S-call is ignored.
This will be indicated by ‹C› and the A-register will be 00h.
For S21, S22, S23, S24, and S28-calls, however, the A-register will return certain error conditions. When ‹C› is set, and 00h of A-register is not encountered, it means that an error has been detected. For details, see Chapter IV.

M-call (Receive MIDI)

Calling sequence:

```
DI
CALL    00A5h
```

Registers:

	in	out
[A]	-	*
<C>	-	*
[BC]	-	o
[DE]	-	data/status
[HL]	-	o
[IX/IY]	-	o
[alternate R.]	-	o

M-call scans the MIDI input port.
If data is present at the port, it fetches the data in the D registe (interface status in the E register).
Entry address is 00A5h.
Prior to issuing the M-call, disable the interrupt.
The process will be carried out quickly, and it can be issued even while K-call or P-call is busy.

3-4 IRQ Processing

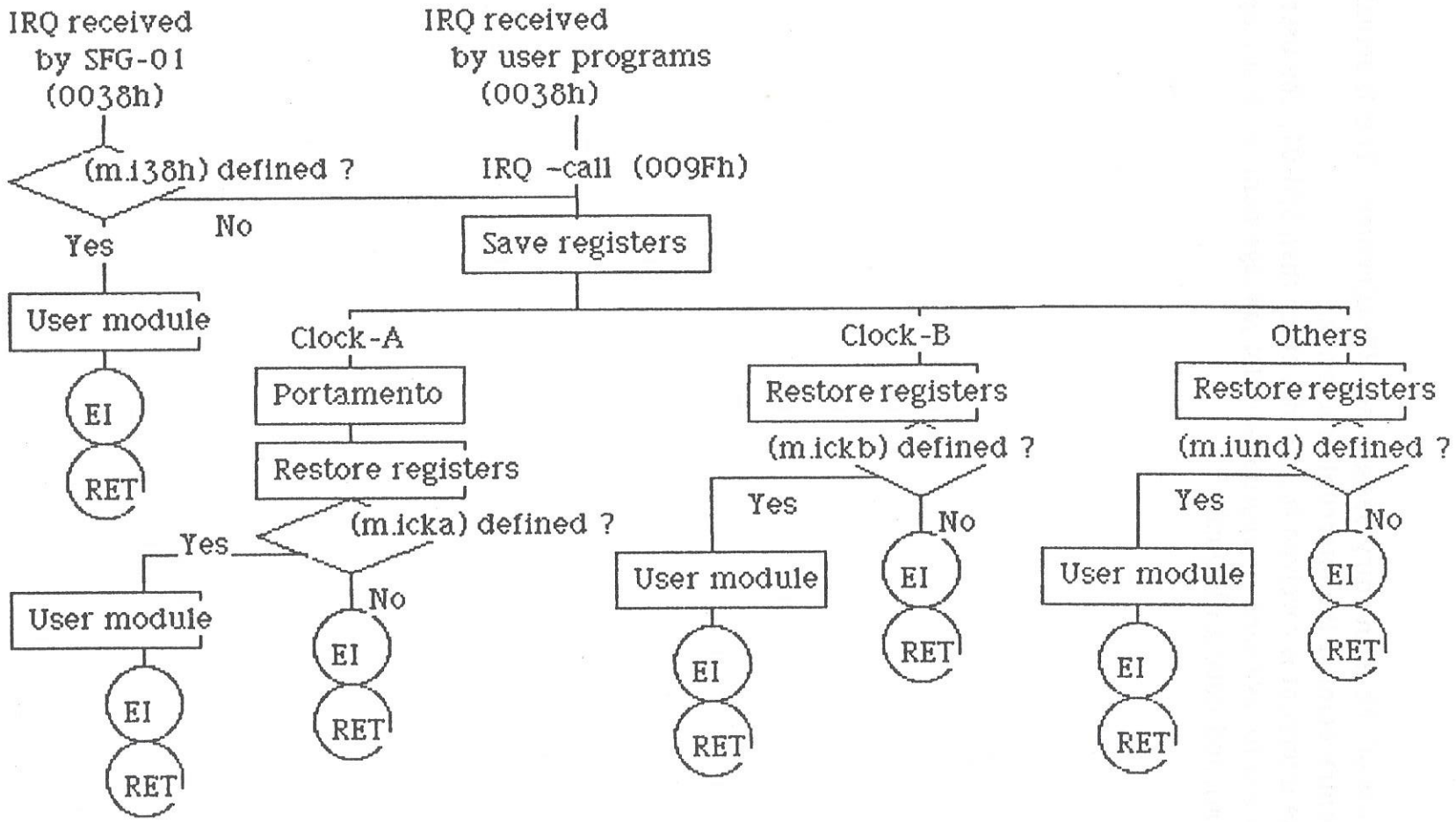
MBIOS operates on IRQ mode 1 of the Z80 CPU. IRQ mode 1 should be set before an I-call is issued.
For MBIOS, there are two sources of interrupts that are generated by the hardware of SFG-01; i.e., Clock-A and clock-B.
When I-call is first issued to initialize MBIOS, clock-A and clock-B are automatically interrupt enabled.
For applications that require disabling the clocks, refer to section 5-6.

Fig. 3.2 dipicts how the interrupts are handled with regard to MBIOS processing.
Depending on which slot , MBIOS slot or the user program slot (such as BASIC interpreter), is mapped in front, two situations can be considered as an interrupt entry; via SFG-01 or via a user slot.
When the MBIOS slot (SFG-01) receives the interrupt directly, it jumps to the right hand portion of the flow in figure 3.2, (IRQ-call, which is in most cases a normal control flow).
At this point though, there is an option provided to route the control entirely to the users own process. This can be accomplished by defining a UISV (User Interrupt Service Vector) at location M.I38H of MIDB.

Throughout the entire control flow of interrupt processing, MBIOS provides the branching capability to the users module at several key points. This again can be acomplished by pre-defining the user hook vectors appropriately in MIDB.

Now , shown after IRQ-call entry of Fig. 2.3, are further breakdowns of interrupt sources.
Once routed into this portion, the interrupt sources are polled with the scanning priority in the order of clock-A, clock-B, and other interrupt sources (from VDP and others).
At this time, the sources of clock-A , and clock-B interrupts are reset, but other sources of interrrupts are left un-reset.
Note that in clock-A handling, portamento processing is first carried out before user hooking is made.
Also the registers are restored before the branch to the user is made, so that the user can just issue ‹EI›‹RET› when he exits from the interrupt routine.

Address 009Fh is an entry designated IRQ-call entry. This is actually the true entry point of the interrupt processing.
If the interrupt is received from a slot other than SFG-01, the user is supposed to take care of appropriate slot management to bring up MBIOS in front, and issue an IRQ-call.



IRQ Flow (Fig. 3.2)

3-5 AST (Asynchronous System Trap)

AST is a means to transfer the program control from MBIOS to the user program asynchronously.
This is used when MBIOS wants to let the user know the timing of an error, or the occurrence of MK or CHORD-MK triggers. The timing of these events is, by nature, unknown, thus asynchronous.

When an AST is required, the user is required to define the trap vectors in MIDB prior to the start of real time handling.
If vectors are not defined, the trapping will not occur.
There are two trap vectors in MIDB; ‹M.TRMK› for MK, CHORD-MK and ‹M.TRER› for error.

Since AST may be generated right in the middle of an SV-call, issuing another SV-call or enabling the interrupt is not allowed in the AST handler.
When returning from the AST routine, issue a ‹RET›.
Restoring the registers is not necessary.

Register contents when AST is invoked:

[A]	trap code
‹C›	-
[BC]	-
[DE]	arg
[HL]	-
[IX/IY]	-
[alternate R.]	-

3-6 Direct Access to MIDB and IDB

The MIDB (Master Instrument Definition Block) occupies a 256 byte fixed area starting at EC00h.
The map of MIDB is shown in section 4-8.

Part of the MIDB is used as a work area by MBIOS.
MBIOS maintains various system status bytes and the user can refer to them to know what is going on (reporting bytes).
Some bytes are related to the parameters of synthesizer performance, such as transposition, clock interval variables, and so on.
It is possible to directly change these to affect the synthesizer perfomance.
There are also vectors for USIV (User service Interrupt Vector), and AST's.
When using interrupt and trap functions of MBIOS, it is necessary that these vectors be loaded appropiately by the user.
Finally, some SV-calls require related parameters to be set in MIDB before the call is made.
These calls are related to those for LFO handling, noise, and file name specification for the CMT handler.

The IDB also can be directly accessed to dynamically modify the instrument parameters.
While the MIDB parameters affect entire system performance, the parameters in the IDB affect the performance of individual instruments.
Since the IDB is defined by the user program, its address should be known to the user. In section 4-9, the IDB map is depicted.

CHAPTER IV MBIOS Syntax

4 - 1 I - call

I Initialize MBIOS

Registers:

	In	Out
[A]	-	*
<C>	-	-
[BC]	-	*
[DE]	-	*
[HL]	-	*

Initialization includes the following:

1. Enable clock-A, and clock-B.
2. Initialize the MIDB.
3. Clear AST and USIV tables.
4. Clear IDB, UVL, and EVB buffers.
5. Set RHB#0 through RHB#5 with default patterns.
6. Load Bass-line-note-offset tables with default values.

Summary of default settings in the MIDB and system status during an I-call:

MIDB:

Clock-A	m.clka:	8000h (enable interrupt)
clock-B	n.clkb:	8000h (enable interrupt)
Master transposition	m.trns:	0000h
LFO speed	m.lfo:	00h
LFO waveform	m.ctrl:	0h (saw-tooth)
AMD	m.add:	00h
PMD	m.pmd:	00h
Noise	m.nois:	00h (disabled)
UISV table	m:i38h:	all 00h
AST table	m.trmk:	all 00h
RHB (for chord and bass)	t.rhy1:	RHB#0-RHB#5 filled with default values. Rest cleared.
RHB (percussion)	t.rhy2:	RHB#0-RHB#5 filled with default values. Rest cleared.
Bass-line-note-offset table for major chord	t.rhy3:	0, +200, +400, +500, +700, +900, +1000, +1200 cents
Bass-line-note-offset table for minor chord	t.rhy4:	0, +200, +300, +500, +700, +800, +1000, +1200 cents

System status:

Auto-rhythm mode	mode=00h (refer to R-13 call)
Default RHB pointed	RHB#0
IDB's	all cleared
EVB and UVL	all cleared
Brilliance	ffh
Pitchbend	00h

4-2 R - call

R - 00 System All-note-off

This issues All-note-off events into all 11 Queues, and issues a Note-off event to the Chord-KB.

Registers:

[A]	00h	00h
<C>	-	-
[BC]	-	*
[DE]	-	*
[HL]	-	*

R - 01 All-note-off

Issues All-note-off event to designated Queue

Registers

[A]	01h	00h
<C>		<0>
[BC]	QU*/-	*
[DE]	-	*
[HL]	-	*

QU*[B] = 00h - 0Ah

R - 02 Set Event into Queue

Sets event into designated Queue

Registers:

[A]	02h	00h
<C>	-	error
[BC]	QU#/-	*
[DE]	Event	*
[HL]	-	*

QU#[B] = 00h - 0Ah

1) An error will be set when the QUEUE is already full, and the corresponding event will not be registered into the Queue.
2) The event format follows that described in Section 2-1-1.

R - 04 Set Event into Chord- KB

Issues event to the Chord-KB.

Registers:

[A]	04h	00h
<C>	-	error
[BC]	-/event	*
[DE]	-	*
[HL	-	*

---

bit7 0

event [C]

*	0	chord#

bit0-5 Chord#
bit6 always 0
bit7 <0> Chord off
<1> Chord on

1) Normally, the event will be registered when CHORD-KB and KB#C are linked.
An error will be set when the Queue is already full.

2) Except for the condition that Chord-KB is chord-off, This updates the Chord# for auto rhythm.

R - 05 Set Chord# into CHORD- KB

Issues the Chord# to the Chord-KB

Registers:

[A]	05h	00h
<C>	-	error
[B]	-/event	*
[DE]	-	*
[HL]	-	*

---

bit7 0

event[C] | * | 0 | chord# |

bit0-5	chord#
bit6	always 0
bit7	<0> chord off
	<1> chord on

1) While the Chord-KB is in the Chord-off, this function will change the Chord# of the Chord-KB whithout issuing a Chord-on command to the Chord-KB.
2) An error will be set if Queue is full.

R - 08 Start Recording

Start recording from the designated general instrument Queue to the EVB.

Regisiters:

[A]	08h	00h
<C>	-	<0>
[BC]	qu#/-	*
[DE]	-	*
[HL]	-	*

---

qu#[B]

1	0	0	0	0	Qu#

bit0-2	QU#
bit3-6	always 0
bit7	always 1

1) This will be ignored when the EVB is undefined, or during recording/playback.

R - 09 Set Recording Clock

This provides the timing clock for recording.

Registers:

[A]	09h	00h
<C>	-	<0>
[BC]	-	*
[DE]	-	*
[HL]	-	*

1) This will be ignored in any mode but recording.

2) To formulate the clock pulse train, this is normally issued successively (with interrupt clock).

R - OA Stop Recording

This stops the recording from Queue

Registers:

[A]	OAh	00h
‹C›	-	‹0›
[BC]	-	*
[DE]	-	*
[HL]	-	*

1) This registers an All-note-off event for a Queue that was being recorded.
2) This will be ignored in all modes, except for recording.

R - 0B Start Playback

This carries out playback from the EVB for the designated Queue.

Registers:

[A]	0Bh	00h
<C>	-	<0>
[BC]	qu*/-	*
[DE]	-	*
[HL]	-	*

---

qu* [B]

bit7					0
1	0	0	0	0	QU*

bit0-2	QU*
bit3-6	always 0
bit7	always 1

1) This will be ignored when the EVB is undefined, or during recording/playback.

R - 0C Set Playback Clock

This provides the timing clock for playback.

Registers:

[A]	0Ch	00h
<C>	-	<0>
[BC]	-	*
[DE]	-	*
[HL]	-	*

1) This will be ignored during every mode except for playback.

2) To obtain a clock pulse train, this call is normally issued successively in the clock interrupt routine.

R - 0D Stop Playback

This stops the playback.

Registers:

[A]	0Dh	00h
<C>	-	<0>
[BC]	-	*
[DE]	-	*
[HL]	-	*

1) This sends an All-note-off event for the Queue that was being played back.
2) This will be ignored during all modes, except for playback.

R - 10 Start Auto Rhythm

This starts the automatic rhythm generator.

Registers:

[A]	10h	00h
<C>	-	<0>
[BC]	-/Mode	*
[DE]	Division/-	*
[HL]	-	*

Mode [C]

0	0	0	0	0	0	0	*

bit0

<0> Starts immediately.

<1> Starts in synchronization with the Chord on trigger of Chord-MK (Sync to MK if Chord-MK is not set)

Division [D]

Inserts a divisor number indicating how many R-11 calls a 1/48th note are equivalent to.
<00h> is equivalent to 256.

1) If a rhythm pattern is already being used, issuing this call will start again from the beginning of the RHB.

R - 11 Set Auto-Rhythm Clock

This provides the clock for the automatic rhythm generator.

Registers:

[A]	11h	Clock*
<C>	-	<C>
[BC]	-	*
[DE]	-	*
[HL]	-	*

Clock*
This indicates the present reference position of the RHB (in 96 bytes).
Divided down by 3 when synchronization starts.
During 8/4 time, the CLOCK* is as shown below.

FFh
FFh Chord-MK / Chord-on
00h---Execute event 0
01h
01h
01h---Execute event 1
02h

5Eh---Execute event 94
5Fh
5Fh
5Fh---Execute event 95
00h
00h
00h---Excute event 0
01h

1) This will be ignored during all modes except while playing the auto rhythm pattern.

R - 12 Stop Auto-Rhythm

Stops automatic rhythm generator.

Registers:

[A]	12h	00h
<C>	-	<0>
[BC]	-	*
[DE]	-	*
[HL]	-	*

1) Regardless of whether the automatic rhythm generator is on or not, this registers All-note-off events to all Queues that have been used by the automatic rhythm generator.

R - 13 Select Auto-Rhythm Queue

Selects the queue for auto rhythm

Registers:

[A]	13h	00h
<C>	-	<0>
[BC]	-/Mode	*
[DE]	-	*
[HL]	-	*

Mode[C]

0	0	0	0	*	*	*	*

bit0
<0> Chord-KB and QU*C are linked.
<1> QU*C and automatic rhythm generator are linked.
bit1 QU*B automatic rhythm generator
bit2 QU*P automatic rhythm generator
bit3 QU*C automatic rhythm generator

1) An All-note-off event will be sent to QU*C when bit 0 and bit 3 have been set.
Or All-note-off event will be sent to QU*B when bit 1 has been set.
Or All-note-off event will be sent to QU*P when bit 2 has been set.

R - 14 Select RHB

Selects RHB for automatic rhythm generator.

Registers:

[A]	14h	00h
<C>	-	<0>
[BC]	-/RHB#	*
[DE]	-	*
[HL]	-	*

RHB#[C] = 0 to 15

R-18 Load LFO

Load LFO parameters into the FM sound generator IC.

Registers:

[A]	18h	00h
<C>	-	<0>
[BC]	-	*
[DE]	-	*
[HL]	-	*

Preset the following LFO parameters into the MIDB prior to issuing this command.

MIDB entries:

M.LFO	Speed
M.AMD	amd
M.PMD	pmd
M.CTRL	wave form
M.NOIS	noise

R-19 Load KC

Loads the KC into the FM sound generator IC.

	in	out
[A]	19h	00h
<C>	-	<0>
[BC]	-	*
[DE]	-	*
[HL]	-	*

1) It is used to load KC dynamically during transposition or portamento

2) Issue this command in synchronization with CLOCK-A , so that update timing of the pitch to accomplish poratamento is synchronized to clock-A.

3) This command does not have to be issued when master transpose, poratamento, or pitch bend are not being used.

R-21 Send data through MIDI

Outputs a given single byte of data to the MIDI output port.

Registers:

	in	out
[A]	21h	00h
<C>	-	error
[BC]	-/data	*
[DE]	-	*
[HL]	-	*

---

1) An error will be set if the command is issued while TxRDY is not ready.

4-3 K-Call

K-00 **Init MK**

Initializes MK, sets sync-hold for CHORD-MK , sets velocity for MK, and establishes the link up between the MK and the specified queue.

Registers:

	in	out
[A]	00h	00h
<C>	-	busy
[BC]	link/mode	*
[DE]	velocity/-	*
[HL]	-	*

---

link [B] | * | 0 0 0 0 | * * * |

bit0-2	QU# (0-7)
bit7	<0> No link with queue
	<1> Link with queue

mode [C] | 0 0 0 0 0 0 0 | * |

lsb	<0> Use MK only
	<1> Use Chord-MK and MK

Velocity [D] | * * * * * * * * |

bit 0-bit7	00h min.
	ffh max.

K-01 Scan MK

Scans the MK. Event detected will be written into (linked) queue.

Registers:

	in	out
[A]	01h	00h
<C>	-	busy
[BC]	-	*
[DE]	-	*
[HL]	-	*

---

1) The output of this command is to write the detected event into the queue.

2) Normally, to scan the MK, this call needs to be issued successively.

3) If AST vectors for MK or Chord-MK has been specified in the MIDB, it will cause AST trapping via AST vectors (see 4-7).

K-02 Report MK status

Scans the MK, and returns the on/off status of the MK.

Registers:

	in	out
[A]	02h	00h
<0>	-	busy
[BC]	-	*
[DE]	buffer address	*
[HL]	-	*

1) The buffer is comprised of 9 bytes as shown below.

	msb							lsb	
0:	0	C	B	A#	0	A	G#	G	higher KC#
	.....								
7:	0	F#	F	E	0	D#	D	C#	
8:	0	C	0	0	0	0	0	0	lower KC#

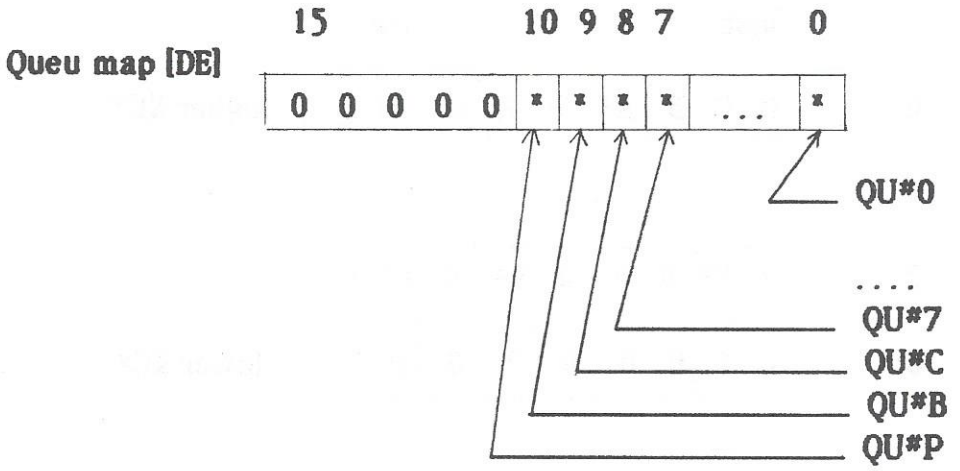
4 - 4 P - call

P	Play

This retrieves events from the designated name, and processes it by using the IDB.

Registers:

	in	out
[A]	-	00h
<C>	-	busy
[BC]	-	*
[DE]	Queue map	*
[HL]	-	*



1) The retrieval of events from the designated Queue is carried out repeatedly until the Queue becomes empty.

4-5 S-call

S - 00 Define IDB

Either defines or cancels the IDB.

Registfers:

	in	out
[A]	00h	00h
<C>	-	busy
[BC]	IDB#/-	*
[DE]	IDB address	*
[HL]	-	*

---

1) Cancels the IDB when the IDB address (contents of DE) = 0000h.

2) Cancelling the IDB, while it is still engaged in key-on, should be avoided.

Initial setting of the IDB parameters:

KC range	00h to 7eh
Pitchbend depth	00h
transposition by instrument	00h
portamento speed	00h
RR(default sustain value)	03h
volume	coh
Voice data	cleared

Initial setting of the IDB mode (held in the system):

sustain-off
multi-triggered
fingered portamento (with 0 speed)
pitchbend enabled (with 0 depth)

* The above setting is equivalent to issuing, mode=0, via an S-12 call.

S - 02 Define EVB

Either defines or cancels the event buffer (EVB).

Registers:

	in	out
[A]	02h	ooh
<C>	-	busy
[BC]	size	*
[DE]	address	*
[HL]	-	*

1) The EVB is calceled when the address in [DE] is 0000h.

2) When defined, the contents of the EVB will not be cleared.

S - 03 Define UVL

Either defines or cancels the user voice library (UVL)

Registers:

	in	out
[A]	03h	ooh
<C>	-	busy
[BC]	-	*
[DE]	address	*
[HL]	-	*

1) The UVL is cancelled when the address in [DE] is 0000h.

2) When defined, the contents of UVL will not be cleared.

S - 04 Initialize EVB

This initializes the event buffer (EVB).

Registers:

	in	out
[A]	04h	ooh
<C>	-	busy
[BC]	-	*
[DE]	-	*
[HL]	-	*

S - 09 Assign channel

This allocates the channels of the FM sound generator IC for the requesting IDB.

Registers:

	in	out
[A]	09h	ooh
<C>	-	busy
[BC]	ch*0 to 3	*
[DE]	ch*4 to 7	*
[HL]	-	*

---

	15 8	7 0		
[BC]	ch*0	ch*1	ch*2	ch*3

IDB* corresponding to ch*3

[DE]	ch*4	ch*5	ch*6	ch*7

1) The function assigns IDB* between 0 and Ah using 4-bit slot for each channel number.
2) IDB*C , IDB*B, IDB*P0, and IDB*P1 use the following fixed channels:
   ch*3,4 for IDB*C, ch*5 for IDB*B, ch*6 for IDB*P0, and ch*7 for IDB*P1. For IDB*P0 and IDB*P1, use then at the same time.
3) For a single channelled IDB, assign a channel to the IDB whose number is equivalent to that of channel. (Foe ex., ch*2 for IDB*2).
4) Assigning a channel will not alter the previous settings of the LFO.

S - 0A Assign IDB to Queue and/or MIDI channel

This assigns the corresponding input Queue and MIDI output channel to the designated general IDB.

Registers:

		in	out
[A]		0Ah	ooh
<C>		-	busy
[BC]		IDB#/-	*
[DE]	Queuelink/MIDIlink		*
[HL]		-	*

---

valid IDB# 00h - 07h

Queuelink [D]

1	0	0	0	0	Qu#

MIDIlink [E]

*	0	0	0	MIDI#

bit7 <1> MIDI is assgined

1) When this call is issued, an All-Note-Off event will be executed for the corresponding IDB.

2) When linking with MIDI, be sure to assign at least one channel of the FM sound generator IC to the IDB.

S - 0B Issue All-Note-Off to IDB

This issues and executes an All-note-off to designated IDB.

Registers:

	in	out
[A]	0Bh	ooh
<C>	-	busy
[BC]	IDB#/-	*
[DE]	-	*
[HL]	-	*

Valid UDB# 00h to 0Bh

S - 0C Initialize MIDI

This initializes the MIDI port.

Registers:

	in	out
[A]	0Ch	ooh
<C>	-	busy
[BC]	-	*
[DE]	-	*
[HL]	-	*

---

1) When called, this routine disables both the RxRDY and TxRDY interrupts of MIDI.
In other words, MBIOS uses MIDI under a non-interrupted condition.

S - 10 Set Brilliance

This sets the system parameter , Brilliance.

Registers:

	in	out
[A]	10h	ooh
<C>	-	busy
[BC]	-/Brilliance	*
[DE]	-	*
[HL]	-	*

Brilliance [C] 00h (dark) - ffh (bright)

1) Since the Brilliance is a system parameter, this affects the whole system, not merely a single instrument.

S-11 Set pitchbend

This sets the system parameter, pitchbend.

Registers:

	in	out
[A]	11h	ooh
<C>	-	busy
[BC]	-/Pitchbend	*
[DE]	-	*
[HL]	-	*

Pitchbend [C]	2's complement representation	
	80h	-100%
	00h	0%
	7fh	+100%

1) As in portamento, to realize pitchbend, a R-19 call (updating KC) should be repeatedly issued in syncronization to the A-clock interrupts.

S - 12 Define Play-mode

This sets the performance mode of the designated IDB (by the FM sound generator IC).

Registers:

	in	out
[A]	12h	ooh
<C>	-	busy
[BC]	IDB#/Mode	*
[DE]	-	*
[HL]	-	*

IDB# [B] 00h to 07h

Mode [C]

0	0	0	0	*	*	*	*

lsb <1> Sustain-On
bit1 <1> Single triggered
<0> Multi triggered
bit2 <1> Full portamento
<0> Fingered portamento
bit3 <1> Disable pitchbend
<0> Enable pitchbend

S-13 Set Volume

This sets the Volume of the designated IDB.

Registers:

	in	out
[A]	13h	ooh
<C>	-	busy
[BC]	IDB#/Volume	*
[DE]	-	*
[HL]	-	*

IDB# [B]	00h to 0Bh
Volume [C]	00h (min) to ffh (max)

S-14 Load Voice

This loads the voicing parameter information of designated IDB into the FM sound generator IC.

Registers:

	in	out
[A]	14h	ooh
<C>	-	busy
[BC]	IDB#/-	*
[DE]	-	*
[HL]	-	*

IDB# [B] 00h to 0Ch

S - 15 Get Voice

This transfers voicing parameter information from the Voice library to the voicing parameter area of the designated IDB.

Registers:

	in	out
[A]	15h	ooh
<C>	-	busy
[BC]	IDB#/Voice#	*
[DB]	-	*
[HL]	-	*

IDB# [B]	00h to 0Ch
Voice# [C]	00h to 2Fh for SVL 40h to 6Fh for UVL

1) The transfer will be ignored if the UVL has not been defined.

S - 16 Put Voice

This transfers the voice parameter information from the IDB to the UVL.

Registers:

	in	out
[A]	16h	ooh
<C>	-	busy
[BC]	IDB#/Voice#	*
[DE]	-	*
[HL]	-	*

IDB# [B]	00h to 0Ch
Voice#[C]	00h to 2Fh for SVL 40h to 6Fh for UVL

1) The transfer will be ignored if the UVL has not been defined.

S - 21 Read UVL

This reads in the UVL from the CMT.

Registers:

	in	out
[A]	21h	error#
<C>	-	error
[BC]	-	*
[DE]	-	*
[HL]	-	*

Error# [A]	00h	Normal end
	FFh	Size Error (not enough buffer)
	Other non-0#	MSX-BASIC error

1) The file name on the tape is assumed to be "VOICE". Search on the tape will be made until "VOICE" is found.

2) When used, the hook information at H.KEYI and H.ERRO of BASIC working area will be destroyed.

3) If UVL has not been defined, error will be flagged out as "size" error.

S - 22 Write UVL

This writes the UVL into the CMT.

Registers:

	in	out
[A]	22h	error*
<C>	-	error
[BC]	-	*
[DE]	-	*
[HL]	-	*

Error* [A]	00h	Normal end
	non-0	MSX-BASIC Error

1) The file name of the data being written is always assumed to be "VOICE".

2) When used, the hook information at H.KEYI and H.ERRO of the BASIC working area will be destroyed.

3) Prior to issuing the call, the UVL must have been defined. Omission of the UVL could cause the system to crash.

S-23 Read EVB

This reads in the EVB from the CMT.

Registers:

	in	out
[A]	23h	Error*
<C>	-	error
[BC]	-	*
[DE]	-	*
[HL]	-	*

Error* [A]	00h	Normal End
	FFh	Size Error
	Other non-0*	MSX-BASIC Error

1) Prior to this call, a file name must be placed at M.EFVB of MIDB.
The searching of the filename on the tape will be continuously carried out until the file name is found.

2) When used, the hook information at H.KEYI and H.ERRO of the BASIC working area will be destroyed.

3) If the EVB has not been defined prior to this call, the error will be flagged out via size error.

S - 24 Write EVB

This writes the EVB into the CMT.

Registers:

	in	out
[A]	24h	Error#
<C>	-	error
[BC]	-	*
[DE]	-	*
[HL]	-	*

Error#	00h	Normal End
	Non-0#	MSX-BASIC Error

1) Prior to this call, a file name must be placed at M.EFVB of MIDB.

2) When used, the hook information at H.KEYI and H.ERRO of the BASIC working area will be destroyed.

3) Prior to issuing the call, UVL must have been defined. Omission of the UVL could cause the system to crash.

S - 28 CSM Voicing

This will call the CSM vocal synthesis driver.

Registers:

	in	out
[A]	28h	Error#
<C>	-	error
[BC]	-	*
[DE]	CSM buffer address	*
[HL]	-	*

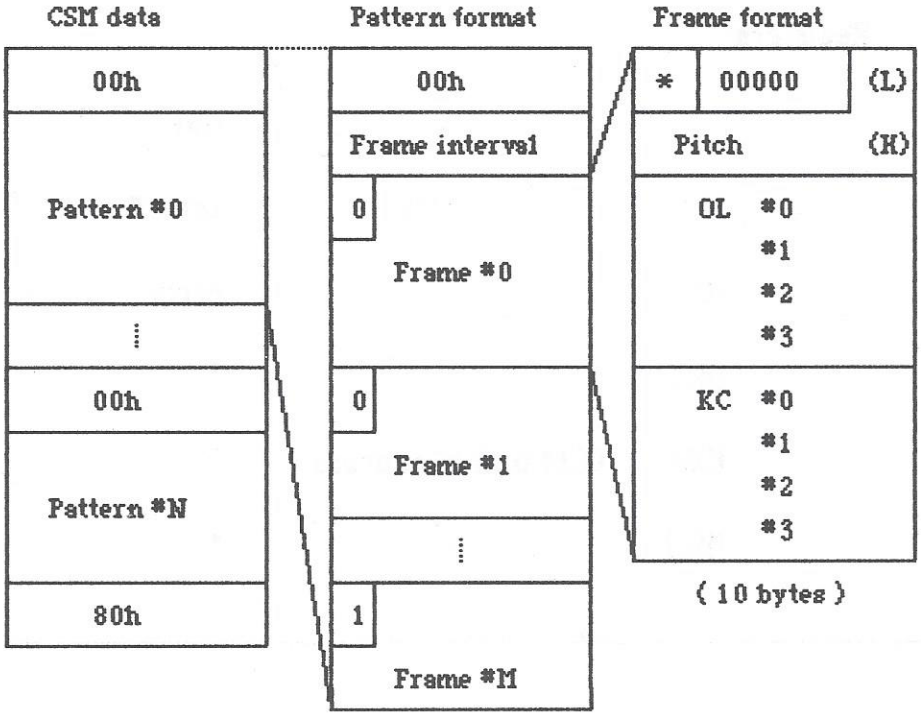
Error#	00h	Normal End
	01h	Run time error

1) Prior to calling this command, assign all the channels to the IDB#CSM, with voice#46 linked up to it.

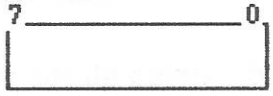
2) In IRQ processing, the control to UISV (user interrupt service vector) will not be granted at all during the CSM processing.

CSM data format

CSM data is comprised of multiple patterns, which in turn are comprised of multiple frames.



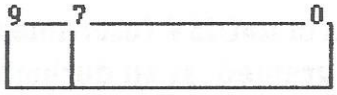
Frame interval (clock-B)



00h - FFh

$\{(256 - value) \cdot 0.286 \cdot 10^{-3}\}^{-1}$ Hz

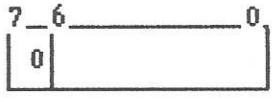
Pitch (clock-A)



000h - 3FFh

$\{(1024 - value) \cdot 17.88 \cdot 10^{-6}\}^{-1}$ Hz

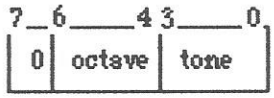
OL (amplitude)



00h - 7Fh

$(-0.75 \cdot value)$ db

KC* (Frequency)



4-6 M - call

M **Receive MIDI**

This scans the input port of the MIDI interface, and return the data if there is a data.

Registers:

	in	out
[A]	-	*
<C>	-	*
[BC]	-	o
[DE]	-	data/staus
[HL]	-	o

Status [C]

0	0	*	*	0	0	*	0

bit1	RxRDY
bit4	Overrun Error
bit5	Framing Error

4-7 AST (Asynchronous System Trap)

AST - 01 MK trigger

This is an AST caused by the trigger from the MK (Note-on, Note-off)

Registers:

	out
[A]	01h
<C>	-
[BC]	-
[DE]	Event
[HL]	-

1). Event [DE] contains the same event data used in Queue.

2). To receive this AST control, MK scan (K-01) should be issued elsewhere in the program.
In other words, this AST is invoked during the execution of K-01 routine, if the vector is defined in the MIDB.
Users however don't have to worry about the synchronicity of both (K-01 call and AST-01 handler) routines.

AST - 02 Error

This is an AST caused by the error generated by MBIOS.

Registers:

	out
[A]	02h
<C>	-
[BC]	-
[DE]	error#/-
[HL]	-

Error#[D]	0 0 0 1 Qu#	The Qu# has overflowed.
	0 0 1 0 0 0 0 0	EVB is full (end of recording) (All-note-off issued)
	0 0 1 0 0 0 0 1	EVB end is encountered (End of playback)
	0 0 1 1 MIDI#	MIDI time-out Error

1) 1xh error will occur during MK processing or automatic rhythm generation. If detected, issue an All-note-off to correspoding Queue.

2) When playback/recording is stopped by usual SV-call, AST (20h and 21h) will not occur.

3) 3xh will occur during play processing while MIDI out is specified as well. If this occurs, it implies a possible hardware error.

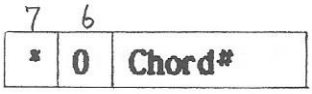
AST - 03 Chord-MK Trigger

This is an AST caused by the Chord-MK trigger (Chord-on, Chord-off)

Registers:

	out
[A]	03h
<C>	-
[BC]	-/Event
[DE]	-
[HL]	-

Event [C]



bit0-bit5		Chord#
bit7	<1>	Chord-on
	<0>	Chord-off

1). In order to utilize this trap, the same attention used in AST-01 should be paid.
That is, issue K-01 call elsewhere in the program.

4-8 MIDB

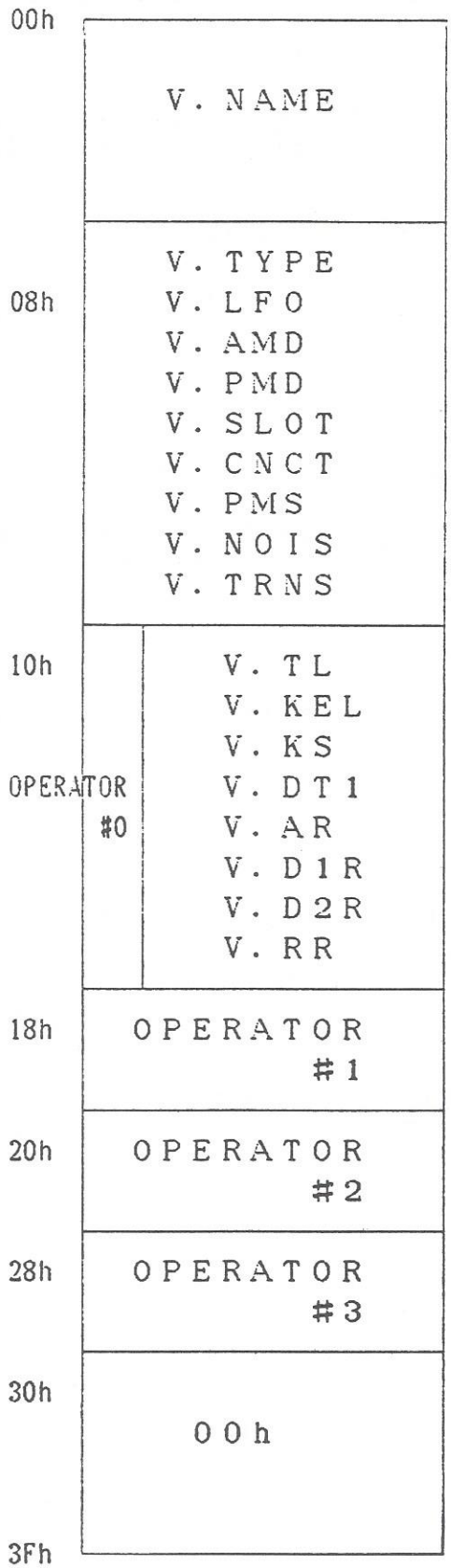
name	address	purpose to use	comments
m.clka	$EC00	Modify directly	clock-a interval
		[+0] **00 0000	clock-a (low)
		[+1] **** ****	clock-a (high)
		0000h ---> 18.2 ms (max)	
		8000h ---> 9.1 ms	
		FFC0h ---> 0.071 ms (min)	
			linear rate
m.clkb	$EC02	Modify directly	clock-b interval
		[+0] ---- ----	clock-b (low)
		[+1] **** ****	clock-b (high)
		0000h ---> 72.8 ms (max)	
		8000h ---> 36.4 ms]	
		FF00h ---> 0.285 ms (min)	
			linear rate
m.trns	$EC04	Modify directly	Transpose(Master)
		[+0] **** ****	fraction
		[+1] **** ****	KC
			2's complement
m.lfo	$EC10	SV-call	
		**** ****	LFO frequency
m.amd	$EC11	SV-call	
		0*** ****	amd
m.pmd	$EC12	SV-call	
		0*** ****	pmd
m.ctrl	$EC13	SV-call	
		0000 00**	LFO wave form
m.nois	$EC14	SV-call	
		*00- ----	<1>noise Enable
		-00* ****	noise frequency

f.evb	$EC1B	Status	recording, playback status
		0000 00*-	<0>playback <1>recording
		0000 00-*	<1>on execution
m.chrd	$EC23	Status	CHORD-KB current status
		*0-- ----	<0>off state <1>on state
		-0** ****	chord#
m.i38h	$EC30	Vector	0038h UISV (User Interrupt Service Vector)
m.icka	$EC32	Vector	irq-a UISV
m.ickb	$EC34	Vector	irq-b UISV
m.iund	$EC38	Vector	irq-undefined UISV
m.trmk	$EC3C	Vector	MK,CHORD-MK trigger trap address
m.trer	$EC3E	Vector	error trap address
m.fevb	$EC44	SV-call	[evb] file name <6 B>
t.rhy0	$EC88	Table	RHB time sigunature <16 B>
t.rhy1	$EC98	Pointer	RHB(chord,bass) address <32 B>
t.rhy2	$ECB8	Pointer	RHB(purcussion) address <32 B>
t.rhy3	$ECD8	Table	bass-line-pitch offset table (major) <8 B>
t.rhy4	$ECE0	Table	(minor) <8 B>

4-9 IDB

name	address	purpose to use	comments
i.krng	$00	Modify directly	KC range
		[+0] 0*** ****	KC# (maximum)
		[+1] 0*** ****	KC# (minimum)
i.pchb	$02	Modify directly	
		0000 ****	pitchbend depth 00h ---> 0 cent 01h ---> 100 cent ... 08h ---> 1200 cent
i.trns	$03	Modify directly	Transpose(Instrument)
		[+0] **** ****	fraction
		[+1] **** ****	KC 2's complement
i.port	$05	Modify directly	
		**** ****	portamento speed 00h ---> non-portamento 01h ---> fast ... FFh ---> slow
i.sust	$06	Modify directly	
		0000 ****	[RR] in sustain/on mode (same rate for all operators)
v.name	$10	SV-call	Voice data area

4-10 VOICE DATA



---

v.name	$00	voice name	20h (space) 30h-39h , 41h-5Ah First chr. must be 41h-5Ah
v.type	$07	user id-code	
v.lfo	$08	LFO frequency	00h ---> .0008Hz(slow) 80h ---> .2134Hz FFh ---> 52.9Hz(fast) logarithmic rate
v.amd	$09		
		*--- ----	<1>enable load LFO data
		-*** ****	amd 7Fh ---> deep
v.pmd	$0A		
		*--- ----	<1>sync LFO in key/trigger
		-*** ****	pmd 7Fh ---> deep
v.slot	$0B		
		0*-- -000	operator#3 enable
		0-*- -000	operator#2 enable
		0--* -000	operator#1 enable
		0--- *000	operator#0 enable
v.cnct	$0C		
		*--- ----	<1>stereo/L output enable
		-*-- ----	<1>stereo/R output enable
		--** *---	feedback level 7 ---> 4 pai (deepest)
		---- -***	algorithm number
v.pms	$0D		
		0*** 00--	pms 0 ---> +/- 0 cent 1 ---> +/- 5 cent 2 ---> +/- 10 cent 3 ---> +/- 20 cent 4 ---> +/- 50 cent 5 ---> +/-100 cent 6 ---> +/-400 cent 7 ---> +/-700 cent
		0--- 00**	ams 0 ---> 0db 1 ---> -24db 2 ---> -48db 3 ---> -96db

v.nois $0E

```
*--- ----      enable noise
-**- ----      LFO wave form
                    00 ---> sawtoothed
                    01 ---> rectangler
                    10 ---> triangler
                    11 ---> sample & hold
---* ****      noise frequency
```

v.trns $0F

```
**** ****      Transpose(Voice)
                    2's complement
                    -12700 cent - 12700 cent
                                    (100 cent)
```

---

v.tl $00

```
0*** ****      OL(original)
                    0db - 95.25db (0.75db)
```

v.vel $01

```
*--- 000-      keyboard level scaling type
                  <0> Low pass
                  <1> High pass
-*** 000-      velocity depth
                  carrier  0db - +/-10.5db (1.5 db)
                  index    0db - +/-5.25db (0.75db)
---- 000*      enable brilliance
```

v.ks $02

```
**** ----      keyboard level scaling depth
                    0db - -22.5db (1.5db)
---- ****      TL(adjust)
                    0db - -11.25db (0.75db)
```

v.dt1 $03

```
-*** ----      DT1
-*-- ----        sign of DT1
                    0 ---> positive(up)
                    1 ---> negative(down)
--** ----        data of DT1
                    3 ---> max
---- ****      multiple
                     0 ---> *0.5
                     1 ---> *1
                     2 ---> *2
                      ...
                    15 ---> *15
```

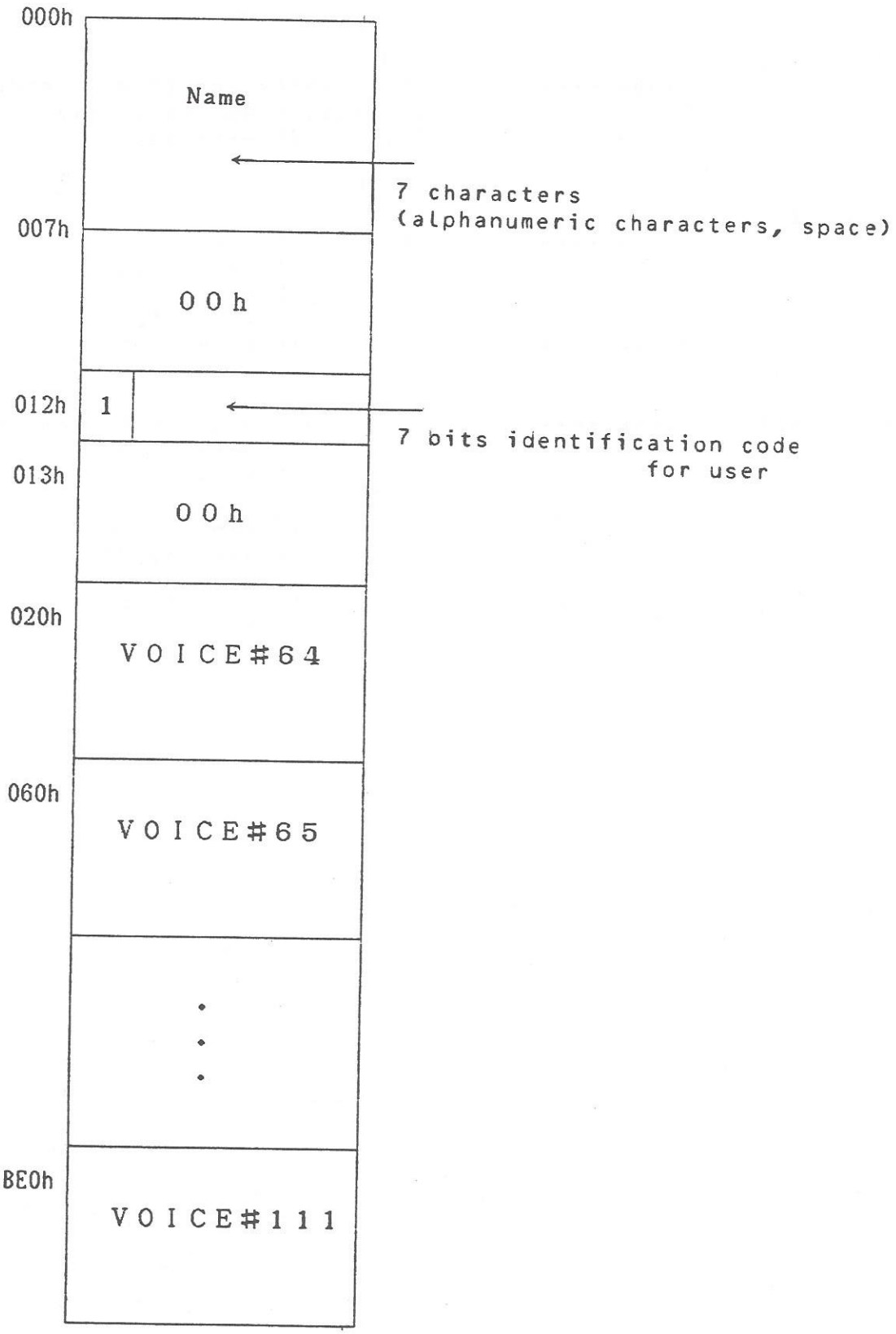
```
v.ar     $04
                **0- ----       keyboard rate scaling
                                   the higher KC#, the faster rate
                --0* ****       AR      31 ---> fast

v.d1r    $05
                *00- ----       <0> disable ams (= modulator)
                                <1> enable ams (= carrier)
                -00* ****       D1R     31 ---> fast

v.d2r    $06
                ***- ----       DT2      0 ---> 0
                                          1 ---> *1.41
                                          2 ---> *1.57
                                          3 ---> *1.73
                ---* ****       D2R     15 ---> fast

v.rr     $07    **** ----       SL   turning point to 2nd decay
                                         0 --->   0db
                                         1 --->  - 3db
                                           ...
                                        13 ---> -39db
                                        14 ---> -42db
                                        15 ---> -93db
                ---- ****       RR      15 ---> fast
```

4-11 UVL


000h
Name
7 characters
(alphanumeric characters, space)
007h
00h
012h
1
7 bits identification code
for user
013h
00h
020h
VOICE#64
060h
VOICE#65
BE0h
VOICE#111

4-12 Setting-up Information

Setting-up Information (fig. 4.1)

o effective
- not effective

item	number of channels idb#0-7 1	idb#0-7 over 1	idb#c	idb#b	idb#p0/p1
KC Range	o	o	-	-	-
Pitchbend	o	o	-	-	-
Pitchbend Enable	o	o	-	-	-
Pitchbend Depth	o	o	-	-	-
Portamento Mode	o	-	-	-	-
Portamento Speed	o	-	-	-	-
Trigger Mode	o	-	-	-	-
Sustain Mode	o	o	-	-	-
Volume	o	o	o	o	o
Brilliance	o	o	o	o	o
Brilliance Enable	o	o	o	o	o
Transpose(Master)	o	o	o	o	-
Transpose(Instrument)	o	o	o	o	-
Transpose(Voice)	o	o	o	o	o
LFO Speed	o	o	o	o	o
LFO Wave form	o	o	o	o	o
AMD	o	o	o	o	o
PMD	o	o	o	o	o
AMS	o	o	o	o	o
PMS	o	o	o	o	o
AMS Enable	o	o	o	o	o
LFO Trigger Sync	o	o	-	-	-
Noise Enable	o	o	-	-	-
Noise Frequency	o	o	-	-	-
OL(Original)	o	o	o	o	o
OL(Adjust)	o	o	o	o	o
SL	o	o	o	o	o
AR	o	o	o	o	o
D1R	o	o	o	o	o
D2R	o	o	o	o	o
RR	o	o	o	o	o
RR(sustain)	o	o	-	-	-
Operator Enable	o	o	o	o	o
Velocity Depth	o	o	-	-	-
KS(Rate)	o	o	o	o	o
KS(Level) Type	o	o	-	-	-
KS(Level) Depth	o	o	-	-	-
Multiple	o	o	o	o	o
DT1	o	o	o	o	o
DT2	o	o	o	o	o
Feedback Level	o	o	o	o	o
Algorithm	o	o	o	o	o
Stereo L/R	o	o	o	o	o

Access Method of Setting-up Information (fig. 4.2)

item	access method
KC Range	i.krng (idb)
Pitchbend	S-11
Pitchbend Enable	S-12
Pitchbend Depth	i.pchb (idb)
Portamento Mode	S-12
Portamento Speed	i.port (idb)
Trigger Mode	S-12
Sustain Mode	S-12
Volume	S-13
Brilliance	S-10
Brilliance Enable	S-14 (v.vel/voice)
Transpose(Master)	m.trns (midb)
Transpose(Instrument)	i.trns (idb)
Transpose(Voice)	S-14 (v.trns/voice)
LFO Speed	S-18 (m.lfo /midb) , S-14 (v.lfo /voice)
LFO Wave form	S-18 (m.ctrl/midb) , S-14 (v.nois/voice)
AMD	S-18 (m.amd /midb) , S-14 (v.amd /voice)
PMD	S-18 (m.pmd /midb) , S-14 (v.pmd /voice)
AMS	S-14 (v.pms /voice)
PMS	S-14 (v.pms /voice)
AMS Enable	S-14 (v.d1r /voice)
LFO Trigger Sync	S-14 (v.pmd /voice)
Noise Enable	S-18 (m.nois/midb) , S-14 (v.nois/voice)
Noise Frequency	S-18 (m.nois/midb) , S-14 (v.nois/voice)
OL(Original)	S-14 (v.tl /voice)
OL(Adjust)	S-14 (v.ks /voice)
SL	S-14 (v.rr /voice)
AR	S-14 (v.ar /voice)
D1R	S-14 (v.d1r /voice)
D2R	S-14 (v.d2r /voice)
RR	S-14 (v.rr /voice)
RR(sustain)	i.sust (idb)
Operator Enable	S-14 (v.slot/voice)
Velocity Depth	S-14 (v.vel /voice)
KS(Rate)	S-14 (v.ar /voice)
KS(Level) Type	S-14 (v.vel /voice)
KS(Level) Depth	S-14 (v.ks /voice)
Multiple	S-14 (v.dt1 /voice)
DT1	S-14 (v.dt1 /voice)
DT2	S-14 (v.d2r /voice)
Feedback Level	S-14 (v.cnct/voice)
Algorithm	S-14 (v.cnct/voice)
Stereo L/R	S-14 (v.cnct/voice)

CHAPTER V Writing programs

5-1 Program example

The following sample program will demonstrate the following instruments.

manual performance by MK with IDB*0
auto play of IDB*1
auto rhythm of IDB*P0 and IDB*P1

Program explanations will be made in the following sections.
With the example program, it is assumed that the VDP and the PPI of the MSX system have already been initialized by MSX-BASIC.

5.1.1 Program structure

Lines between 35 and 50 show the framework of the sample program.
Here, the IRQ mode is set to mode-1. Stack area is defined, and SFG-0,1 is set to slot 3.
As well as initializing the system, UISV and AST vectors are defined in the MIDB.
Note the clock-B vector, defined in line 46, routes the control to interrupt processing entry (lines 195 and 196); then real time rhythm handler "auto:" (lines 224 to 244) is invoked.
The idea of the program is to use clock-B for the auto-rhythm clock, handle the events there, and play them in the main loop of the program.

5.1.2 Definition of instruments

In the sample program, IDB*P0, IDB*P1, IDB*0 and IDB*1 are defined in the line numbers between 59 and 129.
Between lines 141 and 144, the channels of the FM sound generator IC are assigned to the above IDB's.
Assignment details are;

IDB*0	channels	0,2,3,4,5
IDB*1	channel	1
IDB*P0	channel	6
IDB*P1	channel	7

5.1.3 Performance of IDB*1

IDB*1 is supposed to play the performance data that is defined in the lines between 246 and 262.
With the auto-rhythm clock synchronized to IDB*P0/1, the event data is loaded into queue (QU*1). This is done in the lines between 225 and 244.
This routine "auto:" is of course driven by the interrupt , and the event requests in the queue are then played by the main loop between line 165 and 178. The playing is carried out by the lines from 166 and 168.

5.1.4 Manual play by MK

MK is initialized in the lines between 136 and 139. Here, CHORD-MK is not used. Velocity is set to a default value of 80h. MK is linked up with QU#0.
With this setting, the lines 169 to 170 in the main loop issue the MK scanning request. If key actuation is detected, the events are sent into QU#0, which will be played by a P-call between lines 166 and 168.

In this example, since the MK scan is placed in the main loop, no MK trapping is used. The method of detecting MK events via MK trap (by defining AST vector in MIDB) would also be possible.

5.1.5 Auto-rhythm performance

Between lines 146 and 148, the queue for auto rhythm is selected. Here only QU#P is chosen.
Then between lines 149 and 151, the RHB (rhythm buffer) to be used is selected. This determines the rhythm pattern. Here, preset pattern #0 is chosen.
Clock handling for the rhythm is done in clock-B handler "auto:" between lines 225 and 244.
Here, with lines 225 and 226, an auto-rhythm clock is issued in synchronization to the clock-B interrupt.
Actual start of the auto rhythm is specified in the lines between 152 and 155. In this case, the rhythm is will begin with the first note-on from MK.

Note that, as explained before, the IDB#1 performance is supposed to be synchronized to the auto ryhthm. Therefore in "auto:" routine, line 227 checks if the rhythm has already started or not (before it loads the event into QU#1).

```
MAIN       z80.i8080 ass.vr-1.2 9-SEP-84   19:52     PAGE  1

  1                                ;
  2                                ;
  3                                ;////////////////////////////////////////////////////////
  4                                ;/                                                      /
  5                                ;/              idb#0           mk (5 channels)         /
  6                                ;/              idb#1           auto-play (6 channels)  /
  7                                ;/              idb#p0/p1       auto percussion         /
  8                                ;/                                                      /
  9                                ;////////////////////////////////////////////////////////
 10                                ;
 11          0099                  io.vdp  =       $99             ; vdp status register
 12          00A8                  io.ppa  =       $A8             ; primary slot register
 13                                ;
 14          0090                  icall   =       $0090           ; I-call entry
 15          0093                  rcall   =       $0093           ; R-call entry
 16          0096                  kcall   =       $0096           ; K-call entry
 17          0099                  pcall   =       $0099           ; P-call entry
 18          009C                  scall   =       $009C           ; S-call entry
 19                                ;
 20          D000                  stack   =       $D000           ; stack area
 21                                ;
 22          EC32                  m.icka  =       $EC32           ; hook (irq-a)
 23          EC34                  m.ickb  =       $EC34           ; hook (irq-b)
 24          EC38                  m.iund  =       $EC38           ; hook (irq-undefined)
 25          EC3E                  m.trer  =       $EC3E           ; hook (trap-error)
 26                                ;
 27    8000                              .=$8000
 28    8000  00                    flag:   .byte   $0              ; queue/full flag
 29                                ;
 30                                ;
 31                                ;////////////////////////////////////////////////////////
 32                                ;/      Initial  Setup  Procedure                       /
 33                                ;////////////////////////////////////////////////////////
 34                                ;
 35    8001  F3                          di                        ; disable IRQ
 36    8002  ED  56                      im1                       ; IRQ-mode is <mode-1>
 37    8004  31  00  D0                  ld      sp,stack          ; define stack area
 38                                ;
 39    8007  3E  03                      ld      a,$03             ; select SFG-01 slot
 40    8009  D3  A8                      out     io.ppa            ;  SFG-01=slot#3(primary)
 41                                ;
 42    800B  CD  90  00                  call    icall             ; I-call
 43    800E  21  10  81                  ld      hl,irqa           ; define 'HOOK'
 44    8011  22  32  EC                  ld      (m.icka),hl       ;
 45    8014  21  1B  81                  ld      hl,irqb           ;
 46    8017  22  34  EC                  ld      (m.ickb),hl       ;
 47    801A  21  35  81                  ld      hl,irqv           ;
 48    801D  22  38  EC                  ld      (m.iund),hl       ;
 49    8020  21  BF  81                  ld      hl,trap           ; define 'TRAP'
 50    8023  22  3E  EC                  ld      (m.trer),hl       ;
 51                                ;
 52                                      .page
```

```
MAIN        z80.i8080 ass.vr-1.2 9-SEP-84   19:52     PAGE  2

  53                                ;////////////////////////////////////////////////////////
  54                                ;/      IDB  Setup  Procedure                           /
  55                                ;////////////////////////////////////////////////////////
  56                                ;
  57                                ;       ///  idb#bp0 ///
  58                                ;
  59        8026  3E  00            ld      a,$00           ; define idb#p0
  60        8028  06  0A            ld      b,$0A           ;
  61        802A  11  00  EB        ld      de,$EB00        ;
  62        802D  CD  9C  00        call    scall           ;
  63        8030  3E  15            ld      a,$15           ; get voice
  64        8032  01  2C  0A        ld      bc,$0A2C        ;
  65        8035  CD  9C  00        call    scall           ;
  66        8038  3E  14            ld      a,$14           ; load voice
  67        803A  06  0A            ld      b,$0A           ;
  68        803C  CD  9C  00        call    scall           ;
  69        803F  3E  13            ld      a,$13           ; set volume-balance
  70        8041  01  FF  0A        ld      bc,$0AFF        ;
  71        8044  CD  9C  00        call    scall           ;
  72                                ;
  73                                ;       ///  idb#p1  ///
  74                                ;
  75        8047  3E  00            ld      a,$00           ; define idb#p1
  76        8049  06  0B            ld      b,$0B           ;
  77        804B  11  80  EB        ld      de,$EB80        ;
  78        804E  CD  9C  00        call    scall           ;
  79        8051  3E  15            ld      a,$15           ; get voice
  80        8053  01  2D  0B        ld      bc,$0B2D        ;
  81        8056  CD  9C  00        call    scall           ;
  82        8059  3E  14            ld      a,$14           ; load voice
  83        805B  06  0B            ld      b,$0B           ;
  84        805D  CD  9C  00        call    scall           ;
  85        8060  3E  13            ld      a,$13           ; set volume-balance
  86        8062  01  FF  0B        ld      bc,$0BFF        ;
  87        8065  CD  9C  00        call    scall           ;
  88                                ;
  89                                .page
```

```
MAIN        z80.i8080 ass.vr-1.2 9-SEP-84   19:52     PAGE  3

 90                                     ;
 91                                     ;       ///  idb#1  ///
 92                                     ;
 93      8068  3E  00                            ld      a,$00           ; define idb#1
 94      806A  06  01                            ld      b,$01           ;
 95      806C  11  80  EA                        ld      de,$EA80        ;
 96      806F  CD  9C  00                        call    scall           ;
 97      8072  3E  0A                            ld      a,$0A           ; assign idb#1 to qu#1
 98      8074  06  01                            ld      b,$01           ;
 99      8076  11  00  81                        ld      de,$8100        ;
100      8079  CD  9C  00                        call    scall           ;
101      807C  3E  15                            ld      a,$15           ; get voice
102      807E  01  09  01                        ld      bc,$0109        ;
103      8081  CD  9C  00                        call    scall           ;
104      8084  3E  14                            ld      a,$14           ; load voice
105      8086  06  01                            ld      b,$01           ;
106      8088  CD  9C  00                        call    scall           ;
107      808B  3E  13                            ld      a,$13           ; set volume-balance
108      808D  01  C0  01                        ld      bc,$01C0        ;
109      8090  CD  9C  00                        call    scall           ;
110                                     ;
111                                     ;       ///  idb#0  ///
112                                     ;
113      8093  3E  00                            ld      a,$00           ; define idb#0
114      8095  06  00                            ld      b,$00           ;
115      8097  11  00  EA                        ld      de,$EA00        ;
116      809A  CD  9C  00                        call    scall           ;
117      809D  3E  0A                            ld      a,$0A           ; assign idb#0 to qu#0
118      809F  06  00                            ld      b,$00           ;
119      80A1  11  00  80                        ld      de,$8000        ;
120      80A4  CD  9C  00                        call    scall           ;
121      80A7  3E  15                            ld      a,$15           ; get voice
122      80A9  01  00  00                        ld      bc,$0000        ;
123      80AC  CD  9C  00                        call    scall           ;
124      80AF  3E  14                            ld      a,$14           ; load voice
125      80B1  06  00                            ld      b,$00           ;
126      80B3  CD  9C  00                        call    scall           ;
127      80B6  3E  13                            ld      a,$13           ; set volume-balance
128      80B8  01  FF  00                        ld      bc,$00FF        ;
129      80BB  CD  9C  00                        call    scall           ;
130
131                                             .page
```

```
MAIN       z80.i8080 ass.vr-1.2 9-SEP-84   19:52     PAGE  4

  132                           ;////////////////////////////////////////////////////////
  133                           ;/      Other  Setup                                    /
  134                           ;////////////////////////////////////////////////////////
  135                           ;
  136   80BE  3E  00                    ld      a,$00           ; define MK
  137   80C0  01  00  80                ld      bc,$8000        ;       link MK to qu#0
  138   80C3  11  00  80                ld      de,$8000        ;       velocity = 80h
  139   80C6  CD  96  00                call    kcall           ;
  140                           ;
  141   80C9  3E  09                    ld      a,$09           ; assign channels
  142   80CB  01  00  01                ld      bc,$0100        ;
  143   80CE  11  AB  00                ld      de,$00AB        ;
  144   80D1  CD  9C  00                call    scall           ;
  145                           ;
  146   80D4  3E  13                    ld      a,$13           ; select auto-rhythm queue
  147   80D6  0E  04                    ld      c,$04           ;
  148   80D8  CD  93  00                call    rcall           ;
  149   80DB  3E  14                    ld      a,$14           ; select RHB
  150   80DD  0E  00                    ld      c,$00           ;
  151   80DF  CD  93  00                call    rcall           ;
  152   80E2  3E  10                    ld      a,$10           ; start auto-rhythm
  153   80E4  0E  01                    ld      c,$01           ;
  154   80E6  16  01                    ld      d,$01           ;
  155   80E8  CD  93  00                call    rcall           ;
  156                           ;
  157   80EB  3E  00                    ld      a,$00           ; clear queue/full flag
  158   80ED  32  00  80                ld      (flag),a        ;
  159                           ;
  160                           ;
  161                           ;////////////////////////////////////////////////////////
  162                           ;/      Main  Loop                                      /
  163                           ;////////////////////////////////////////////////////////
  164                           ;
  165   80F0  FB                10$:    ei                      ; Enable IRQ
  166   80F1  3E  00                    ld      a,$00           ; P-call
  167   80F3  11  03  04                ld      de,$0403        ;
  168   80F6  CD  99  00                call    pcall           ;
  169   80F9  3E  01                    ld      a,$01           ; MK-scan
  170   80FB  CD  96  00                call    kcall           ;
  171   80FE  3A  00  80                ld      a,(flag)        ; check queue/full flag
  172   8101  A7                        and     a               ;
  173   8102  28  EC                    jrz     10$             ;
  174   8104  3E  00                    ld      a,$00           ; reset queue/full flag
  175   8106  32  00  80                ld      (flag),a        ;
  176   8109  3E  00                    ld      a,$00           ; all note off (system)
  177   810B  CD  93  00                call    rcall           ;
  178   810E  18  E0                    jr      10$             ;
  179                           ;
  180                                   .page
```

```
MAIN    z80.i8080 ass.vr-1.2 9-SEP-84   19:52     PAGE  5

  181                           ;
  182                           ;////////////////////////////////////////////////////////
  183                           ;/      IRQ  Procedure                                  /
  184                           ;////////////////////////////////////////////////////////
  185                           ;
  186                           ;       ///  irq-a  ///
  187                           ;
  188   8110  CD  2B  81        irqa:   call    regsav          ; save register
  189   8113  3E  19                    ld      a,$19           ; load opm KC
  190   8115  CD  93  00                call    rcall           ;
  191   8118  C3  21  81                jmp     regget          ; restore register
  192                           ;
  193                           ;       ///  irq-b  ///
  194                           ;
  195   811B  CD  2B  81        irqb:   call    regsav          ; save register
  196   811E  CD  3B  81                call    auto            ; auto-rhythm & playback
  197   8121  FD  E1            regget: pop     iy              ; restore register
  198   8123  DD  E1                    pop     ix              ;
  199   8125  D1                        pop     de              ;
  200   8126  C1                        pop     bc              ;
  201   8127  F1                        pop     af              ;
  202   8128  E1                        pop     hl              ;
  203   8129  FB                        ei                      ;
  204   812A  C9                        ret                     ;
  205                           ;
  206   812B  E3                regsav: ex      (sp),hl         ; save register
  207   812C  F5                        push    af              ;
  208   812D  C5                        push    bc              ;
  209   812E  D5                        push    de              ;
  210   812F  DD  E5                    push    ix              ;
  211   8131  FD  E5                    push    iy              ;
  212   8133  E5                        push    hl              ;
  213   8134  C9                        ret                     ;
  214                           ;
  215                           ;       ///  irq-vdp  ///
  216                           ;
  217   8135  F5                irqv:   push    af              ; save register
  218   8136  DB  99                    in      io.vdp          ; reset vdp-irq
  219   8138  F1                        pop     af              ; restore register
  220   8139  FB                        ei                      ;
  221   813A  C9                        ret                     ;
  222                           ;
  223                                   .page
```

```
MAIN      z30.i8080 ass.vr-1.2 9-SEP-84   19:52     PAGE  6

 224                                  ;
 225   813B  3E  11                  auto:   ld      a,$11           ; set auto-rhythm clock
 226   813D  CD  93  00                       call    rcall           ;
 227   8140  FE  FF                          cpi     $FF             ;   already started ?
 228   8142  C8                             ret     z               ;
 229   8143  16  00                          ld      d,$00           ; get event for idb#1
 230   8145  5F                             ld      e,a             ;
 231   8146  21  5F  81                     ld      hl,100$         ;
 232   8149  19                             add     hl,de           ;
 233   814A  7E                             ld      a,(hl)          ;
 234   814B  E6  FF                          and     $FF             ;   event exist ?
 235   814D  C8                             ret     z               ;
 236   814E  06  01                          ld      b,$01           ; set event into qu#1
 237   8150  57                             ld      d,a             ;       on/off, kc#
 238   8151  1E  80                          ld      e,$80           ;       velocity
 239   8153  3E  02                          ld      a,$02           ;
 240   8155  CD  93  00                       call    rcall           ;
 241   8158  D0                             ret     nc              ;   queue#1 full ?
 242   8159  3E  FF                          ld      a,$FF           ; set queue/full flag
 243   815B  32  00  80                       ld      (flag),a        ;
 244   815E  C9                             ret                     ;
 245                                  ;
 246   815F  BE  00  00  00  00     100$:   .byte   $BE,$00,$00,$00,$00,$00
 247   8165  00  00  3E  C8                 .byte   $00,$00,$3E,$C8,$00,$00
 248   816B  00  00  00  00                 .byte   $00,$00,$00,$00,$00,$48
 249   8171  BE  00  00  00                 .byte   $BE,$00,$00,$00,$00,$3E
 250   8177  BE  00  00  00                 .byte   $BE,$00,$00,$00,$00,$00
 251   817D  00  00  3E  C8                 .byte   $00,$00,$3E,$C8,$00,$00
 252   8183  00  00  00  00                 .byte   $00,$00,$00,$00,$00,$48
 253   8189  BE  00  00  00                 .byte   $BE,$00,$00,$00,$00,$3E
 254                                  ;
 255   818F  BE  00  00  00                 .byte   $BE,$00,$00,$00,$00,$00
 256   8195  00  00  3E  C8                 .byte   $00,$00,$3E,$C8,$00,$00
 257   819B  00  00  00  00                 .byte   $00,$00,$00,$00,$00,$48
 258   81A1  BE  00  00  00                 .byte   $BE,$00,$00,$00,$00,$3E
 259   81A7  BE  00  00  00                 .byte   $BE,$00,$00,$00,$00,$00
 260   81AD  00  00  3E  C8                 .byte   $00,$00,$3E,$C8,$00,$00
 261   81B3  00  00  00  00                 .byte   $00,$00,$00,$00,$00,$48
 262   81B9  BE  00  00  00                 .byte   $BE,$00,$00,$00,$00,$3E
 263                                  ;
 264                                  ;
 265                                  ;       ///  trap (error)  ///
 266                                  ;
 267   81BF  7A                     trap:   ld      a,d             ; check queue/full ?
 268   81C0  E6  F0                          ani     $F0             ;
 269   81C2  FE  10                          cpi     $10             ;
 270   81C4  C0                             ret     nz              ;
 271   81C5  32  00  80                       ld      (flag),a        ; set queue/full flag
 272   81C8  C9                             ret                     ;
 273                                  ;
 274                                          .end
```

5.2 Supplementary explanation for recording and playback

The following outlines the procedures necessary to carry out recording and playback.

-Define necessary IDB's and associated queues.
-Define EVB via S-02 call
-Initialize EVB via S-04 call

-Use R-0c call for time clock for playback
-Use R-09 call for time clock of recording
-Clock can be free running, regardless of whether recording or playback is is actually taking place.
-Start recording via R-08 call
-End recording via R-0a call
-Recording will also end when All-note-off is issued into queue, or when EVB gets filled (Trap by <M.TRER> of the MIDB will also occure if so defined).

-Start playback via R-0B call
-End playback via R-0D call
-Playback will also end when the end of data in the EVB is encountered (Trap by <M.TRER> of MIDB will also occure if so defined)

5-3 Supplementary explanation for Auto-rhythm

The following procedures are necessary to carry out the auto-rhythm performance.

-Define necessary IDB's.
-Define RHB (rhythm buffer) by writing the pointers to RHB in <T.RHY1> (for chord/bass) and <T.RHY2> (for percussion) of MIDB.
These are 32 byte tables for each, serving 16 entry pointers each.
0000h means that corrsponding RHB is not defined.
-Specify time signature by writing it into <T.RHY0> of MIDB.
There are 16 entries for 16 RHB's.

-Use an R-11 call for auto rhythm clock.
-The rhythm clock can be free running.
-An R-11 call will return the current pointer in the RHB, so that it may be used to synchronize the particular accent of the rhythm to another processes.

-Use the R-13 call to specify which queues (QU*C, QU*B, QU*P) are to be used.
-Use the R-04 or R-05 call to send event to the CHORD-KB.
-The CHORD-KB will provide note information to the chord generator and the walking bass-line genarator, as well as providing events to QU*C.
-Use the R-14 call to select the RHB.

-Start auto rhythm via an R-10 call.
-Stop the rhythm via an R-12 call.

5-4 Supplementary information for the CMT usage

-Use the S-03 call to define the UVL.

-Use the S-21 call to load the UVL into memory (from the CMT).
-Use S-22 call to save the UVL to the CMT
-For the UVL load/save, the file name is fixed to "VOICE".

-Use the S-23 call to load the EVB into memory.
-Use the S-24 call to save the EVB to the CMT.
-The file name for the EVB can be specified via <M.FEVB> of the MIDB.

5-5 Monitor usage

Even though M-monitor can be invoked via BASIC's "CALL MUSIC", it can also be called by an assembler program.
Once called by the user program, depressing the <ESC> key will return the control to the user again.
Calling sequence:

```
DI
CALL    00A2h
```

Some points worth noting are:

-M-monitor will use a user supplied stack.
-In calling M-Monitor, the folowing IDB's must have been assigned at the shown address.

IDB#0	E900h
IDB#1	E980h
IDB#C	EA00h
IDB#B	EA80h
IDB#P0	EB00h
IDB#P1	EB80h

-Define the UVL so that M-monitor can use it.
-Define the EVB so that M-monitor can use it, and this case, be sure to initialize it (use S-04 call).
For 16K RAM system, reserve at least 4K bytes for the EVB.
For 32K RAM system, reserve at least 8K bytes for the EVB.

5-6 Problems and solutions

1. LFO synchronization

   To ensure that the LFO functions properly, immediately before the P-call, move msb of V.PMD (internal offset in IDB is 1Ah) to msb of Y.SYNC (internal offset in IDB is 0Ah).
   It is recommended to carry out this procedure each time the P-call is issued.

2. Disabling noise

   When shifting from the noise-enabled state to the noise disabled state, set noise-disable on <M.NOIS> of MIDB and issue an R-18.

3. Disabling clock-A and clock-B.

   -Make sure that address 0087h is 00h.
   -Write 15h into address 3FF0h.
   -Write 00h into address 3FF1h.

   Note that clock-A and clock-B will be enabled again by an I-call.

4. Statement call from BASIC

   When "extension call" is made to MBIOS from MSX-BASIC, MBIOS does not retain the original contents of the HL registers.
   Because of this, the Muisc Macro program will not run if the cartridge is inserted in the slot after the SFG-01. The Music Macro program should always be placed in a lower alot than the SFG-01.

5. Output level after I-call

   Even after an I-call has been issued, sometimes the output level may not necessarily decrease, thereby causing sound to still be heard .
   To make sure of complete silence, following an I-call, assign every channel to the IDB, and issue an All-note-off (S-0B call) to the IDB.

6. Only a single event will be processed at a time for P-calls corresponding to QU*C, QU*B, and QU*P.

7. During the playback processing, the very last event in the EVB will not be processed.

8. Velocity

If velocity is 00h or odd number (its lsb is on), normal operation may not be possible during MIDI output.
It is recommended to avoid these values for velocity.

9. The problem of IRQ flag

When R, K, P and S - calls are made under IRQ-enabled mode, the interrupt status might be reset to the disabled mode when the service is completed.
After the call, if so desired, it is a good practice to set IRQ enable again.

10. Changing the KC (Key code) range

Prior to changing the KC range, make sure to issue All-Note-Off command to the designated IDB.